Klasszikus programozás

Java nyelven

I.

Bevezető gyakorlatok

Eclipse (Neon, Oxygen)

Klement András

2016 – 2018

Utolsó tartalmi módosítás: 2018. május 4.

Naplemente

Az arany nap lankadva ballag, parazsa hamvad.

Tűzgolyónk süllyedő útja nyírfák mögé bukik.

Meleg est lehelete terjed szerte felettem.

A hűsítő tó fölé ezüst holdsugár kúszik.

Hatalmas vadalmafa alatt

csendesen mereng egy delejes

tündérszárnyú kígyóbűvölő.

Körmend, 2018. február 11.

Klement András

Tartalomjegyzék

1. Algoritmikus alapszerkezetek: elágazások és ciklusok 4

2. Adatszerkezetek: számok, karakterek, szöveg és tömb 12

3. Beolvasás, kiírás: a képernyő és a szöveges fájlok kezelése 20

4. Elemi algoritmusok: összegzés, eldöntés, keresés, megszámolás, maximum-kiválasztás és kiválogatás ... 30

5. Rendezések: Egyszerű cserés-, minimum-kiválasztásos-, buborékos- és beillesztéses rendezés 36

6. Rendezés két szempont szerint: két- ill. egy lépésben 40

7. A maximumkiválasztás általánosítása: versenykiértékelések 46

8. A leghosszabb adott tulajdonságú részsorozat keresése:

a leghosszabb prímsorozat egy véletlen sorozatban 50

9. Ismétlődés nélküli véletlen sorozat előállítása: lottóhúzás 53

10. Statisztikai minta gyakorisági táblázata: érték szerinti indexelés és általános módszer 57

11. Számelméleti feladatok: euklidészi algoritmus, prímfelbontás, osztók száma 61

12. Rekurzió: Fibonacci-sorozat, zárójelezés érvényessége 67

13. Verselemzés: Üllői-úti fák 71

14. Érettségi mintafeladat: Autók 84

Folytatás: Klasszikus programozás Java nyelven

II. Az emelt szintű informatika érettségi programozási feladatainak megoldása 94

1. Algoritmikus alapszerkezetek: elágazások és ciklusok

```java
/**
 * Algoritmikus alapszerkezetek: elágazások és ciklusok
 *
 * @author Klemand
 */
public class BevGyak01 {

	public static void main(String[] args) {
		// main + CTRL space

		System.out.println("1. Feltételes utasítás függvényhívással\n");
		/*
		 * Kiíratás a képernyőre: sysout + CTRL space println: soremeléssel print:
		 * soremelés nélkül plusz sortörés beszúrása: \n
		 */

		System.out.println("Egy kockadobást szimulálunk.");
		int k = kockadobas();
		// Függvényhívás. A függvény a main metódus után található.

		System.out.println("A dobásod: " + k);
		if (k == 6) {
			/*
			 * a=5: értékadás
			 * a==5: az egyenlőség vizsgálata
			 */
			System.out.println("Gratulálok! Szerencsés ember vagy.");
		}

		vonal();
		// Eljáráshívás. Az eljárás a main metódus után található.

		System.out.println("2. Kétirányú elágazás \n");

		System.out.println("Egy kockadobást szimulálunk.");
		k = kockadobas();
		System.out.println("A dobásod: " + k);
		if (k == 6) {
			System.out.println("Ne bízd el magad! Egyszer fenn, máskor lenn.");
		} else {
			System.out.println("Ne csüggedj! A szerencse forgandó.");
		}

		vonal();

		System.out.println("3. Egymásba ágyazott elágazások \n");
		System.out.println("Egy kockadobást szimulálunk.");

		k = kockadobas();
		System.out.println("A dobásod: " + k);
		if (k == 6) {
			System.out.println("Mindig törekedj a maximumra, de maradj szerény!");
		} else if (k == 4 || k == 5) {
			// "vagy" feltétel megadása
			System.out.println("Csak " + (6 - k) + " hiányzott!");
		} else if (k < 4 && k % 2 == 1) {
			// "és" feltétel megadása, ill. osztási maradék meghatározása
			System.out.println("Kicsi a bors, de erős!");
		} else {
			System.out.println("Ez igen! Az egyetlen páros prímszámot sikerült dobnod!");
		}
		vonal();
```

```
System.out.println("4. Többirányú elágazás \n");

System.out.println("Egy kockadobást szimulálunk.");
k = kockadobas();
System.out.println("A dobásod: " + k);

switch (k) {
case 6: {
        System.out.println("Mondd csak, nem csalsz?");
        break;
}
case 4: {
        System.out.println("Ez éppen annyi, mint a 28/6 egész osztás eredménye: " + 28 / 6);
        break;
}
case 1: {
        System.out.println("Szerencsére ez most nem osztályzat!");
        break;
}
default:
        System.out.println("Nagyszerű! Prímszámot dobtál.");
}

vonal();

System.out.println("5. Egyszerű számlálós ciklus formázás nélkül \n");

int i;
int negyzet;
int recint;
double recdouble;
// Számlálós ciklus: kezdőérték, bennmaradási feltétel, léptetés
for (i = 1; i <= 12; i++) {
        // A ciklusmag kezdete: {
        negyzet = i * i;
        recint = 1 / i;
        recdouble = (double) 1 / i;
        // Típuskényszerítés
        System.out.println(
                        i + " négyzete: " + negyzet + ", egész reciproka: " + recint
                        + ", valós reciproka: " + recdouble);
} // A ciklusmag vége: }
System.out.println("");

// Automatikus formázás: Shift CTRL F

vonal();

System.out.println("6. Formázással \n");

// A változókat nem kell újra deklarálni
for (i = 1; i <= 12; i++) {
        negyzet = i * i;
        recint = 1 / i;
        recdouble = (double) 1 / i;
        System.out.printf("%2d", i);
        /*
         * Szélesség megadása. A formázás után vessző, és a formázott változó lehet
         * csak. a printf sem emel sort
         */
        System.out.printf(" négyzete: %3d", negyzet);
        System.out.printf(",\t egész reciproka: %2d", recint);
        // Tabulátor beszúrása: \t
        System.out.printf(",\t valós reciproka: %.3f", recdouble);
        // Tizedesjegyek számának megadása
        System.out.println("");
}
vonal();
```

```
System.out.println("7. Egymásba ágyazott számlálós ciklus \n");

int j;

for (i = 1; i <= 14; i++) {
        // Külső ciklus: 14 sor

        System.out.print("xxxxxx----------------------------------------");
        // Ez minden sorban kiíratódik

        for (j = 1; j <= i; j++) {
                // Az i-edik sorban plusz i db + kiíratása a belső ciklussal
                System.out.print("+");
        }
        System.out.println("");
}

for (i = 13; i >= 1; i--) {
        // 13 sor

        System.out.print("xxxxxx----------------------------------------");

        for (j = 1; j <= i; j++) {
                System.out.print("+");
        }
        System.out.println("");
}

vonal();

System.out.println("8. Szorzótábla \n");

System.out.print("*" + "\t");
for (j = 1; j <= 10; j++) {

        System.out.print(j + "\t");
}
System.out.println("\n");
for (i = 1; i <= 10; i++) {
        // Külső ciklus: 10 sor
        System.out.print(i + "\t");
        for (j = 1; j <= 10; j++) {
                // Belső ciklus: 10 oszlop
                System.out.print(i * j + "\t");
        }
        System.out.println("");
}

vonal();
```

```
System.out.println("9. Elöltesztelő ciklus számlálóval \n");
System.out.println("Addig íratjuk ki a számok négyzetét, amíg el nem éri a 100-at \n");

i = 1;
// A számláló kezdőértékének megadása
while (i * i < 100) {
        // Belépési feltétel megadása
        System.out.printf(i + "^2 = %2d", i * i);
        System.out.println("");
        i++;
        /*
         * A számláló értékét a ciklusmagban kell módosítani, különben végtelen ciklus
         * jön létre!
         */
}

vonal();

System.out.println("10. Elöltesztelő ciklus számláló nélkül \n");
System.out.println("Addig lépünk be a ciklusba, amíg az előállított véletlenszám nagyobb 0,5-nél \n");

double v;
while ((v = Math.random()) > 0.5) {
        // Belépési feltétel megadása
        System.out.printf("%.4f", v);
        System.out.println("");
}
// Lehet, hogy egyszer sem lép be

vonal();

System.out.println("11. Elöltesztelő ciklus függvényhívással \n");
System.out.println("Addig lépünk be a ciklusba, amíg legalább 4-est dobunk \n");

int sdb = 0;
while ((k = kockadobas()) >= 4) {
        // Belépési feltétel megadása
        sdb++;// a sikeres dobások számolása
        System.out.println(sdb + ". dobás: " + k);
}
System.out.println("A sikeres dobások száma: " + sdb);

vonal();

System.out.println("12. Hátultesztelő ciklus \n");
// Egyszer mindenképpen végrehajtódik
System.out.println("Addig próbálkozunk, amíg hatost nem sikerül dobnunk \n");
int db = 0;
do {
        db++;
        k = kockadobas();
        System.out.println(k);
} while (k < 6);
System.out.println("A próbálkozások száma: " + db);

vonal();
```

```
                System.out.println("12+1. Rajzoljunk! \n");

                for (i = 1; i <= 10; i++) {
                        for (j = 1; j <= 39; j++) {
                                if (j == 20 - (i - 1) || j == 20 + (i - 1)) {
                                        System.out.print("M");
                                } else {
                                        System.out.print(" ");
                                }
                        }
                        System.out.println("");
                }

                for (i = 11; i <= 20; i++) {
                        for (j = 1; j <= 39; j++) {
                                if (j == 20 - (i - 1) || j == 20 + (i - 1)) {
                                        System.out.print("M");
                                } else if (j == 20 - 2 * (i - 11) || j == 20 + 2 * (i - 11)) {
                                        System.out.print("W");
                                } else {
                                        System.out.print(" ");
                                }
                        }
                        System.out.println("");
                }

                vonal();
        }

        // Eljárások, függvények a fő metódus után
        public static void vonal() {
                // Eljárás: void típusú, nincs visszaadott érték

        System.out.println("\n================================================================================ \n");
        }

        public static int kockadobas() {
                // Függvény: a visszaadott érték típusának megadásával
                int dobas = (int) (Math.random() * 6) + 1;
                // Típuskényszerítés
                return dobas;
                // A visszaadott érték
        }

}
```

```
1. Feltételes utasítás függvényhívással

Egy kockadobást szimulálunk.
A dobásod: 4

================================================================================

2. Kétirányú elágazás

Egy kockadobást szimulálunk.
A dobásod: 5
Ne csüggedj! A szerencse forgandó.

================================================================================

3. Egymásba ágyazott elágazások

Egy kockadobást szimulálunk.
A dobásod: 1
Kicsi a bors, de erős!

================================================================================

4. Többirányú elágazás

Egy kockadobást szimulálunk.
A dobásod: 6
Mondd csak, nem csalsz?

================================================================================

5. Egyszerű számlálós ciklus formázás nélkül

1 négyzete: 1, egész reciproka: 1, valós reciproka: 1.0
2 négyzete: 4, egész reciproka: 0, valós reciproka: 0.5
3 négyzete: 9, egész reciproka: 0, valós reciproka: 0.3333333333333333
4 négyzete: 16, egész reciproka: 0, valós reciproka: 0.25
5 négyzete: 25, egész reciproka: 0, valós reciproka: 0.2
6 négyzete: 36, egész reciproka: 0, valós reciproka: 0.16666666666666666
7 négyzete: 49, egész reciproka: 0, valós reciproka: 0.14285714285714285
8 négyzete: 64, egész reciproka: 0, valós reciproka: 0.125
9 négyzete: 81, egész reciproka: 0, valós reciproka: 0.1111111111111111
10 négyzete: 100, egész reciproka: 0, valós reciproka: 0.1
11 négyzete: 121, egész reciproka: 0, valós reciproka: 0.09090909090909091
12 négyzete: 144, egész reciproka: 0, valós reciproka: 0.08333333333333333


================================================================================

6. Formázással

 1 négyzete:   1,        egész reciproka:  1,    valós reciproka: 1,000
 2 négyzete:   4,        egész reciproka:  0,    valós reciproka: 0,500
 3 négyzete:   9,        egész reciproka:  0,    valós reciproka: 0,333
 4 négyzete:  16,        egész reciproka:  0,    valós reciproka: 0,250
 5 négyzete:  25,        egész reciproka:  0,    valós reciproka: 0,200
 6 négyzete:  36,        egész reciproka:  0,    valós reciproka: 0,167
 7 négyzete:  49,        egész reciproka:  0,    valós reciproka: 0,143
 8 négyzete:  64,        egész reciproka:  0,    valós reciproka: 0,125
 9 négyzete:  81,        egész reciproka:  0,    valós reciproka: 0,111
10 négyzete: 100,        egész reciproka:  0,    valós reciproka: 0,100
11 négyzete: 121,        egész reciproka:  0,    valós reciproka: 0,091
12 négyzete: 144,        egész reciproka:  0,    valós reciproka: 0,083

================================================================================
```

7. Egymásba ágyazott számlálós ciklus

```
xxxxxx----------------------------------------+
xxxxxx----------------------------------------++
xxxxxx----------------------------------------+++
xxxxxx----------------------------------------++++
xxxxxx----------------------------------------+++++
xxxxxx----------------------------------------++++++
xxxxxx----------------------------------------+++++++
xxxxxx----------------------------------------++++++++
xxxxxx----------------------------------------+++++++++
xxxxxx----------------------------------------++++++++++
xxxxxx----------------------------------------+++++++++++
xxxxxx----------------------------------------++++++++++++
xxxxxx----------------------------------------+++++++++++++
xxxxxx----------------------------------------++++++++++++++
xxxxxx----------------------------------------+++++++++++++
xxxxxx----------------------------------------++++++++++++
xxxxxx----------------------------------------+++++++++++
xxxxxx----------------------------------------++++++++++
xxxxxx----------------------------------------+++++++++
xxxxxx----------------------------------------++++++++
xxxxxx----------------------------------------+++++++
xxxxxx----------------------------------------++++++
xxxxxx----------------------------------------+++++
xxxxxx----------------------------------------++++
xxxxxx----------------------------------------+++
xxxxxx----------------------------------------++
xxxxxx----------------------------------------+
```

==========

8. Szorzótábla

*	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

==========

9. Elöltesztelő ciklus számlálóval

Addig íratjuk ki a számok négyzetét, amíg el nem éri a 100-at

```
1^2 =  1
2^2 =  4
3^2 =  9
4^2 = 16
5^2 = 25
6^2 = 36
7^2 = 49
8^2 = 64
9^2 = 81
```

==========

```
10. Elöltesztelő ciklus számláló nélkül

Addig lépünk be a ciklusba, amíg az előállított véletlenszám nagyobb 0,5-nél

0,5256
0,6619
0,5375

================================================================================

11. Elöltesztelő ciklus függvényhívással

Addig lépünk be a ciklusba, amíg legalább 4-est dobunk

1. dobás: 4
2. dobás: 4
3. dobás: 5
A sikeres dobások száma: 3

================================================================================

12. Hátultesztelő ciklus

Addig próbálkozunk, amíg hatost nem sikerül dobnunk

4
2
6
A próbálkozások száma: 3

================================================================================

12+1. Rajzoljunk!

                    M
                   M M
                  M   M
                 M     M
                M       M
               M         M
              M           M
             M             M
            M               M
           M                 M
          M         W         M
         M        W   W        M
        M       W       W       M
       M      W           W      M
      M     W               W     M
     M    W                   W    M
    M   W                       W   M
   M  W                           W  M
  M W                               W M
 MW                                   WM

================================================================================
```

2. Adatszerkezetek: számok, karakterek, szöveg és tömb

```
/**
 * Adatszerkezetek: számok, karakterek, szöveg és tömb
 *
 * @author Klemand
 */
public class BevGyak02 {

        public static void main(String[] args) {

                System.out.println("1. Egész számok \n   ============");

                System.out.println("Két 1 és 100 közötti véletlen egész szám előállítása: ");

                int m = veletlenEgesz();
                // Egész típusú szám (32 bites) deklarálása értékadással
                int n = veletlenEgesz();
                int max = Math.max(m, n);
                int min = Math.min(m, n);

                System.out.println("Az első egész szám: " + n + ", a második pedig: " + m);
                System.out.println(max + " a nagyobb.");
                System.out.println("Összegük: " + max + " + " + min + " = " + (max + min));
                System.out.println("Szorzatuk: " + max + " * " + min + " = " + max * min);
                System.out.println("Egész osztásuk eredménye: " + max + " / " + min + " = " + max / min);
                System.out.println("Maradékos osztásuk eredménye: " + max + " % " + m + " = " + max % min);
                System.out.print("Valós osztásuk eredménye 4 tizedesjegyre kerekítve: " + max + " / " + min + " = ");
                System.out.printf("%.4f \n", (double) max / min);
                // típuskényszerítés
                System.out.println("");

                long fakt;
                // Hosszú egész típusú szám (64 bites) deklarálása értékadás nélkül

                fakt = 1;
                int i;
                for (i = 1; i <= 20; i++) {
                        fakt = fakt * i;
                        // Rövidebben: fakt*=i;
                }

                System.out.println("20! = " + fakt);
                System.out.println("");

                System.out.println("Figyelni kell a túlcsordulásra!");
                fakt = 1;
                for (i = 1; i <= 21; i++) {
                        fakt = fakt * i;
                }
                System.out.println("21! = " + fakt);
                System.out.println("");

                System.out.println("2. Valós számok \n   ============");

                System.out.println("Két 0 és 100 közötti véletlen valós szám előállítása: ");

                double u = veletlenValos();
                // Dupla pontosságú valós típusú szám (64 bites) deklarálása
                double v = veletlenValos();

                System.out.printf("Az első valós szám 3 tizedes pontossággal: u =  %.3f", u);
                System.out.printf(", a második pedig: v =  %.3f", v);
                System.out.println("");
```

```
System.out.printf("Összegük: %.3f", u);
System.out.printf(" + %.3f", v);
System.out.printf(" = %.3f \n", (u + v));
// Kiíratás esetén összeadásnál kötelező a zárójel!
System.out.printf("Szorzatuk 3 tizedesjeggyel: %.3f", u);
System.out.printf(" * %.3f", v);
System.out.printf(" = %.3f \n", u * v);
System.out.printf("Különbségük abszolút értéke 3 tizedesjeggyel: |%.3f", u);
System.out.printf(" - %.3f", v);
System.out.printf("| = %.3f \n", Math.abs(u - v));
System.out.printf("Hányadosuk 3 tizedesjeggyel: %.3f", u);
System.out.printf(" / %.3f", v);
System.out.printf(" = %.3f \n", u / v);
System.out.print("Az első szám köbének és a második négyzetgyökének szorzata
                 3 tizedesjegyre kerekítve: ");
System.out.printf("%.3f \n", Math.pow(u, 3) * Math.sqrt(v));
System.out.println("");

System.out.println("3. Karakterek és sztringek \n   ======================");

char k = '*';

System.out.println("A * karakter ASCII-kódja: " + (int) k);

System.out.println("\nAz ASCII-kódokhoz tartozó karakterek");

for (i = 32; i <= 370; i++) {
        System.out.print(i + ": " + (char) i + "\t");
        if ((i - 31) % 10 == 0) { // 10 oszlopos kiíratás
                System.out.println("");
        }
}
System.out.println("\n");

System.out.println("A Tündérszárnyú Kígyóbűvölő kifejezés karaktereinek ASCII-kódjai:");
String ekezetes = "Tündérszárnyú Kígyóbűvölő";
// A String osztály, ezért a deklarációja nagykezdőbetűs

for (i = 1; i <= ekezetes.length(); i++) {
        System.out.print((int) ekezetes.charAt(i - 1) + " ");
        // Az indexelés mindig 0-val kezdődik, ezért van a -1-es eltolás
}

System.out.println("\n");

System.out.println("Összefűzés:");

System.out.println("" + k + k + k + ekezetes + k + k + k);
// Fontos üres Stringgel kezdeni a kiíratást, különben a * kódját háromszorozza
// az elején!
System.out.println("");

System.out.println("Megfordítás a charAt fügvénnyel:");
String ford = "";

for (i = ekezetes.length(); i >= 1; i--) {
        ford += ekezetes.charAt(i - 1);
}
System.out.println(ford);
System.out.println("");

System.out.println("Visszafordítás a substring fügvénnyel:");
String vissza = "";
```

```
for (i = ford.length(); i >= 1; i--) {
        vissza += ford.substring(i - 1, i);
        // A substring eredménye sosem karakter, mindig string!
}
System.out.println(vissza);
System.out.println("");

System.out.println("Nagybetűssé alakítás:");
System.out.println(ekezetes.toUpperCase());
System.out.println("");

System.out.println("Kisbetűssé alakítás:");
System.out.println(ekezetes.toLowerCase() + "\n");

System.out.println("Darabolás:");
ekezetes = ekezetes.trim();
// A trim() függvény levágja a szöveg elején, ill. végén lévő esetleges
// szóközöket

String[] daraboltEkezetes = ekezetes.split(" ");
// A split függvény az idézőjelek közt megadott szóköz karakter szerint
// darabolja fel a Stringet
System.out.println("A kifejezés " + daraboltEkezetes.length + " szóból áll.");
// A String hossza: length(), a String darabok száma length

for (i = 1; i <= daraboltEkezetes.length; i++) {
        System.out.println("Az " + i + ". szó: " + daraboltEkezetes[i - 1]);
}

System.out.println("Összehasonlítjuk a két szót.");
System.out.println("Az első szó a szöveg, a második a compareTo paramétere.");
System.out.println("Ha az érték 0, akkor egyenlőek, ha pozitív, akkor a szöveg a nagyobb,");
System.out.println("ha negatív, akkor a paraméter:");
System.out.println(daraboltEkezetes[0].compareTo(daraboltEkezetes[1]) + "\n");

System.out.println("Mindkét szóból véletlenszerűen kiválasztunk egy karaktert,");
System.out.println("és összehasonlítjuk az ábécében elfoglalt helyzetüket: \n");

int elso = veletlenParameteres(daraboltEkezetes[0].length());
int masodik = veletlenParameteres(daraboltEkezetes[1].length());

String elsoKar = daraboltEkezetes[0].substring(elso - 1, elso);
String masodikKar = daraboltEkezetes[1].substring(masodik - 1, masodik);
/*
 * A kiválasztott karakterek String típusúak! Az (int) típuskényszerítéssel csak
 * a karakter típus ASCII-kódja kapható meg! De az ékezetes betűk kódjai miatt
 * az ASCII-kód nem is alkalmas az összehasonlításra!
 */
System.out.println("Az első szóból kiválasztott karakter: " + elsoKar);
System.out.println("A második szóból kiválasztott karakter: " + masodikKar);

if (elsoKar.equals(masodikKar)) {
        /*
         * Stringek egyezésének vizsgálata Stringek esetén az == nem alkalmazható, csak
         * az equals függvény!
         */
        System.out.println("A két kiválasztott karakter megegyezik");
} else if (abcsorrend(elsoKar, masodikKar)) {
        System.out.println("Az első szóból kiválasztott karakter van előrébb az ábécében.");
} else {
        System.out.println("A második szóból kiválasztott karakter van előrébb az ábécében.");
}
System.out.println("");
```

```
System.out.println("4. Egydimenziós tömbök \n   ==================");

System.out.println("Egy megadott pozitív egészeket tartalmazó tömb deklarálása \n");

int[] X = { 12, 121, 34, 2, 67, 49, 30, 122, 800, 1256, 2016, 3, 21, 77, 4, 28, 84, 1, 156, 56 };
int hossz = X.length;

System.out.println("A tömb elemeinek kiíratása for ciklussal:");
for (i = 1; i <= hossz; i++) {
        System.out.print(X[i - 1] + " ");
}
System.out.println("\n");

System.out.println("1 és 100 közötti véletlen pozitív egészeket tartalmazó tömb előállítása \n");

int[] Y = new int[30];
// Deklaráláskor meg kell adni a tömb maximális méretét!

for (i = 1; i <= 20; i++) {
        Y[i - 1] = veletlenEgesz();
}
System.out.println("A tömb elemeinek kiíratása for ciklussal:");

for (i = 1; i <= Y.length; i++) {
        /*
         * Ha nem tudjuk pontosan, hogy hány eleme lesz a tömbnek, akkor a length
         * helyett célszerűbb a mindig megszámolni az elemeket, különben zavaró 0-kat ír
         * a végére!
         */
        System.out.print(Y[i - 1] + " ");
}
System.out.println("\n");

System.out.println("A mintaszöveg karaktereinek String típusú elemekből álló tömbje:");
String[] ekezetesTomb = new String[ekezetes.length()];
for (i = 1; i <= ekezetes.length(); i++) {
        ekezetesTomb[i - 1] = ekezetes.substring(i - 1, i);
}
for (i = 1; i <= ekezetes.length(); i++) {
        System.out.print(ekezetesTomb[i - 1] + " ");
}
System.out.println("\n\nKiíratás nagy-, ill. kisbetűs párokkal:");

for (i = 1; i <= ekezetes.length(); i++) {
        System.out.print(ekezetesTomb[i - 1].toUpperCase() + ekezetesTomb[i - 1].toLowerCase() + " ");
        // A tömb elemei Stringek, alkalmazhatók rájuk a String függvények.
}
```

```
		System.out.println("\n\nA tömb magánhangzóinak kicserélése távirati formára:");

		for (i = 1; i <= ekezetes.length(); i++) {
			if (maganhangzoE(ekezetesTomb[i - 1])) {
				switch (ekezetesTomb[i - 1]) {
				case "á":
					ekezetesTomb[i - 1] = "aa";
					break;
				/*
				 * Break nélkül végrehajtaná a következő utasítást is, így mindig "uue" lenne az
				 * eredmény!
				 */
				case "é":
					ekezetesTomb[i - 1] = "ee";
					break;
				case "í":
					ekezetesTomb[i - 1] = "ii";
					break;
				case "ó":
					ekezetesTomb[i - 1] = "oo";
					break;
				case "ú":
					ekezetesTomb[i - 1] = "uu";
					break;
				case "ö":
					ekezetesTomb[i - 1] = "oe";
					break;
				case "ü":
					ekezetesTomb[i - 1] = "ue";
					break;
				case "ő":
					ekezetesTomb[i - 1] = "ooe";
					break;
				case "ű":
					ekezetesTomb[i - 1] = "uue";
				}
			}
		}

		for (i = 1; i <= ekezetes.length(); i++) {
			System.out.print(ekezetesTomb[i - 1] + " ");
		}

		System.out.println("\n");

		System.out.println("5. Kétdimenziós tömbök \n   ====================");

		System.out.println("1000 db 1 és 75 közötti véletlen egész számhármas előállítása,");
		System.out.println("és tárolása kétdimenziós tömbben");
		System.out.println("A számhármasok hány százaléka határoz meg háromszöget?\n");

		int[][] szamharmasok = new int[1000][3];

		for (i = 1; i <= 1000; i++) {
			for (int j = 1; j <= 3; j++) {
				szamharmasok[i - 1][j - 1] = veletlenParameteres(75);
			}
		}

		System.out.println(
				"Az első számhármas: " + szamharmasok[0][0] + ", " + szamharmasok[0][1]
						+ ", " + szamharmasok[0][2]);

		System.out.println("Az utolsó számhármas: " + szamharmasok[999][0] + ", " + szamharmasok[999][1] +
", "
				+ szamharmasok[999][2]);
```

```
                int hszdb = 0;
                for (i = 1; i <= 1000; i++) {
                        if (haromszogE(szamharmasok[i - 1][0], szamharmasok[i - 1][1], szamharmasok[i - 1][2])) {
                                hszdb++;
                        }
                }
                System.out.printf("A háromszöget meghatározók aránya: %.1f", ((double) hszdb) / 10);
                // Először a hszdb-t kell valóssá alakítani, csak azután szabad osztani! (/1000
                // és *100)
                System.out.println("%\n");

        }

        // Eljárások, függvények
        public static int veletlenEgesz() {
                // Fix felső határ
                int egesz = (int) (Math.random() * 100) + 1;
                // Típuskényszerítés
                return egesz;
        }

        public static double veletlenValos() {
                double valos = Math.random() * 100;
                // 0 <= valos < 100
                return valos;
        }

        public static int veletlenParameteres(int h) {
                // A függvény paraméterként kapja meg a felső határt
                int egesz = (int) (Math.random() * h + 1);
                return egesz;
        }

        public static boolean abcsorrend(String a, String b) {
                String abc = "aábcdeéfghiíjklmnoóöőpqrstuúüűvwxyz";
                return abc.indexOf(a) < abc.indexOf(b);
        }

        public static boolean maganhangzoE(String a) {
                String mgh = "aáeéiíoóöőuúüű";
                return mgh.contains(a);
                // A String tartalmazza-e a paramétert
        }

        public static boolean haromszogE(int a, int b, int c) {
                return (a + b > c && a + c > b && b + c > a);
        }

}
```

```
1. Egész számok
   ============
Két 1 és 100 közötti véletlen egész szám előállítása:
Az első egész szám: 88, a második pedig: 81
88 a nagyobb.
Összegük: 88 + 81 = 169
Szorzatuk: 88 * 81 = 7128
Egész osztásuk eredménye: 88 / 81 = 1
Maradékos osztásuk eredménye: 88 % 81 = 7
Valós osztásuk eredménye 4 tizedesjegyre kerekítve: 88 / 81 = 1,0864

20! = 2432902008176640000

Figyelni kell a túlcsordulásra!
21! = -4249290049419214848

2. Valós számok
   ============
Két 0 és 100 közötti véletlen valós szám előállítása:
Az első valós szám 3 tizedes pontossággal: u =  50,876, a második pedig: v =  74,293
Összegük: 50,876 + 74,293 = 125,169
Szorzatuk 3 tizedesjeggyel: 50,876 * 74,293 = 3779,734
Különbségük abszolút értéke 3 tizedesjeggyel: |50,876 - 74,293| = 23,418
Hányadosuk 3 tizedesjeggyel: 50,876 / 74,293 = 0,685
Az első szám köbének és a második négyzetgyökének szorzata 3 tizedesjegyre kerekítve: 1135033,894

3. Karakterek és sztringek
   =======================
A * karakter ASCII-kódja: 42

Az ASCII-kódokhoz tartozó karakterek
32:      33: !   34: "   35: #   36: $   37: %   38: &   39: '   40: (   41: )
42: *   43: +   44: ,   45: -   46: .   47: /   48: 0   49: 1   50: 2   51: 3
52: 4   53: 5   54: 6   55: 7   56: 8   57: 9   58: :   59: ;   60: <   61: =
62: >   63: ?   64: @   65: A   66: B   67: C   68: D   69: E   70: F   71: G
72: H   73: I   74: J   75: K   76: L   77: M   78: N   79: O   80: P   81: Q
82: R   83: S   84: T   85: U   86: V   87: W   88: X   89: Y   90: Z   91: [
92: \   93: ]   94: ^   95: _   96: `   97: a   98: b   99: c   100: d  101: e
102: f  103: g  104: h  105: i  106: j  107: k  108: l  109: m  110: n  111: o
112: p  113: q  114: r  115: s  116: t  117: u  118: v  119: w  120: x  121: y
122: z  123: {  124: |  125: }  126: ~  127: □  128: ?  129: ?  130: ?  131: ?
132: ?  133: ?  134: ?  135: ?  136: ?  137: ?  138: ?  139: ?  140: ?  141: ?
142: ?  143: ?  144: ?  145: ?  146: ?  147: ?  148: ?  149: ?  150: ?  151: ?
152: ?  153: ?  154: ?  155: ?  156: ?  157: ?  158: ?  159: ?  160:    161: ?
162: ?  163: ?  164: ¤  165: ?  166: ¦  167: §  168: ¨  169: ©  170: ?  171: «
172: ¬  173: -  174: ®  175: ?  176: °  177: ±  178: ?  179: ?  180: ´  181: µ
182: ¶  183: ·  184: ¸  185: ?  186: ?  187: »  188: ?  189: ?  190: ?  191: ?
192: ?  193: Á  194: Â  195: ?  196: Ä  197: ?  198: ?  199: Ç  200: ?  201: É
202: ?  203: Ë  204: ?  205: Í  206: Î  207: ?  208: ?  209: ?  210: ?  211: Ó
212: Ô  213: ?  214: Ö  215: ×  216: ?  217: ?  218: Ú  219: ?  220: Ü  221: Ý
222: ?  223: ß  224: ?  225: á  226: â  227: ?  228: ä  229: ?  230: ?  231: ç
232: ?  233: é  234: ?  235: ë  236: ?  237: í  238: î  239: ?  240: ?  241: ?
242: ?  243: ó  244: ô  245: ?  246: ö  247: ÷  248: ?  249: ?  250: ú  251: ?
252: ü  253: ý  254: ?  255: ?  256: ?  257: ?  258: Ă  259: ă  260: Ą  261: ą
262: Ć  263: ć  264: ?  265: ?  266: ?  267: ?  268: Č  269: č  270: Ď  271: ď
272: Đ  273: đ  274: ?  275: ?  276: ?  277: ?  278: ?  279: ?  280: Ę  281: ę
282: Ě  283: ě  284: ?  285: ?  286: ?  287: ?  288: ?  289: ?  290: ?  291: ?
292: ?  293: ?  294: ?  295: ?  296: ?  297: ?  298: ?  299: ?  300: ?  301: ?
302: ?  303: ?  304: ?  305: ?  306: ?  307: ?  308: ?  309: ?  310: ?  311: ?
312: ?  313: Ĺ  314: Í  315: ?  316: ?  317: Ľ  318: ľ  319: ?  320: ?  321: Ł
322: ł  323: Ń  324: ń  325: ?  326: ?  327: Ň  328: ň  329: ?  330: ?  331: ?
332: ?  333: ?  334: ?  335: ?  336: Ő  337: ő  338: ?  339: ?  340: Ŕ  341: ŕ
342: ?  343: ?  344: Ř  345: ř  346: Ś  347: ś  348: ?  349: ?  350: Ş  351: ş
352: Š  353: š  354: Ţ  355: ţ  356: Ť  357: ť  358: ?  359: ?  360: ?  361: ?
362: ?  363: ?  364: ?  365: ?  366: Ů  367: ů  368: Ű  369: ű  370: ?
```

```
A Tündérszárnyú Kígyóbűvölő kifejezés karaktereinek ASCII-kódjai:
84 252 110 100 233 114 115 122 225 114 110 121 250 32 75 237 103 121 243 98 369 118 246 108 337

Összefűzés:
***Tündérszárnyú Kígyóbűvölő***

Megfordítás a charAt fügvénnyel:
őlövűbóygíK úynrázsrédnüT

Visszafordítás a substring fügvénnyel:
Tündérszárnyú Kígyóbűvölő

Nagybetűssé alakítás:
TÜNDÉRSZÁRNYÚ KÍGYÓBŰVÖLŐ

Kisbetűssé alakítás:
tündérszárnyú kígyóbűvölő

Darabolás:
A kifejezés 2 szóból áll.
Az 1. szó: Tündérszárnyú
Az 2. szó: Kígyóbűvölő
Összehasonlítjuk a két szót.
Az első szó a szöveg, a második a compareTo paramétere.
Ha az érték 0, akkor egyenlőek, ha pozitív, akkor a szöveg a nagyobb,
ha negatív, akkor a paraméter:
9

Mindkét szóból véletlenszerűen kiválasztunk egy karaktert,
és összehasonlítjuk az ábécében elfoglalt helyzetüket:

Az első szóból kiválasztott karakter: y
A második szóból kiválasztott karakter: ó
A második szóból kiválasztott karakter van előrébb az ábécében.

4. Egydimenziós tömbök
   ==================
Egy megadott pozitív egészeket tartalmazó tömb deklarálása

A tömb elemeinek kiíratása for ciklussal:
12 121 34 2 67 49 30 122 800 1256 2016 3 21 77 4 28 84 1 156 56

1 és 100 közötti véletlen pozitív egészeket tartalmazó tömb előállítása

A tömb elemeinek kiíratása for ciklussal:
40 8 100 83 56 46 63 3 14 62 65 29 23 42 8 34 81 33 61 76 0 0 0 0 0 0 0 0 0 0

A mintaszöveg karaktereinek String típusú elemekből álló tömbje:
T ü n d é r s z á r n y ú   K í g y ó b ű v ö l ő

Kiíratás nagy-, ill. kisbetűs párokkal:
Tt Üü Nn Dd Éé Rr Ss Zz Áá Rr Nn Yy Úú    Kk Íí Gg Yy Óó Bb Űű Vv Öö Ll Őő

A tömb magánhangzóinak kicserélése távirati formára:
T ue n d ee r s z aa r n y uu   K ii g y oo b uue v oe l ooe

5. Kétdimenziós tömbök
   ====================
1000 db 1 és 75 közötti véletlen egész számhármas előállítása,
és tárolása kétdimenziós tömbben
A számhármasok hány százaléka határoz meg háromszöget?

Az első számhármas: 43, 57, 55
Az utolsó számhármas: 44, 9, 35
A háromszöget meghatározók aránya: 50,2%
```

3. Beolvasás, kiírás: a képernyő és a szöveges fájlok kezelése

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Beolvasás, kiírás: a képernyő és a szöveges fájlok kezelése
 *
 * @author Klemand
 */

public class BevGyak03 {
	public static void main(String[] args) throws IOException {
		// A throws IOException csak fájlműveletek esetén kell
		// Importálás is szükséges hozzá: Source menüpont Organize Imports

		System.out.println("1. Egész számok beolvasása a billentyűzetről \n");
		/* Mivel az érettségin feltételezhetjük, hogy helyes számformátumot és értéket
		 * adunk meg, kivételkezeléssel és ellenőrzéssel nem kell foglalkoznunk.
		 */

		Scanner sc1 = new Scanner(System.in);
		// Source menüpont Organize Imports!

		int n, m;

		System.out.print("Kérek egy egész számot a billentyűzetről: ");
		n = sc1.nextInt();
		System.out.println();
		System.out.print("Kérek egy másik egész számot is a billentyűzetről: ");
		m = sc1.nextInt();
		System.out.println("Az első szám: " + n + ", a második szám: " + m);
		System.out.println("A két szám távolsága a számegyenesen: " + Math.abs(n - m));
		double atlag = (double) (n + m) / 2;
		System.out.println("Átlaguk: " + atlag);

		int nagyobb = Math.max(n, m);
		String bin, hex;
		bin = Integer.toBinaryString(nagyobb);
		hex = Integer.toHexString(nagyobb);
		hex = hex.toUpperCase();

		System.out.print("A nagyobb szám: " + nagyobb + ", bináris alakja: " + bin);
		System.out.println(", hexadecimális alakja: " + hex);

		vonal();

		System.out.println("2. Egy hosszú egész szám beolvasása a billentyűzetről \n");

		Scanner sc2 = new Scanner(System.in);

		System.out.println("Addig adogatjuk össze a számjegyeket,");
		System.out.println("míg végül az összeg egyjegyű nem lesz.");

		long szam;

		System.out.print("Kérek egy 12-18 jegyű számot a billentyűzetről: ");

		szam = sc2.nextLong();
```

```
long osszeg;
long maradek;

do {
        osszeg = 0;
        do {
                maradek = szam % 10; // a szám utolsó jegye
                osszeg += maradek; // az utolsó jegy hozzáadása az aktuális összeghez
                szam = szam / 10; // a szám utolsó jegyének elhagyása
        } while (szam > 0);// ha még nem fogyott el a szám, folytatjuk a számjegyek összeadását
        szam = osszeg;
} while (osszeg > 9);// ha a számjegyek összege még többjegyű, megismételjük az algoritmust

System.out.println("A végső összeg: " + osszeg);

vonal();

System.out.println("3. Valós számok beolvasása a billentyűzetről \n");

Scanner sc3 = new Scanner(System.in);

double x, y;

System.out.println("A tizedestörteket tizedesvesszővel kell beírni!\n");
System.out.print("Kérek egy valós számot a billentyűzetről: ");

x = sc3.nextDouble();
System.out.print("Kérek egy másik valós számot is a billentyűzetről: ");
y = sc3.nextDouble();
System.out.println("Az első szám: " + x + ", a második szám: " + y);
System.out.printf("A két szám távolsága a számegyenesen (3 tizedes pontossággal): %.3f\n",
                  Math.abs(x - y));

if (x >= 0 && y >= 0) {
        double mkoz = Math.sqrt(x * y);
        System.out.printf("A két szám mértani közepe (3 tizedes pontossággal): %.3f\n", mkoz);
} else {
        System.out.println("A mértani közepet csak nemnegatív számokra értelmezzük.");
}

vonal();

System.out.println("4. Szöveg beolvasása a billentyűzetről \n");

System.out.println("a) Egy szövegként beolvasott hexadecimális szám értéke és számjegyeinek összege \n");
System.out.println("Kérek egy hexadecimális számot! ");
System.out.print("A számjegyei: 0 1 2 ... 9 A B C D E F lehetnek. (Pl. babfa88): ");

Scanner sc4 = new Scanner(System.in);

hex = sc4.nextLine().toUpperCase();
// a kisbetűvel megadott számjegyeket rögtön nagybetűssé alakítjuk

long hexErtek = Long.parseLong(hex, 16);

System.out.println(hex + " decimális alakja: " + hexErtek);

System.out.print("\nSzámjegyeienek összege: ");

osszeg = 0;
int i;
for (i = 1; i <= hex.length(); i++) {
        osszeg += Integer.parseInt(hex.substring(i - 1, i), 16);
}

System.out.println(osszeg + "\n");
```

```
System.out.println("b) Egy ékezetes szöveg ékezetes karaktereinek összefűzése \n");
System.out.println("Kérek egy ékezetes szöveget! ");

String szoveg = sc4.nextLine();
String ekezetesek = "";
for (i = 1; i <= szoveg.length(); i++) {
        String betu = szoveg.substring(i - 1, i);
        if (ekezetes(betu)) {
                ekezetesek += betu;
        }
}
System.out.println("A beolvasott szöveg ékezetes betűinek füzére: " + ekezetesek + "\n");

System.out.println("c) Egy egészekből álló számsorozat beolvasása tömbbe \n");
System.out.println("Kérem a számsorozat tagjait egy sorban szóközzel elválasztva: ");
String sor = sc4.nextLine();
sor = sor.trim();// esetleges felesleges szóközök levágása a szöveg széleiről
String[] daraboltSor = sor.split(" ");
int db = daraboltSor.length;
int[] X = new int[db];

for (i = 1; i <= db; i++) {
        X[i - 1] = Integer.parseInt(daraboltSor[i - 1]);
}

System.out.println("A beolvasott számtömb elemei:");
for (i = 1; i <= db; i++) {
        System.out.print(X[i - 1] + " ");
}
System.out.println();

sc1.close();//Minden beolvasást le kell zárni!
sc2.close();
sc3.close();
sc4.close();

vonal();

System.out.println("5. Beolvasás szövegfájlból \n");

System.out.println("a) A szövegfájl első sora tartalmazza a beolvasandó sorok számát.");
System.out.println("Ezután minden sorban szóközzel elválasztva egy-egy adathármas szerepel.");
System.out.println("A tizedestörteket a szövegfájlban tizedesponttal kell megadni!");
System.out.println("Olvassuk be egy kétdimenziós tömbbe adathármasokat!");
System.out.println("Ezután írassuk ki a képernyőre, hogy háromszöget határoznak-e meg,");
System.out.println("és ha igen, akkor adjuk meg a kerületüket, a területüket,");
System.out.println("a beírt és a körülírt körük sugarát!\n");

BufferedReader beolvas1 = new BufferedReader(new FileReader("3szog.txt"));
// Source menüpont Organize Imports

String elsosor;
elsosor = beolvas1.readLine();
/*
 * Ha a szövegfájl UTF-8 kódolású, akkor az elején egy UTF-8 azonosító van, ami
 * konvertálásnál hibát okoz.
 *
 * A probléma kezelésére két lehetőségünk van:
 *
 * 1) Külön olvassuk be az első sort, és ha van az elején UTF-8 azonosító,
 * akkor elhagyjuk:
 *
 * if (elsoSor.charAt(0) == (char) 0xFEFF) {
 * elsoSor = elsoSor.substring(1);
 * }
```

```
 * 2) Beolvasás előtt mentsük el a fájlt ANSI kódolással:
 * szerkesztés / mentés másként / felülírás
 *
 *
 * Frissítés: Ez a probléma az Eclipse 2019-06 verzióban már nincs jelen.
 *
 */

int sordb = Integer.parseInt(elsosor.trim());
// Ha a szám mögött egy szóköz is szerepel (volt rá példa), hibát okozhat!


String[] daraboltFsor;
double[][] hszogek = new double[sordb][3];
String fsor;
int j;
for (i = 1; i <= sordb; i++) {
        fsor = beolvas1.readLine();
        daraboltFsor = fsor.split(" ");
        for (j = 1; j <= 3; j++) {
                hszogek[i - 1][j - 1] = Double.parseDouble(daraboltFsor[j - 1]);
        }
}

beolvas1.close();

double a, b, c;

for (i = 1; i <= sordb; i++) {
        a = hszogek[i - 1][0];
        b = hszogek[i - 1][1];
        c = hszogek[i - 1][2];
        if (haromszogE(a,b,c)) {
                System.out.println(a + ", " + b + " és " + c + " háromszöget határoznak meg.");

                double ker = a + b + c;
                double s = ker / 2;
                double ter = Math.sqrt(s * (s - a) * (s - b) * (s - c));
                double rb = ter / s;
                double rk = a * b * c / (4 * ter);
                System.out.printf("Kerülete: %.3f", ker);
                System.out.printf(", területe: %.3f", ter);

                System.out.printf("\nA beírt körének sugara: %.3f", rb);
                System.out.printf(", a körülírt körének sugara: %.3f\n", rk);
                System.out.println();

        } else {
                System.out.print(a + ", " + b + " és " + c + " nem határoznak meg háromszöget.\n");
        }
}

System.out.println("b) A fájl első sora nem tartalmazza a beolvasandó sorok számát.");
System.out.println("Azt viszont tudjuk, hogy pl. max. 100 sort tartalmazhat!");
System.out.println("Minden sorban szóközzel elválasztva egy vezetéknév, egy keresztnév,");
System.out.println("egy személyi szám és a születéskor nyert életbiztosítási összeg szerepel.");
System.out.println("Olvassuk be az adatokat egy tömblistába.");
System.out.println("Ezután írassuk ki a képernyőre a vezetéknevet max. 10 karakterrel,");
System.out.println("majd tabulátorral elválasztva a keresztnév első két karakterét,");
System.out.println("a személy nemét, születési évét, és életbiztosításának 2020 végén");
System.out.println("várható értékét, ha mindegyik év végén 3 %-kal nő!\n");
```

```
BufferedReader beolvas2 = new BufferedReader(new FileReader("szemely.txt"));

db=0;
String[] vezNev = new String[100];
String[] kerNev = new String[100];
String[] szemSzam = new String[100];
double[] biztAlap = new double[100];

/* Sajnos a Java programozási nyelvből hiányzik a rekord adatszerkezet,
 * a tömblista használata viszont nehézkes, egyszerűbb tömbökkel dolgozni.
 * Az i-edik személy adatait, azaz a képzeletbeli i-edik rekord mezőit
 * vezNev[i-1], kerNev[i-1], szemSzam[i-1] és bizt[i-1] tartalmazza.
 * Vigyázni kell, hogy pl. rendezésnél az összetatozó mezőket együtt mozgassuk!
 */


while ((fsor = beolvas2.readLine()) != null) {
        // Beolvassa a sort és megvizsgálja. Ha létezik, azaz nem null, akkor feldolgozza.
        db++;
        daraboltFsor = fsor.split(" ");
        vezNev[db-1]=daraboltFsor[0];
        kerNev[db-1]=daraboltFsor[1];
        szemSzam[db-1]=daraboltFsor[2];
        biztAlap[db-1]=Double.parseDouble(daraboltFsor[3]);
}

beolvas2.close();

System.out.println("A fájl beolvasása megtörtént.");
System.out.println("A fájl " + db + " adatsort tartalmaz.\n");
System.out.println("A kért adatok:");

String aktVezNev;
int hossz;
for (i = 1; i <= db; i++) {

        aktVezNev=vezNev[i-1];
        hossz = aktVezNev.length();
        if(hossz < 10) {
                for(j=1; j<=10-hossz; j++) {
                        aktVezNev = " " + aktVezNev;
                }
        }else {
                aktVezNev=aktVezNev.substring(0, 10);
        }
        System.out.print(aktVezNev);
        System.out.print("\t" + kerNev[i-1].substring(0, 2));
        String nem = nemMeghat(szemSzam[i-1]);
        int szulEv = evMeghat(szemSzam[i-1]);
        System.out.print("\t" + nem + "\t" + szulEv + "\t");
        int ev = 2021 - szulEv;
        double biztErtek = biztAlap[i-1] * Math.pow(1.03, ev);
        System.out.printf("%,14.0f", biztErtek);
        // Ezredes tagolás, szélesség, pontosság beállítása
        System.out.println(" Ft");
}

vonal();

System.out.println("6. Kiíratás szövegfájlba \n");
System.out.println("Írassuk ki az eredmeny.ki szövegfájlba az előző két feladat eredményét!");
System.out.println("Először írjuk felül az esetlegesen meglévő fájlt, a második feladat");
System.out.println("eredménye pedig két üres sor után következzen.\n");
```

```
PrintWriter kiir = new PrintWriter(new FileWriter("eredmeny.ki", false));
/* Source menüpont Organize Imports
 * A fájlkiterjesztés bármi lehet, nem csak txt
 * false: felülírjuk a meglévő fájlt (elhagyható, ez az alapértelmezés)*/

/* Az a) feladatnál nem tároltuk az eredményeket, csak kiírattuk,
 * ezért most mindent újra kell számolni.
 */

kiir.println("Az a) feladat eredménye: ");
kiir.println();
for (i = 1; i <= sordb; i++) {
        a = hszogek[i - 1][0];
        b = hszogek[i - 1][1];
        c = hszogek[i - 1][2];
        if (haromszogE(a,b,c)) {
                kiir.println(a + ", " + b + " és " + c + " háromszöget határoznak meg.");

                double ker = a + b + c;
                double s = ker / 2;
                double ter = Math.sqrt(s * (s - a) * (s - b) * (s - c));
                double rb = ter / s;
                double rk = a * b * c / (4 * ter);
                kiir.printf("Kerülete: %.3f", ker);
                kiir.printf(", területe: %.3f", ter);
                kiir.println();
                // A \n itt nem működik!
                kiir.printf("A beírt körének sugara: %.3f", rb);
                kiir.printf(", a körülírt körének sugara: %.3f", rk);
                kiir.println();

        } else {
                kiir.println(a + ", " + b + " és " + c + " nem határoznak meg háromszöget.");
        }
}

kiir.println();
kiir.println();

kiir.println("A b) feladat eredménye:");
kiir.println();
for (i = 1; i <= db; i++) {
        aktVezNev=vezNev[i-1];
        hossz = aktVezNev.length();
        if(hossz < 10) {
                for(j=1; j<=10-hossz; j++) {
                        aktVezNev = " " + aktVezNev;
                }
        }else {
                aktVezNev=aktVezNev.substring(0, 10);
        }
        kiir.print(aktVezNev);
        kiir.print("\t" + kerNev[i-1].substring(0, 2));
        String nem = nemMeghat(szemSzam[i-1]);
        int szulEv = evMeghat(szemSzam[i-1]);
        kiir.print("\t" + nem + "\t" + szulEv + "\t");
        int ev = 2021 - szulEv;
        double biztErtek = biztAlap[i-1] * Math.pow(1.03, ev);
        kiir.printf("%,14.0f", biztErtek);
        // Ezredes tagolás, szélesség, pontosság beállítása
        kiir.println(" Ft");
}

kiir.close();
// Minden beolvasást és kiíratást le kell zárni.
```

```
                System.out.println("A fájlkiíratás sikeresen befejeződött.");

                vonal();

        }

        public static void vonal() {

        System.out.println("\n********************************************************************************** \n");
        }

        public static boolean ekezetes(String a) {
                String ekezetesABC = "áéíóöőúüűÁÉÍÓÖŐÚÜŰ";
                return ekezetesABC.contains(a);
        }

        public static boolean haromszogE(double a, double b, double c) {
                return (a + b > c && a + c > b && b + c > a);
        }


        public static String nemMeghat(String szsz) {
                if (szsz.substring(0, 1).equals("1") || szsz.substring(0, 1).equals("3")) {
                        // String esetén az == helyett az equals metódust kell alkalmazni!
                        return "férfi";
                } else {
                        return "nő";
                }
        }

        public static int evMeghat(String szsz) {
                int ev = Integer.parseInt(szsz.substring(1, 3));
                if (Integer.parseInt(szsz.substring(0, 1)) <= 2) {
                        return 1900 + ev;
                } else {
                        return 2000 + ev;
                }
        }

}
```

```
1. Egész számok beolvasása a billentyűzetről

Kérek egy egész számot a billentyűzetről: 24

Kérek egy másik egész számot is a billentyűzetről: 76
Az első szám: 24, a második szám: 76
A két szám távolsága a számegyenesen: 52
Átlaguk: 50.0
A nagyobb szám: 76, bináris alakja: 1001100, hexadecimális alakja: 4C

********************************************************************************

2. Egy hosszú egész szám beolvasása a billentyűzetről

Addig adogatjuk össze a számjegyeket,
míg végül az összeg egyjegyű nem lesz.
Kérek egy 12-18 jegyű számot a billentyűzetről: 77777777777777777
A végső összeg: 2

********************************************************************************

3. Valós számok beolvasása a billentyűzetről

A tizedestörteket tizedesvesszővel kell beírni!

Kérek egy valós számot a billentyűzetről: 2,45678
Kérek egy másik valós számot is a billentyűzetről: 8
Az első szám: 2.45678, a második szám: 8.0
A két szám távolsága a számegyenesen (3 tizedes pontossággal): 5,543
A két szám mértani közepe (3 tizedes pontossággal): 4,433

********************************************************************************

4. Szöveg beolvasása a billentyűzetről

a) Egy szövegként beolvasott hexadecimális szám értéke és számjegyeinek összege

Kérek egy hexadecimális számot!
A számjegyei: 0 1 2 ... 9 A B C D E F lehetnek. (Pl. babfa88): babfababa
BABFABABA decimális alakja: 50129975994

Számjegyeienek összege: 99

b) Egy ékezetes szöveg ékezetes karaktereinek összefűzése

Kérek egy ékezetes szöveget!
A hűsítő tó fölé ezüst holdsugár kúszik.
A beolvasott szöveg ékezetes betűinek füzére: űíőóöéüáú

c) Egy egészekből álló számsorozat beolvasása tömbbe

Kérem a számsorozat tagjait egy sorban szóközzel elválasztva:
12 -44 56 -5 0 66
A beolvasott számtömb elemei:
12 -44 56 -5 0 66

********************************************************************************

5. Beolvasás szövegfájlból

a) A szövegfájl első sora tartalmazza a beolvasandó sorok számát.
Ezután minden sorban szóközzel elválasztva egy-egy adathármas szerepel.
A tizedestörteket a szövegfájlban tizedesponttal kell megadni!
Olvassuk be egy kétdimenziós tömbbe adathármasokat!
Ezután írassuk ki a képernyőre, hogy háromszöget határoznak-e meg,
és ha igen, akkor adjuk meg a kerületüket, a területüket,
```

```
a beírt és a körülírt körük sugarát!

5.3, 12.0 és 7.34 háromszöget határoznak meg.
Kerülete: 24,640, területe: 11,740
A beírt körének sugara: 0,953, a körülírt körének sugara: 9,941

4.2, 55.0 és 44.0 nem határoznak meg háromszöget.
3.888, 7.98 és 5.723 háromszöget határoznak meg.
Kerülete: 17,591, területe: 10,400
A beírt körének sugara: 1,182, a körülírt körének sugara: 4,268

2.23, 5.34 és 7.18 háromszöget határoznak meg.
Kerülete: 14,750, területe: 3,880
A beírt körének sugara: 0,526, a körülírt körének sugara: 5,509

b) A fájl első sora nem tartalmazza a beolvasandó sorok számát.
Azt viszont tudjuk, hogy pl. max. 100 sort tartalmazhat!
Minden sorban szóközzel elválasztva egy vezetéknév, egy keresztnév,
egy személyi szám és a születéskor nyert életbiztosítási összeg szerepel.
Olvassuk be az adatokat egy tömblistába.
Ezután írassuk ki a képernyőre a vezetéknevet max. 10 karakterrel,
majd tabulátorral elválasztva a keresztnév első két karakterét,
a személy nemét, születési évét, és életbiztosításának 2020 végén
várható értékét, ha mindegyik év végén 3 %-kal nő!

A fájl beolvasása megtörtént.
A fájl 6 adatsort tartalmaz.

A kért adatok:
Tündérszár      Ír      nő      1980         1 343 960 Ft
Kígyóbűvöl      Ál      férfi   1976         1 134 479 Ft
   Úrhajós      Éz      férfi   2012           652 387 Ft
      Úszó      Őz      nő      2006           311 593 Ft
    Műlábú      Ül      férfi   1956         4 780 988 Ft
    Különc      Úd      férfi   1903         1 635 922 Ft

*******************************************************************************

6. Kiíratás szövegfájlba

Írassuk ki az eredmeny.ki szövegfájlba az előző két feladat eredményét!
Először írjuk felül az esetlegesen meglévő fájlt, a második feladat
eredménye pedig két üres sor után következzen.

A fájlkiíratás sikeresen befejeződött.

*******************************************************************************
```

```
Az eredmeny.ki szövegfájl:

Az a) feladat eredménye:

5.3, 12.0 és 7.34 háromszöget határoznak meg.
Kerülete: 24,640, területe: 11,740
A beírt körének sugara: 0,953, a körülírt körének sugara: 9,941
4.2, 55.0 és 44.0 nem határoznak meg háromszöget.
3.888, 7.98 és 5.723 háromszöget határoznak meg.
Kerülete: 17,591, területe: 10,400
A beírt körének sugara: 1,182, a körülírt körének sugara: 4,268
2.23, 5.34 és 7.18 háromszöget határoznak meg.
Kerülete: 14,750, területe: 3,880
A beírt körének sugara: 0,526, a körülírt körének sugara: 5,509


A b) feladat eredménye:

Tündérszár      Ír      nő      1980         1 343 960 Ft
Kígyóbűvöl      Ál      férfi   1976         1 134 479 Ft
   Űrhajós      Éz      férfi   2012           652 387 Ft
      Úszó      Őz      nő      2006           311 593 Ft
    Műlábú      Ül      férfi   1956         4 780 988 Ft
    Különc      Úd      férfi   1903         1 635 922 Ft
```

4. Elemi algoritmusok: összegzés, eldöntés, keresés, megszámolás, maximum-kiválasztás és kiválogatás

```
import java.util.Scanner;

/**
 * Elemi algoritmusok: összegzés, eldöntés, keresés, megszámolás,
 * maximum-kiválasztás és kiválogatás
 *
 * @author Klemand
 */
public class BevGyak04 {
	public static void main(String[] args) {

		// Az adatok beolvasása a billentyűzetről az X tömbbe
		System.out.println("Kérek egyesével min. 3 és max. 20 db 1 és 100 közötti egész számot! ");
		System.out.println("Befejezés: 0 ");
		//Feltételezhetjük, hogy a felhasználó csak megfelelő számokat ír be.
		int n = 0;
		int[] X = new int[20]; // Az adattömb létrehozása

		Scanner sc = new Scanner(System.in);
		int adat;
		do {
			System.out.print("Kérem az " + (n + 1) + ". számot:  ");
			adat = sc.nextInt();
			if (adat != 0) {
				n++;
				X[n - 1] = adat;
				// Az adattömb feltöltése, az indexelés 0-val kezdődik!
			}
		} while ((n < 3) || (n < 20 && adat != 0));
		//Lépjen vissza, ha nincs meg a 3 adat vagy ha még 20-nál kevesebb van és nem akartuk befejezni.
		sc.close();

		System.out.println("");
		System.out.println("Visszajelzés: ");
		System.out.println("A megadott elemek száma: " + n);
		System.out.println("A megadott elemek: ");
		for (int i = 1; i <= n; i++) {
			System.out.println(i + ".: " + X[i - 1]);
		}
		System.out.println("");

		// 1. Összegzés programozási tétel
		System.out.println("1. Összegzés programozási tétel\n");
		System.out.println("A megadott számok összegét határozza meg.");

		int osszeg = 0;
		for (int i = 1; i <= n; i++) {
			osszeg += X[i - 1];
		}
		System.out.println("");
		System.out.println("A megadott számok összege: " + osszeg);

		rajzol();
```

```
// 2. Eldöntés programozási tétel
System.out.println("2. Eldöntés programozási tétel\n");
System.out.println("Megvizsgálja, hogy a megadott számok között van-e T tulajdonságú elem. ");
System.out.println("Ebben a programban a T tulajdonság azt jelenti,");
System.out.println("hogy az adott elem páros, de nem osztható 4-gyel.");
System.out.println("A vizsgálathoz függvényt használunk.");
System.out.println("Másik tulajdonság esetén csak a függvényt kell átírni.");

int i = 1;
while ((i <= n) && (T(X[i - 1]) == false)) { //vagy while (i<=n && !T(X[i-1]))
        i++;
}
boolean van = (i <= n);

System.out.println("\nA megadott számok között van T tulajdonságú elem: " + van);

rajzol();

// 3. Keresés programozási tétel
System.out.println("3. Keresés programozási tétel\n");
System.out.println("Megvizsgálja, hogy a megadott számok között van-e T tulajdonságú elem. ");
System.out.println("Ha van, akkor megadja az első T tulajdonságú elem sorszámát is. ");
System.out.println("Mindig alkalmazható a Kiválasztás tétel helyett is. ");

int sorszam;
i = 1;
while (i <= n && !T(X[i - 1])) {
        i++;
}
van = (i <= n);
if (van)
{
        sorszam = i;
        System.out.println("Az első T tulajdonságú elem sorszáma: " + sorszam);
        System.out.println("A sorszám ismeretében maga az elem is megadható: " + X[sorszam - 1]);
}else {
        System.out.println("Az elemek között nincs T tulajdonságú.");
}

rajzol();

// 4. Megszámolás programozási tétel
System.out.println("4. Megszámolás programozási tétel\n");
System.out.println("Meghatározza, hogy a megadott számok között hány T tulajdonságú elem van. ");

int db = 0;
for (i = 1; i <= n; i++) {
        if (T(X[i - 1])) // if (T(X[i-1])==true)
        {
                db++;
        }
}

System.out.println("");
System.out.println("A megadott számok között " + db + " darab T tulajdonságú elem van. ");

rajzol();
```

```
		// 5. Maximum-kiválasztás programozási tétel
		System.out.println("5. Maximum-kiválasztás programozási tétel\n");
		System.out.println("5.a) Meghatározza a maximális elemek közül a legelső sorszámát. ");

		int max = 1;
		//Abból indulunk ki, hogy az első elem a legnagyobb, amíg nem találunk nála nagyobbat.
		for (i = 2; i <= n; i++) {
			if (X[i - 1] > X[max - 1]) {
				max = i;
			}
		}

		System.out.println("");
		System.out.println("A maximális elem(ek közül az első) sorszáma: " + max);
		System.out.println("A sorszám ismeretében maga az elem is megadható: " + X[max - 1]);
		System.out.println("");

		System.out.println("5.b) Meghatározza a T tulajdonsággal rendelkező (első) maximális elem sorszámát. ");

		int Tmax = 0;
		for (i = 1; i <= n; i++) {
			if (T(X[i - 1])) {
				if (Tmax == 0) {
					Tmax = i;
				} else {
					if (X[i - 1] > X[Tmax - 1]) {
						Tmax = i;
					}
				}
			}
		}
		System.out.println("");
		if (Tmax > 0) {
			System.out.println("A T tulajdonságú elemek közül a(z első) legnagyobb elem sorszáma: " + Tmax);
			System.out.println("A sorszám ismeretében maga az elem is megadható: " + X[Tmax - 1]);
		} else {
			System.out.println("Nincs az adatok között T tulajdonságú elem.");
		}

		rajzol();

		// 6. Kiválogatás programozási tétel
		System.out.println("6. Kiválogatás programozási tétel\n");
		System.out.println("A T tulajdonságú elemek sorszámait kiválogatjuk egy új tömbbe. ");
		// Egyszerűbb csak a sorszámokat tárolni, mint az elemeket!

		int[] TXS = new int[20]; // A sorszámtömb létrehozása
		db = 0;
		for (i = 1; i <= n; i++) {
			if (T(X[i - 1]))
			{
				db++;
				TXS[db - 1] = i;
			}
		}
		System.out.println("");
		for (i = 1; i <= db; i++) // Ha db=0, egyszer sem hajtódik végre!
		{
			System.out.println("Az " + i + ". T tulajdonságú elem sorszáma: " + TXS[i - 1] + ", maga az elem:  "
					+ X[TXS[i - 1] - 1]);
		}

		rajzol();

	}
```

```
        // Eljárások, függvények

        public static void rajzol() {
                // Eljárás: void típusú
                System.out.println("\n * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *");
                System.out.println("* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *\n");
        }

        // A T tulajdonság megadása függvénnyel
        public static boolean T(int n) {
                return (n % 2 == 0 && n % 4 != 0);
        }

}
```

```
Kérek egyesével min. 3 és max. 20 db 1 és 100 közötti egész számot!
Befejezés: 0
Kérem az 1. számot:  0
Kérem az 1. számot:  3
Kérem az 2. számot:  24
Kérem az 3. számot:  6
Kérem az 4. számot:  66
Kérem az 5. számot:  84
Kérem az 6. számot:  76
Kérem az 7. számot:  99
Kérem az 8. számot:  1
Kérem az 9. számot:  77
Kérem az 10. számot:  42
Kérem az 11. számot:  0

Visszajelzés:
A megadott elemek száma: 10
A megadott elemek:
1.: 3
2.: 24
3.: 6
4.: 66
5.: 84
6.: 76
7.: 99
8.: 1
9.: 77
10.: 42

1. Összegzés programozási tétel

A megadott számok összegét határozza meg.

A megadott számok összege: 478

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

2. Eldöntés programozási tétel

Megvizsgálja, hogy a megadott számok között van-e T tulajdonságú elem.
Ebben a programban a T tulajdonság azt jelenti,
hogy az adott elem páros, de nem osztható 4-gyel.
A vizsgálathoz függvényt használunk.
Másik tulajdonság esetén csak a függvényt kell átírni.

A megadott számok között van T tulajdonságú elem: true

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

3. Keresés programozási tétel

Megvizsgálja, hogy a megadott számok között van-e T tulajdonságú elem.
Ha van, akkor megadja az első T tulajdonságú elem sorszámát is.
Mindig alkalmazható a Kiválasztás tétel helyett is.
Az első T tulajdonságú elem sorszáma: 3
A sorszám ismeretében maga az elem is megadható: 6

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

```
4. Megszámolás programozási tétel

Meghatározza, hogy a megadott számok között hány T tulajdonságú elem van.

A megadott számok között 3 darab T tulajdonságú elem van.

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

5. Maximum-kiválasztás programozási tétel

5.a) Meghatározza a maximális elemek közül a legelső sorszámát.

A maximális elem(ek közül az első) sorszáma: 7
A sorszám ismeretében maga az elem is megadható: 99

5.b) Meghatározza a T tulajdonsággal rendelkező (első) maximális elem sorszámát.

A T tulajdonságú elemek közül a(z első) legnagyobb elem sorszáma: 4
A sorszám ismeretében maga az elem is megadható: 66

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

6. Kiválogatás programozási tétel

A T tulajdonságú elemek sorszámait kiválogatjuk egy új tömbbe.

Az 1. T tulajdonságú elem sorszáma: 3, maga az elem:  6
Az 2. T tulajdonságú elem sorszáma: 4, maga az elem:  66
Az 3. T tulajdonságú elem sorszáma: 10, maga az elem:  42

 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
```

5. Rendezések: Egyszerű cserés-, minimum-kiválasztásos-, buborékos- és beillesztéses rendezés

```
**
 * Rendezések: Egyszerű cserés-, minimum-kiválasztásos-, buborékos-
 * és beillesztéses rendezés
 *
 * @author Klemand
 */

public class BevGyak05 {
        public static void main(String[] args) {

                System.out.println("Megadunk egy 20 elemű tömböt,");
                System.out.println("majd különböző algoritmusokkal nagyság szerint rendezzük az elemeket.");

                // Az alaptömb megadása

                int[] alaptomb = { 12, 121, 34, 2, 67, 49, 30, 122, 800, 1256, 2016, 3, 21, 77, 4, 28, 84, 1, 156, 56 };
                int m = alaptomb.length;
                System.out.println("");

                int[] X = alaptomb;

                // 1. Egyszerű cserés rendezés
                System.out.println("1. Egyszerű cserés rendezés");
                System.out.println("<<<<<<<<<<<<<<<<<<<<<<<<<<\n");

                System.out.println("Az első még nem rendezett elemet hasonlítjuk össze a mögötte levőkkel,");
                System.out.println("és ha kisebbet találunk nála, megcseréljük őket.");
                System.out.println("Hátránya a sok mozgatás és a távoli elemek cseréje.");
                int i,j,asztal;

                for (i = 1; i <= m; i++) {
                        for (j = i + 1; j <= m; j++) {
                                if (X[j - 1] < X[i - 1]) {
                                        asztal = X[i - 1];// elemcsere
                                        X[i - 1] = X[j - 1];
                                        X[j - 1] = asztal;
                                }
                        }
                }

                System.out.println("");
                System.out.println("A nagyság szerint rendezett elemek: ");
                for (i = 1; i <= m; i++) {
                        System.out.print(X[i - 1] + " ");
                }
                System.out.println("");

                rajzol();

                // 2. Minimum-kiválasztásos rendezés
                System.out.println("2. Minimum-kiválasztásos rendezés");
                System.out.println("<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n");

                System.out.println("Csak minden ciklus végén cseréljük ki az aktuális első elemet a mögötte levő");
                System.out.println("legkisebbel, melynek az indexét követjük nyomon.");
                System.out.println("Kevesebb mozgatás történik, de a távoli elemek cseréje miatt");
                System.out.println("alkalmatlan a több szempont szerinti rendezésekre.");

                X = alaptomb;
```

```
for (i = 1; i <= m; i++) {
        int min = i;
        for (j = i + 1; j <= m; j++) {
                if (X[j - 1] < X[min - 1]) {
                        min = j;
                }
        }
        asztal = X[i - 1];// elemcsere
        X[i - 1] = X[min - 1];
        X[min - 1] = asztal;
}

System.out.println("");
System.out.println("A nagyság szerint rendezett elemek: ");
for (i = 1; i <= m; i++) {
        System.out.print(X[i - 1] + " ");
}
System.out.println("");

rajzol();

// 3. Buborékos rendezés
System.out.println("3. Buborékos rendezés");
System.out.println("<<<<<<<<<<<<<<<<<<<<\n");

System.out.println("Először az első elemtől az n-1-edikig eljutva minden elemet összehasonlítunk");
System.out.println("a mögötte levővel, és ha a sorrendjük nem jó, akkor megcseréljük őket. ");
System.out.println("Ezáltal a kisebb elemek a sorozat eleje, a nagyobbak pedig a vége felé");
System.out.println("mozdulnak el, és a legnagyobb biztosan a végére kerül. ");
System.out.println("A következő ciklusban már csak az n-2-ik elemig kell eljutnunk, ");
System.out.println("végül pedig csak az elsőt kell összehasonlítanunk a másodikkal.\n");
System.out.println("Többszöri végrehajtással alkalmas több szempont szerinti rendezésre is.");

X = alaptomb;

for (i = m - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (X[j - 1] > X[j]) {
                        asztal = X[j - 1];// elemcsere
                        X[j - 1] = X[j];
                        X[j] = asztal;
                }
        }
}

System.out.println("");
System.out.println("A nagyság szerint rendezett elemek: ");
for (i = 1; i <= m; i++) {
        System.out.print(X[i - 1] + " ");
}
System.out.println("");

rajzol();

// 4. Beillesztéses rendezés
System.out.println("4. Beillesztéses rendezés");
System.out.println("<<<<<<<<<<<<<<<<<<<<<<<<\n");

System.out.println("A módszer lényege: ");
System.out.println("- Egyetlen elem mindig rendezett ");
System.out.print("- Ha van egy rendezett részsorozatunk, ");
System.out.println("abba mindig be tudunk illeszteni egy új elemet a megfelelő helyre.\n");
System.out.println("Ezt a módszert alkalmazzuk pl. bizonyítványok névsorba rendezéséhez.");

X = alaptomb;
```

```
        for (i = 2; i <= m; i++) {
            j = i - 1;
            while (j >= 1 && X[j - 1] > X[j]) {
                asztal = X[j - 1];// elemcsere
                X[j - 1] = X[j];
                X[j] = asztal;
                j--;
            }
        }

        System.out.println("");
        System.out.println("A nagyság szerint rendezett elemek: ");
        for (i = 1; i <= m; i++) {
            System.out.print(X[i - 1] + " ");
        }
        System.out.println("");

        rajzol();

    }

    public static void rajzol() {

    System.out.println("\n<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n");
    }

}
```

```
Megadunk egy 20 elemű tömböt,
majd különböző algoritmusokkal nagyság szerint rendezzük az elemeket.

1. Egyszerű cserés rendezés
<<<<<<<<<<<<<<<<<<<<<<<<<<<

Az első még nem rendezett elemet hasonlítjuk össze a mögötte levőkkel,
és ha kisebbet találunk nála, megcseréljük őket.
Hátránya a sok mozgatás és a távoli elemek cseréje.

A nagyság szerint rendezett elemek:
1 2 3 4 12 21 28 30 34 49 56 67 77 84 121 122 156 800 1256 2016

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

2. Minimum-kiválasztásos rendezés
<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Csak minden ciklus végén cseréljük ki az aktuális első elemet a mögötte levő
legkisebbel, melynek az indexét követjük nyomon.
Kevesebb mozgatás történik, de a távoli elemek cseréje miatt
alkalmatlan a több szempont szerinti rendezésekre.

A nagyság szerint rendezett elemek:
1 2 3 4 12 21 28 30 34 49 56 67 77 84 121 122 156 800 1256 2016

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

3. Buborékos rendezés
<<<<<<<<<<<<<<<<<<<<<

Először az első elemtől az n-1-edikig eljutva minden elemet összehasonlítunk
a mögötte levővel, és ha a sorrendjük nem jó, akkor megcseréljük őket.
Ezáltal a kisebb elemek a sorozat eleje, a nagyobbak pedig a vége felé
mozdulnak el, és a legnagyobb biztosan a végére kerül.
A következő ciklusban már csak az n-2-ik elemig kell eljutnunk,
végül pedig csak az elsőt kell összehasonlítanunk a másodikkal.

Többszöri végrehajtással alkalmas több szempont szerinti rendezésre is.

A nagyság szerint rendezett elemek:
1 2 3 4 12 21 28 30 34 49 56 67 77 84 121 122 156 800 1256 2016

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

4. Beillesztéses rendezés
<<<<<<<<<<<<<<<<<<<<<<<<<

A módszer lényege:
- Egyetlen elem mindig rendezett
- Ha van egy rendezett részsorozatunk, abba mindig be tudunk illeszteni egy új elemet a megfelelő helyre.

Ezt a módszert alkalmazzuk pl. bizonyítványok névsorba rendezéséhez.

A nagyság szerint rendezett elemek:
1 2 3 4 12 21 28 30 34 49 56 67 77 84 121 122 156 800 1256 2016

<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
```

6. Rendezés két szempont szerint: két- ill. egy lépésben

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Két szempont szerinti rendezés
 *
 * @author Klemand
 */

public class BevGyak06 {
        public static void main(String[] args) throws IOException {

                System.out.println("Egy iskolai csapat tagjainak vezetéknevét, első keresztnevét,");
                System.out.println("születési évét és lakhelyét tároljuk a csapat.txt fájl");
                System.out.println("soraiban szóközökkel elválasztva (max. 20 fő).\n");
                System.out.println("A listát születési év és azon belül név szerint akarjuk rendezni.");

                System.out.println("Először olvassuk be az adatokat,");
                System.out.println("majd rendezzük a tömböket először a mellékszempont (név),");
                System.out.println("majd második lépésben a főszempont (születési év) szerint.");
                System.out.println("Az eredményt írassuk ki az eredetihez hasonló formában");
                System.out.println("a képernyőre és a csapat2l.txt fájlba is!\n");

                System.out.println("Beolvasás");

                BufferedReader beolvas = new BufferedReader(new FileReader("csapat.txt"));
                // Source menüpont Organize Imports
                String[] nev = new String[20];
                int[] szev = new int[20];
                String[] lakhely = new String[20];

                String fsor;
                String[] daraboltFsor;
                int db = 0;
                while ((fsor = beolvas.readLine()) != null) {
                        // Beolvassa a sort és megvizsgálja. Ha nem null, akkor feldolgozza.
                        daraboltFsor = fsor.split(" ");
                        db++;
                        nev[db - 1] = daraboltFsor[0] + " " + daraboltFsor[1];
                        szev[db - 1] = Integer.parseInt(daraboltFsor[2]);
                        lakhely[db - 1] = daraboltFsor[3];
                }

                beolvas.close();
                System.out.println("A fájl beolvasása megtörtént.");
                System.out.println("A csapat " + db + " tagból áll.\n");

                int i, j;
                for (i = 1; i <= db; i++) {
                        System.out.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
                }
```

```
System.out.println("\nElőször a mellékszempont szerint bármelyik rendezést választhatjuk.");
System.out.println("Alkalmazzuk az egyszerű cserés rendezést!\n");
/*
 * Ha az ékezetes betűket a magyar ábécé szerint kívánjuk rendezni, akkor külön
 * függvényt kell írnunk az összehasonlításra (lásd BevGyak12), viszont az
 * érettségin nem kell ékezetes betűket kezelni, ezért most a beépített
 * összehasonlító függvényt alkalmazzuk.
 */

String StringAsztal;
int intAsztal;

for (i = 1; i <= db; i++) {
        for (j = i + 1; j <= db; j++) {
                if (nev[j - 1].compareTo(nev[i - 1]) < 0) {
                        // A paraméter a nagyobb
                        StringAsztal = nev[i - 1];
                        nev[i - 1] = nev[j - 1];
                        nev[j - 1] = StringAsztal;
                        // Nem elég a neveket cserélni, a hozzá tartozó adatokat is kell!

                        intAsztal = szev[i - 1];
                        szev[i - 1] = szev[j - 1];
                        szev[j - 1] = intAsztal;

                        StringAsztal = lakhely[i - 1];
                        lakhely[i - 1] = lakhely[j - 1];
                        lakhely[j - 1] = StringAsztal;
                }
        }
}

System.out.println("A csapatlista a mellékszempont (név) szerinti rendezés után:\n");

for (i = 1; i <= db; i++) {
        System.out.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
}

System.out.println("\nA  főszempont szerinti rendezésnél már csak olyan");
System.out.println("algoritmust választhatunk, amelyik nem rontja el a korábbit.");
System.out.println("Nem lehetnek benne távoli cserék!");
System.out.println("Alkalmazzuk a buborékos rendezést!\n");

for (i = db - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (szev[j - 1] > szev[j]) {

                        intAsztal = szev[j - 1];
                        szev[j - 1] = szev[j];
                        szev[j] = intAsztal;

                        StringAsztal = nev[j - 1];
                        nev[j - 1] = nev[j];
                        nev[j] = StringAsztal;

                        StringAsztal = lakhely[j - 1];
                        lakhely[j - 1] = lakhely[j];
                        lakhely[j] = StringAsztal;
                }
        }
}
```

```
		System.out.println("A csapatlista a főszempont (születési év) szerinti rendezés után:\n");

		for (i = 1; i <= db; i++) {
			System.out.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
		}

		PrintWriter kiir = new PrintWriter(new FileWriter("csapat2l.txt"));
		// Source menüpont Organize Imports

		for (i = 1; i <= db; i++) {
			kiir.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
		}

		kiir.close();

	System.out.println("\n*********************************************************************************\n");

		System.out.println("Ezután olvassuk be újra az adatokat a csapat.txt fájlból,");
		System.out.println("majd rendezzük a csapatlistát egyetlen lépésben ");
		System.out.println("születési év és azon belül név szerint is.");
		System.out.println("Az eredményt írassuk ki az eredetihez hasonló formában");
		System.out.println("a képernyőre és a csapat1l.txt fájlba is!\n");

		System.out.println("Beolvasás");

		beolvas = new BufferedReader(new FileReader("csapat.txt"));
		db = 0;
		while ((fsor = beolvas.readLine()) != null) {
			daraboltFsor = fsor.split(" ");
			db++;
			nev[db - 1] = daraboltFsor[0] + " " + daraboltFsor[1];
			szev[db - 1] = Integer.parseInt(daraboltFsor[2]);
			lakhely[db - 1] = daraboltFsor[3];
		}

		beolvas.close();

		System.out.println("A fájl beolvasása megtörtént.");

		System.out.println("A csapat " + db + " tagból áll.\n");

		for (i = 1; i <= db; i++) {
			System.out.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
		}

		System.out.println("\nEgyszerű cserés rendezés egy fő- és mellékszempont szerint\n");

		for (i = 1; i <= db - 1; i++) {
			for (j = i + 1; j <= db; j++) {
				if ((szev[j - 1] < szev[i - 1])
					|| (szev[j - 1] == szev[i - 1]) && (nev[j - 1].compareTo(nev[i - 1]) < 0)) {

					StringAsztal = nev[i - 1];
					nev[i - 1] = nev[j - 1];
					nev[j - 1] = StringAsztal;

					intAsztal = szev[i - 1];
					szev[i - 1] = szev[j - 1];
					szev[j - 1] = intAsztal;

					StringAsztal = lakhely[i - 1];
					lakhely[i - 1] = lakhely[j - 1];
					lakhely[j - 1] = StringAsztal;
				}
			}
		}
```

```
                System.out.println("A csapatlista a rendezés után:\n");
                for (i = 1; i <= db; i++) {
                        System.out.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
                }

                kiir = new PrintWriter(new FileWriter("csapat1l.txt"));

                for (i = 1; i <= db; i++) {
                        kiir.println(nev[i - 1] + " " + szev[i - 1] + " " + lakhely[i - 1]);
                }

                kiir.close();

        }

}
```

```
Egy iskolai csapat tagjainak vezetéknevét, első keresztnevét,
születési évét és lakhelyét tároljuk a csapat.txt fájl
soraiban szóközökkel elválasztva (max. 20 fő).

A listát születési év és azon belül név szerint akarjuk rendezni.
Először olvassuk be az adatokat,
majd rendezzük a tömböket először a mellékszempont (név),
majd második lépésben a főszempont (születési év) szerint.
Az eredményt írassuk ki az eredetihez hasonló formában
a képernyőre és a csapat2l.txt fájlba is!

Beolvasás
A fájl beolvasása megtörtént.
A csapat 16 tagból áll.

Kovács Aladár 1999 Őriszentpéter
Illés Imre 1998 Körmend
Vas Eszter 2002 Szentgotthárd
Zala Eszter 2001 Körmend
Alföldi Gizella 1999 Egyházasrádóc
Badacsonyi Béla 2000 Körmend
Arany Írisz 2000 Szombathely
Hentes Oszkár 1998 Vasvár
Fürge Ferenc 1999 Vasvár
Hajós Annamária 2001 Csákánydoroszló
Bíró Judit 1998 Körmend
Deák Dóra 1999 Körmend
Nagy János 1998 Körmend
Mázsa Lenke 2001 Ják
Mázsa Nóra 2001 Ják
Zala Ibolya 2001 Szalafő

Először a mellékszempont szerint bármelyik rendezést választhatjuk.
Alkalmazzuk az egyszerű cserés rendezést!

A csapatlista a mellékszempont (név) szerinti rendezés után:

Alföldi Gizella 1999 Egyházasrádóc
Arany Írisz 2000 Szombathely
Badacsonyi Béla 2000 Körmend
Bíró Judit 1998 Körmend
Deák Dóra 1999 Körmend
Fürge Ferenc 1999 Vasvár
Hajós Annamária 2001 Csákánydoroszló
Hentes Oszkár 1998 Vasvár
Illés Imre 1998 Körmend
Kovács Aladár 1999 Őriszentpéter
Mázsa Lenke 2001 Ják
Mázsa Nóra 2001 Ják
Nagy János 1998 Körmend
Vas Eszter 2002 Szentgotthárd
Zala Eszter 2001 Körmend
Zala Ibolya 2001 Szalafő

A  főszempont szerinti rendezésnél már csak olyan
algoritmust választhatunk, amelyik nem rontja el a korábbit.
Nem lehetnek benne távoli cserék!
Alkalmazzuk a buborékos rendezést!

A csapatlista a főszempont (születési év) szerinti rendezés után:

Bíró Judit 1998 Körmend
Hentes Oszkár 1998 Vasvár
Illés Imre 1998 Körmend
Nagy János 1998 Körmend
```

```
Alföldi Gizella 1999 Egyházasrádóc
Deák Dóra 1999 Körmend
Fürge Ferenc 1999 Vasvár
Kovács Aladár 1999 Őriszentpéter
Arany Írisz 2000 Szombathely
Badacsonyi Béla 2000 Körmend
Hajós Annamária 2001 Csákánydoroszló
Mázsa Lenke 2001 Ják
Mázsa Nóra 2001 Ják
Zala Eszter 2001 Körmend
Zala Ibolya 2001 Szalafő
Vas Eszter 2002 Szentgotthárd

********************************************************************************

Ezután olvassuk be újra az adatokat a csapat.txt fájlból,
majd rendezzük a csapatlistát egyetlen lépésben
születési év és azon belül név szerint is.
Az eredményt írassuk ki az eredetihez hasonló formában
a képernyőre és a csapat1l.txt fájlba is!

Beolvasás
A fájl beolvasása megtörtént.
A csapat 16 tagból áll.

Kovács Aladár 1999 Őriszentpéter
Illés Imre 1998 Körmend
Vas Eszter 2002 Szentgotthárd
Zala Eszter 2001 Körmend
Alföldi Gizella 1999 Egyházasrádóc
Badacsonyi Béla 2000 Körmend
Arany Írisz 2000 Szombathely
Hentes Oszkár 1998 Vasvár
Fürge Ferenc 1999 Vasvár
Hajós Annamária 2001 Csákánydoroszló
Bíró Judit 1998 Körmend
Deák Dóra 1999 Körmend
Nagy János 1998 Körmend
Mázsa Lenke 2001 Ják
Mázsa Nóra 2001 Ják
Zala Ibolya 2001 Szalafő

Egyszerű cserés rendezés egy fő- és mellékszempont szerint

A csapatlista a rendezés után:

Bíró Judit 1998 Körmend
Hentes Oszkár 1998 Vasvár
Illés Imre 1998 Körmend
Nagy János 1998 Körmend
Alföldi Gizella 1999 Egyházasrádóc
Deák Dóra 1999 Körmend
Fürge Ferenc 1999 Vasvár
Kovács Aladár 1999 Őriszentpéter
Arany Írisz 2000 Szombathely
Badacsonyi Béla 2000 Körmend
Hajós Annamária 2001 Csákánydoroszló
Mázsa Lenke 2001 Ják
Mázsa Nóra 2001 Ják
Zala Eszter 2001 Körmend
Zala Ibolya 2001 Szalafő
Vas Eszter 2002 Szentgotthárd
```

7. A maximumkiválasztás általánosítása: versenykiértékelések

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

        /**
         * A maximumkiválasztás általánosítása: versenykiértékelések
         *
         * @author Klemand
         */

public class BevGyak07 {

        public static void main(String[] args) throws IOException {

                System.out.println("A verseny.txt szövegfájl sorai szóközzel elválasztva egy verseny");
                System.out.println("max. 100 indulójának azonosítóit és az elért pontszámait tartalmazzák.");
                System.out.println("Az adatok beolvasása");

                String[] azon = new String[100];
                int[] pont = new int[100];

                BufferedReader beolvas = new BufferedReader(new FileReader("verseny.txt"));

                String sor;
                String[] daraboltSor;
                int db = 0;

                while ((sor = beolvas.readLine()) != null) {
                        db++;
                        daraboltSor = sor.split(" ");
                        azon[db - 1] = daraboltSor[0];
                        pont[db - 1] = Integer.parseInt(daraboltSor[1]);
                }

                beolvas.close();
```

```
System.out.println("\n1. A k darab legmagasabb pontszám meghatározása");
System.out.println("Azt is vizsgálni kell, hogy van-e k db különböző pontszám!");
System.out.println("A helyezettek kiíratása a pontszámok alapján az eredmeny.txt szövegfájlba");

int k;
Scanner sc = new Scanner(System.in);

System.out.print("Kérek egy 1 és " + db + " közötti egész számot a billentyűzetről: ");
k = sc.nextInt();

int[] maxpontok = new int[k];
int minpont = pont[0];

int i;
for (i = 2; i <= db; i++) {
        if (pont[i - 1] < minpont) {
                minpont = pont[i - 1];
        }
}

int hatar = 1000; // irreálisan magas pontszám
int maxpont = 0; // irreális kezdőérték

i = 1;
int j;
while (i <= k && hatar > minpont) {
        maxpont = 0; // irreális kezdőérték
        for (j = 1; j <= db; j++) {
                if (pont[j - 1] < hatar) {
                        if (pont[j - 1] > maxpont) {
                                maxpont = pont[j - 1];
                        }
                }
        }
        maxpontok[i - 1] = maxpont;
        hatar = maxpont;
        i++;
}

boolean van = (i > k);
int k1 = i - 1; // a különböző értékek tényleges száma

if (van) {
        System.out.println("A(z) " + k + ". legnagyobb pontszám: " + maxpontok[k - 1]);
        System.out.println();
} else {
        System.out.println("Nincs " + k + " darab különböző pontszám.\n");
}
PrintWriter kiir = new PrintWriter(new FileWriter("eredmeny.txt"));

for (i = 1; i <= k1; i++) {
        kiir.println(i + ". helyezett(ek): ");
        for (j = 1; j <= db; j++) {
                if (pont[j - 1] == maxpontok[i - 1]) {
                        kiir.println(azon[j - 1] + " " + pont[j - 1] + " pont");
                }
        }
}
kiir.println();
```

```
                System.out.println("\n2. Az első k legjobb eredményt elérő versenyző meghatározása");
                System.out.println("Holtverseny esetén további versenyzők is szerepelnek!");
                System.out.println("Ilyenkor k értéke 1 és " + db + " között bármennyi lehet.");
                System.out.println("A helyezettek kiíratása a tényleges sorrend alapján az eredmeny.txt szövegfájlba");

                System.out.print("Kérek egy 1 és " + db + " közötti egész számot a billentyűzetről: ");
                k = sc.nextInt();
                sc.close();

                hatar = 1000; // irreálisan magas pontszám
                int hely = 0;
                int helyDb = 0;
                while (helyDb < k) {
                        maxpont = 0; // irreális kezdőérték
                        hely = helyDb + 1;

                        for (j = 1; j <= db; j++) {
                                if (pont[j - 1] < hatar) {
                                        if (pont[j - 1] > maxpont) {
                                                maxpont = pont[j - 1];
                                        }
                                }
                        }
                        for (j = 1; j <= db; j++) {
                                if (pont[j - 1] == maxpont) {
                                        helyDb++;
                                        kiir.println(hely + ". helyezett: " + azon[j - 1] + " " + pont[j - 1] + " pont");
                                }
                        }
                        hatar = maxpont;
                }

                System.out.println("A fájlkiírás megtörtént.");
                kiir.close();
        }

}
```

```
A verseny.txt szövegfájl sorai szóközzel elválasztva egy verseny
max. 100 indulójának azonosítóit és az elért pontszámait tartalmazzák.
Az adatok beolvasása

1. A k darab legmagasabb pontszám meghatározása
Azt is vizsgálni kell, hogy van-e k db különböző pontszám!
A helyezettek kiíratása a pontszámok alapján az eredmeny.txt szövegfájlba
Kérek egy 1 és 16 közötti egész számot a billentyűzetről: 8
Nincs 8 darab különböző pontszám.


2. Az első k legjobb eredményt elérő versenyző meghatározása
Holtverseny esetén további versenyzők is szerepelnek!
Ilyenkor k értéke 1 és 16 között bármennyi lehet.
A helyezettek kiíratása a tényleges sorrend alapján az eredmeny.txt szövegfájlba
Kérek egy 1 és 16 közötti egész számot a billentyűzetről: 8
A fájlkiírás megtörtént.


Az eredmeny.txt szövegfájl:

1. helyezett(ek):
123456 80 pont
zold 80 pont
2. helyezett(ek):
FFF 77 pont
fizikus 77 pont
3. helyezett(ek):
3puputeve 72 pont
karoly 72 pont
c786 72 pont
4. helyezett(ek):
zx428 67 pont
mz/x 67 pont
oroszlan 67 pont
5. helyezett(ek):
b66 66 pont
kishableany 66 pont
6. helyezett(ek):
matrix77 54 pont
aladar 54 pont
asdf 54 pont
iqbetyar 54 pont

1. helyezett: 123456 80 pont
1. helyezett: zold 80 pont
3. helyezett: FFF 77 pont
3. helyezett: fizikus 77 pont
5. helyezett: 3puputeve 72 pont
5. helyezett: karoly 72 pont
5. helyezett: c786 72 pont
8. helyezett: zx428 67 pont
8. helyezett: mz/x 67 pont
8. helyezett: oroszlan 67 pont
```

8. A leghosszabb adott tulajdonságú részsorozat keresése: a leghosszabb prímsorozat egy véletlen sorozatban

```
public class BevGyak07 {

        /**
         * A leghosszabb adott tulajdonságú részsorozat keresése: prímsorozat
         * véletlen sorozatban
         *
         * @author Klemand
         */

        public static void main(String[] args) {

                System.out.println("500 db 1 és 100 közötti véletlen egész szám előállítása\n");

                int m = 500;
                int n = 100;
                int[] X = new int[m];
                int i;

                for (i = 1; i <= m; i++) {
                        X[i - 1] = (int) (Math.random() * n) + 1;
                }

                for (i = 1; i <= m; i++) {
                        System.out.printf("%3d", X[i - 1]);
                        if (T(X[i - 1])) {
                                System.out.print("*\t");
                        } else {
                                System.out.print(" \t");
                        }
                        if (i % 10 == 0) {
                                System.out.println();
                        }
                        if (i % 100 == 0) {
                                System.out.println();
                        }
                }
                System.out.println();

                int aktTkezdet = -1; // irreális kezdőérték
                int aktThossz = 0;
                int maxTkezdet = -1; // irreális kezdőérték
                int maxThossz = 0;

                for (i = 1; i <= m; i++) {
                        if (T(X[i - 1])) {
                                if (aktTkezdet == -1) {
                                        aktTkezdet = i;
                                }
                                aktThossz++;
                                if (aktThossz > maxThossz) {
                                        maxTkezdet = aktTkezdet;
                                        maxThossz = aktThossz;
                                }
                        } else {
                                aktTkezdet = -1;
                                aktThossz = 0;
                        }
                }
```

```
                if (maxTkezdet > -1) {
                        System.out.println("A leghosszabb prímsorozat kezdete: " + maxTkezdet + ". elem: "
                                        + X[maxTkezdet - 1]);
                        System.out.println("Hossza: " + maxThossz);
                        System.out.print("Az elemek: ");
                        for (i = maxTkezdet; i <= maxTkezdet + maxThossz - 1; i++) {
                                System.out.print(X[i - 1] + "\t");
                                if ((i - maxTkezdet + 1) % 10 == 0) {
                                        System.out.println();
                                }
                        }
                } else {
                        System.out.println("A sorozatban nincs prímszám.");
                }
                System.out.println();

        }

        // Eljárások, függvények

        public static boolean T(int p) { // T: prím-e
                if(p==1 || (p>2 && p%2==0) || (p>3 && p%3==0) || (p>5 && p%5==0) || (p>7 && p%7==0)) {
                        //p max. 100 lehet, így a prímosztókat elég 10-ig vizsgálni!
                        return false;
                }else return true;
        }

}
```

```
500 db 1 és 100 közötti véletlen egész szám előállítása

 67*   18    98    80    92    66    18    55    29*   64
  8     8    68    25    65    11*    2*   36    30    56
 36    10    98    67*    2*   36    27    80     9    43*
 37*   83*   49    88    44    51    39    94    92    88
 47*   53*   38    47*   95     1    81     2*   18    35
 50    77    11*   40    52   100    37*   15     9    91
 51    61*   52    27    56    30    90    33    30    56
 37*   96    45     8    76    52    74    42    24    43*
 73*   88    86    90    25    48    22    12     7*   16
 76    98   100    56     5*   85    12    89*   19*   39

 81    34    96    96    66    74    29*    4    97*   54
 85    94    28    52    83*   38    97*   90     5*   29*
 35    30    39    44    56    37*   34    84   100    47*
 81    25    55    91    88    84    39    59*   30    75
 10    86    36    44    60    71*   86    35    62    85
 51    45    77    72     8    58    42    51    73*   40
 51    59*   44    78    84    64    43*   47*   54    87
 62    52    29*   55    44    61*    6    69    50    24
 62    67*   62    97*   50    97*   80    48    73*   68
 34    62    54    95    59*   44    23*   79*   64    98

 47*   86    13*   88    68    88    15    52    41*   56
 71*   63    61*   64    11*   63    91    52    50    10
 51    91     7*   78    92    16    54    67*   28    90
 10    34    22    84    45    81    97*   95    58    57
 74    72    40    26    83*   98    95    83*   33    34
 50    67*   88    88    92    96     6    77    56    62
 57    47*    5*   26    94    48   100    74    30    68
 75    69    71*   41*   30    40    76    16     2*   12
 12    92    12    36    49    53*   50    78    77    49
 32     7*   98    82    77    28    54    37*   24    78

 29*   99    82    71*    9    67*   96    43*    6    43*
 11*   85    99    41*   57    14    27    89*   83*   46
 42    67*   62    69    20    30    13*   11*   63    83*
 15    29*   38    30    43*    7*   44    38    12    64
 65    56    14    24    50     5*    6    93    35    18
 37*   12    29*   10    28    41*   50    62    46    25
 72    97*    2*   60    45    20    43*   71*   13*   71*
 28    30    15    91    65    56    94    90    20    98
 26    14    33    32    15    96    39    19*   77    60
 67*   85    60    99    44     5*   59*   47*   75    25

 58    66    46     3*   51    68    47*   61*   31*   71*
 61*   53*   45    62    14    86    19*    7*   71*   94
 75    97*   54    83*    6    77    52    23*   24    27
 57    41*   75     2*   34    33    94    45    77    48
 64    16    82    65    92    44    33    79*    6    42
 39    69    61*   60    38    18    78    64    49    50
 47*   60    72    69    86    42    51    41*   98    98
 20    51    57    27    39    44    23*   47*   48    77
 30    89*   90    13*   66    86    86    64    55    48
 34    76    62     1    39    27    39    72    11*   29*


A leghosszabb prímsorozat kezdete: 407. elem: 47
Hossza: 6
Az elemek: 47      61    31    71    61    53
```

9. Ismétlődés nélküli véletlen sorozat előállítása: lottóhúzás

```
import java.util.Calendar;
import java.util.Scanner;

/**
 * Ismétlődés nélküli véletlen sorozat előállítása
 * Lottóhúzás
 *
 * @author Klemand
 */

public class BevGyak08 {
        public static void main(String[] args) {

                System.out.println("1) Ötöslottó\n");

                Scanner sc = new Scanner(System.in);

                System.out.println("Figyelmeztetés: 14 év alattiak nem lottózhatnak!");

                System.out.print("Kérem a születési dátumát 1982.08.12 alakban: ");
                String szDat = sc.nextLine();
                int szEv = Integer.parseInt(szDat.substring(0, 4));
                int szHo = Integer.parseInt(szDat.substring(5, 7));
                int szNap = Integer.parseInt(szDat.substring(8));

                Calendar cal = Calendar.getInstance();
                int aktEv = cal.get(Calendar.YEAR);
                int aktHo = cal.get(Calendar.MONTH) + 1; // Amerikában január a 0. hónap!
                int aktNap = cal.get(Calendar.DAY_OF_MONTH);

                int kor;
                if (szHo < aktHo) {
                        kor = aktEv - szEv;
                } else if (szHo == aktHo && szNap <= aktNap) {
                        kor = aktEv - szEv;
                } else {
                        kor = aktEv - szEv - 1;
                }

                System.out.println("A mai dátum: " + aktEv+"." + aktHo+"." + aktNap);
                System.out.println("Az Ön életkora a mai napon: " + kor + " év.");

                if (kor < 14) {
                        System.out.println("Találkozzunk " + (14 - kor) + " év múlva!\n");
                } else if (kor > 80) {
                        System.out.println("Ilyen korban már kerülni kell a túlzott izgalmakat!\n");
                } else {
                        System.out.println("Sok szerencsét a játékhoz!\n");

                        System.out.println("Tippelés:\n");

                        int[] tippek = new int[5];

                        int i, n;

                        for (i = 1; i <= 5; i++) {
                                System.out.print(i + ". tipp: ");
                                tippek[i - 1] = sc.nextInt();
                        }
                        System.out.println();
```

```
        System.out.println("Számhúzás:");
        System.out.println("5 db 1 és 90 közötti különböző véletlen egész szám előállítása\n");

        // Szelvénytörlés
        boolean[] szelveny = new boolean[90];
        for (i = 1; i <= 90; i++) {
                szelveny[i - 1] = false;
        }

        // Számhúzás
        int[] kihuzott = new int[5];

        i = 0;
        do {
                n = (int) (90 * Math.random()) + 1;
                // 1 és 90 közötti véletlen szám
                if (szelveny[n - 1] == false) {
                        // még nem volt ilyen szám
                        szelveny[n - 1] = true;
                        // ez többet nem húzható ki!
                        i++;
                        // ciklusváltozó növelése
                        kihuzott[i - 1] = n;
                }
        } while (i < 5);

        System.out.println("A lottósorsolás megtörtént.");
        System.out.println("A nyertes számok nagyság szerint:");

        int db = 0;
        for (i = 1; i <= 90; i++) {
                if (szelveny[i - 1]) {
                        db++;
                        System.out.print(i);
                        if (db < 5) {
                                System.out.print(", ");
                        }
                }

        }
        System.out.println("\n");

        System.out.println("Értékelés:\n");

        int[] nyertesek = new int[5];
        db = 0;
        int aktTipp;

        for (i = 1; i <= 5; i++) {
                aktTipp = tippek[i - 1];
                if (szelveny[aktTipp - 1]) {
                        db++;
                        nyertesek[db - 1] = aktTipp;
                }
        }

        System.out.println("A találatok száma: " + db);
        System.out.print("A nyertes tippek: ");
        for (i = 1; i <= db; i++) {
                System.out.print(nyertesek[i - 1]);
                if (i < db) {
                        System.out.print(", ");
                }
        }
        System.out.println("\n");
}
```

```
		System.out.println("2) A lottóhúzás általánosítása:");
		System.out.println("k db a és f közötti különböző véletlen egész szám előállítása\n");

		int a, f, k, i, n;

		System.out.print("Kérek egy egész számot, az alsó határ értékét: a = ");
		a = sc.nextInt();
		System.out.print("Kérek egy egész számot, a felső határ értékét: f = ");
		f = sc.nextInt();
		System.out.print("Kérem a darabszámot : k = ");
		k = sc.nextInt();
		System.out.println();

		if (k > f - a + 1) {
			System.out.println("Ennyi különböző egész szám");
			System.out.println("nem állítható elő a megadott határok között.");
		} else {
			// Táblatörlés
			boolean[] tabla = new boolean[f - a + 1];

			for (i = 0; i <= f - a; i++) {
				tabla[i] = false;
				//A konvertálást a tömbindexeléshez célszerű igazítani!
			}

			// A véletlen számok előállítása
			int[] veletlen = new int[k];

			i = 0;
			do {
				n = (int) ((f - a + 1) * Math.random());
				// 0 és f - a közötti véletlen szám

				if (!tabla[n]) {
					// még nem volt ilyen szám
					tabla[n] = true;
					// ez többet nem húzható ki!
					i++;
					// ciklusváltozó növelése
					veletlen[i - 1] = n;
				}
			} while (i < k);

			System.out.println("Az előállítás megtörtént.");
			System.out.println("Az előállított véletlen számok nagyság szerint:");

			int db = 0;
			for (i = 0; i <= f - a; i++) {
				if (tabla[i]) {
					db++;
					System.out.print(i + a);
					if (db < k) {
						System.out.print(", ");
					}
					if (db % 10 == 0) {
						System.out.println();
					}
				}
			}
			System.out.println();
		}

		sc.close();
	}

}
```

```
1) Ötöslottó

Figyelmeztetés: 14 év alattiak nem lottózhatnak!
Kérem a születési dátumát 1982.08.12 alakban: 2004.04.22
A mai dátum: 2018.4.22
Az Ön életkora a mai napon: 14 év.
Sok szerencsét a játékhoz!

Tippelés:

1. tipp: 21
2. tipp: 13
3. tipp: 28
4. tipp: 66
5. tipp: 90

Számhúzás:
5 db 1 és 90 közötti különböző véletlen egész szám előállítása

A lottósorsolás megtörtént.
A nyertes számok nagyság szerint:
2, 9, 17, 81, 85

Értékelés:

A találatok száma: 0
A nyertes tippek:

2) A lottóhúzás általánosítása:
k db a és f közötti különböző véletlen egész szám előállítása

Kérek egy egész számot, az alsó határ értékét: a = -10
Kérek egy egész számot, a felső határ értékét: f = 10
Kérem a darabszámot : k = 15

Az előállítás megtörtént.
Az előállított véletlen számok nagyság szerint:
-10, -9, -8, -7, -5, -4, -3, -2, 0, 1,
2, 3, 6, 7, 10
```

10. Statisztikai minta gyakorisági táblázata: érték szerinti indexelés és általános módszer

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

/**
 * Statisztikai minta gyakorisági táblázata: érték szerinti indexelés és
 * általános módszer
 *
 * @author Klemand
 */

public class BevGyak09 {
	public static void main(String[] args) throws IOException {

		System.out.println("1) Kvantitatív minta gyakorisági táblázata érték szerinti indexeléssel\n");

		System.out.println("A januar.txt fájl sorai a januári napi átlaghőmérsékleteket tartalmazzák.");
		System.out.println("Az értékekről tudjuk, hogy -10 és 10 Celsius fok közötti egész értékek.");
		System.out.println("Készítsünk gyakorisági táblázatot!\n");

		BufferedReader beolvas = new BufferedReader(new FileReader("januar.txt"));
		// Source menüpont Organize Imports

		int db = 31;
		int[] januar = new int[db];
		// Tudom, hogy 31 nap van januárban

		int i;
		for (i = 1; i <= db; i++) {
			januar[i - 1] = Integer.parseInt(beolvas.readLine());
		}

		beolvas.close();

		int m = -10;
		int n = 10;

		/*
		 * Először az [m;n] intervallumot átkonvertáljuk a [0 ; n-m] intervallumra, és
		 * készítünk egy egész típusú tömböt, melyben az indexnek megfelelő szám
		 * gyakoriságát tároljuk.
		 */

		int[] munkatomb = new int[n - m + 1];

		for (i = 0; i <= n - m; i++) {
			munkatomb[i] = 0;
			// először lenullázzuk
		}

		for (i = 1; i <= db; i++) {
			munkatomb[januar[i - 1] - m]++;
			/*
			 * Ez az érték szerinti indexelés: A gyakorisági tömbnek a hőmérsékleti érték
			 * eltolásával kapott indexű elemét növeljük eggyel.
			 */
		}
```

```
        // A gyakorisági táblázat elkészítése:

        int[] homerseklet = new int[db];
        int[] homersekletGyakorisag = new int[db];

        db = 0;
        for (i = 1; i <= n - m + 1; i++) {
                if (munkatomb[i - 1] > 0) {
                        db++;
                        homerseklet[db - 1] = i - 1 + m;
                        // A db-edik tulajdonságérték (hőmérséklet) visszakonvertálva
                        homersekletGyakorisag[db - 1] = munkatomb[i - 1];
                        // A db-edik tulajdonságérték gyakorisága
                }
        }

        // A gyakorisági táblázat kiíratása:
        System.out.println("A hőmérsékletek gyakorisági táblázata:\n");
        for (i = 1; i <= db; i++) {
                System.out.printf("Hőmérséklet: %3d", homerseklet[i - 1]);
                System.out.println(" Celsius fok, gyakorisága: " + homersekletGyakorisag[i - 1]);
        }


System.out.println("\n~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n");

        System.out.println("2) Általános módszer tetszőleges minta esetén\n");

        System.out.println("Az osztály.txt fájl sorai szóközökkel elválasztva egy osztály tanulóinak ");
        System.out.println("vezetéknevét, keresztnevét és lakhelyét tartalmazza.");
        System.out.println("Készítsünk gyakorisági táblázatot a lakhelyekről!\n");

        beolvas = new BufferedReader(new FileReader("osztaly.txt"));

        // Nem ismerjük előre az osztálylétszámot, de tudjuk, hogy max. 40 lehet.

        // String[] nev = new String[40]; a névre most nincs szükség!
        String[] lakhely = new String[40];

        String fsor;
        String[] daraboltFsor;
        db = 0;
        while ((fsor = beolvas.readLine()) != null) {
                // Beolvassa a sort és megvizsgálja. Ha nem null, akkor feldolgozza.
                db++;
                daraboltFsor = fsor.split(" ");
                // nev[db-1] = daraboltFsor[0] + " " + daraboltFsor[1]; most felesleges tárolni!
                lakhely[db - 1] = daraboltFsor[2];
        }

        beolvas.close();
```

```
                String[] telepules = new String[db];
                int[] telepulesGyakorisag = new int[db];

                int j;
                int telepulesDb = 0;
                for (i = 1; i <= db; i++) {
                        String aktLakhely = lakhely[i - 1];
                        j = 1;
                        while (j <= telepulesDb && !(aktLakhely.equals(telepules[j - 1]))) {
                                j++;
                        }
                        if (j <= telepulesDb) {
                                telepulesGyakorisag[j - 1]++;
                        } else {
                                telepulesDb++;
                                telepules[telepulesDb - 1] = aktLakhely;
                                telepulesGyakorisag[telepulesDb - 1] = 1;
                        }
                }

                // A gyakorisági táblázat kiíratása:
                System.out.println("A lakhelyek gyakorisági táblázata:\n");
                for (i = 1; i <= telepulesDb; i++) {
                        System.out.print("Lakhely: " + telepules[i - 1]);
                        System.out.println(", gyakorisága: " + telepulesGyakorisag[i - 1]);
                }
        }

}
```

```
1) Kvantitatív minta gyakorisági táblázata érték szerinti indexeléssel

A januar.txt fájl sorai a januári napi átlaghőmérsékleteket tartalmazzák.
Az értékekről tudjuk, hogy -10 és 10 Celsius fok közötti egész értékek.
Készítsünk gyakorisági táblázatot!

A hőmérsékletek gyakorisági táblázata:

Hőmérséklet:  -4 Celsius fok, gyakorisága: 2
Hőmérséklet:  -3 Celsius fok, gyakorisága: 3
Hőmérséklet:  -2 Celsius fok, gyakorisága: 5
Hőmérséklet:   0 Celsius fok, gyakorisága: 4
Hőmérséklet:   1 Celsius fok, gyakorisága: 8
Hőmérséklet:   2 Celsius fok, gyakorisága: 4
Hőmérséklet:   3 Celsius fok, gyakorisága: 4
Hőmérséklet:   4 Celsius fok, gyakorisága: 1

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

2) Általános módszer tetszőleges minta esetén

Az osztály.txt fájl sorai szóközökkel elválasztva egy osztály tanulóinak
vezetéknevét, keresztnevét és lakhelyét tartalmazza.
Készítsünk gyakorisági táblázatot a lakhelyekről!

A lakhelyek gyakorisági táblázata:

Lakhely: Körmend, gyakorisága: 11
Lakhely: Ják, gyakorisága: 4
Lakhely: Csákánydoroszló, gyakorisága: 3
Lakhely: Egyházasrádóc, gyakorisága: 2
Lakhely: Nádasd, gyakorisága: 3
Lakhely: Magyarszecsőd, gyakorisága: 1
Lakhely: Szentgotthárd, gyakorisága: 1
```

11. Számelméleti feladatok: euklidészi algoritmus, prímfelbontás, osztók száma

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Számelméleti feladatok: euklidészi algoritmus, prímfelbontás, osztók száma
 *
 * @author Klemand
 */

public class BevGyak10 {
	public static void main(String[] args) throws IOException {

		System.out.println("Számelméleti feladatok\n");

		System.out.println("1) Euklidészi algoritmus\n");

		System.out.println("Két megadott egész szám");
		System.out.println("legnagyobb közös osztójának");
		System.out.println("és legkisebb közös többszörösének meghatározása");
		System.out.println("az Euklidészi algoritmus segítségével");
		System.out.println("");

		Scanner sc = new Scanner(System.in);

		System.out.print("Kérek egy 1 és 100 000 közötti egész számot! a = ");
		long a = sc.nextLong();
		System.out.println("");
		System.out.print("Kérek egy másik 1 és 100 000 közötti egész számot! b = ");
		long b = sc.nextLong();
		System.out.println("");

		System.out.println("A beírt számok: a = " + a + " és b = " + b);

		long n = Math.max(a, b); // nagyobbik szám
		long k = Math.min(a, b); // kisebbik szám
		long r;
		do {
			r = n % k;
			n = k;
			k = r;
		} while (r > 0);

		long lnko = n; // az utolsó el nem tűnő maradék
		long lkkt = a * b / lnko;

		System.out.println("");
		System.out.println(a + " és " + b + " legnagyobb közös osztója " + lnko);
		System.out.println(a + " és " + b + " legkisebb közös többszöröse " + lkkt);


	System.out.println("\nXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\n");

		System.out.println("2) Prímtényezős felbontás\n");

		System.out.println("Megadott határok közötti pozitív egész számok");
		System.out.println("prímtényezős felbontásának kiíratása a primfelbontas.txt fájlba\n");

		System.out.print("Kérem az alsó határ értékét, egy 2 és 100 000 közötti egész számot: ");
		int al = sc.nextInt();
		System.out.println("");
```

```
        System.out.print("Kérem a felső határ értékét, egy 2 és 100 000 közötti egész számot: ");
        int fel = sc.nextInt();
        System.out.println("");

        sc.close();

        System.out.println("Az alsó határ: " + al + ", a felső határ: " + fel + "\n");

        int[][] primfelbontas = new int[fel][20];
        int[] tenyezokSzama = new int[fel];

        int i, j, akt, tdb;

        for (i = al; i <= fel; i++) {
                akt = i;
                // A prímtényezőket kétdimenziós tömbbe tesszük
                j = 2;
                tdb = 0;
                while (akt > 1) {
                        if (akt % j == 0) {
                                tdb = tdb + 1;
                                primfelbontas[i - 1][tdb - 1] = j;
                                akt = akt / j;
                        } else {
                                j = j + 1;
                        }
                }
                tenyezokSzama[i - 1] = tdb;
        }

        PrintWriter kiir = new PrintWriter(new FileWriter("primfelbontas.txt"));

        for (i = al; i <= fel; i++) {
                kiir.print(i + " = ");
                for (j = 1; j < tenyezokSzama[i - 1]; j++) {
                        kiir.print(primfelbontas[i - 1][j - 1] + "*");
                }
                kiir.print(primfelbontas[i - 1][tenyezokSzama[i - 1] - 1]);
                kiir.println("");
        }

        System.out.println("A prímtényezős felbontás fájlba írása befejeződött.");

        kiir.close();


System.out.println("\nXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\n");

        System.out.println("3) Prímtényezős felbontás hatványalakban és az osztók száma\n");

        System.out.println("Az előbb megadott határok közötti pozitív egész számok");
        System.out.println("prímtényezős felbontásának megadása hatványalakban,");
        System.out.println("az osztóik számának meghatározása,");
        System.out.println("és kiíratásuk az osztokszama.txt fájlba\n");

        int[] osztokSzama = new int[fel];

        kiir = new PrintWriter(new FileWriter("osztokszama.txt"));
```

```
		int pt, kit;

		for (i = al; i <= fel; i++) {
			kiir.print(i + " = ");
			tdb = tenyezokSzama[i - 1];
			pt = primfelbontas[i - 1][0];
			kiir.print(pt);
			osztokSzama[i - 1] = 1;
			kit = 1;
			for (j = 2; j <= tdb; j++) {
				if (primfelbontas[i - 1][j - 1] == pt) {
					kit++;
				} else {
					if (kit > 1) {
						kiir.print("^" + kit);
					}
					osztokSzama[i - 1] *= (kit + 1);
					pt = primfelbontas[i - 1][j - 1];
					kiir.print("*" + pt);
					kit = 1;
				}
			}
			if (kit > 1) {
				kiir.print("^" + kit);
			}
			osztokSzama[i - 1] *= (kit + 1);
			kiir.println(", osztóinak száma: " + osztokSzama[i - 1]);
		}

		System.out.println("A kiíratás befejeződött.");

		kiir.close();


	System.out.println("\nXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX\n");

		System.out.println("4) A prímek és a legösszetettebb számok kiválogatása\n");

		System.out.println("Az előbb megadott határok közötti prímszámok");
		System.out.println("és a legtöbb osztóval rendelkező számok meghatározása\n");

		System.out.println("A prímszámok:");

		int pdb = 0;
		for (i = al; i <= fel; i++) {
			if (osztokSzama[i - 1] == 2) {
				System.out.print(i + "\t");
				pdb++;
				if (pdb % 10 == 0) {
					System.out.println("");
				}
				if (pdb % 100 == 0) {
					System.out.println("");
				}
			}
		}

		System.out.println("\n");
```

```
                int maxindex = 1;
                for (i = al; i <= fel; i++) {
                        if (osztokSzama[i - 1] > osztokSzama[maxindex - 1]) {
                                maxindex = i;
                        }
                }
                int maxszam = osztokSzama[maxindex - 1];
                System.out.println("Az osztók számának maximuma az adott intervallumbam: " + osztokSzama[maxindex - 1]);
                System.out.println("");
                System.out.println("A(z) " + maxszam + " db osztóval rendelkező számok: ");

                for (i = al; i <= fel; i++) {
                        if (osztokSzama[i - 1] == maxszam) {
                                System.out.print(i + "\t");
                        }
                }
                System.out.println("\n");
        }

}
```

```
Számelméleti feladatok

1) Euklidészi algoritmus

Két megadott egész szám
legnagyobb közös osztójának
és legkisebb közös többszörösének meghatározása
az Euklidészi algoritmus segítségével

Kérek egy 1 és 100 000 közötti egész számot! a = 2012

Kérek egy másik 1 és 100 000 közötti egész számot! b = 2040

A beírt számok: a = 2012 és b = 2040

2012 és 2040 legnagyobb közös osztója 4
2012 és 2040 legkisebb közös többszöröse 1026120

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

2) Prímtényezős felbontás

Megadott határok közötti pozitív egész számok
prímtényezős felbontásának kiíratása a primfelbontas.txt fájlba

Kérem az alsó határ értékét, egy 2 és 100 000 közötti egész számot: 2012

Kérem a felső határ értékét, egy 2 és 100 000 közötti egész számot: 2040

Az alsó határ: 2012, a felső határ: 2040

A prímtényezős felbontás fájlba írása befejeződött.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

3) Prímtényezős felbontás hatványalakban és az osztók száma

Az előbb megadott határok közötti pozitív egész számok
prímtényezős felbontásának megadása hatványalakban,
az osztóik számának meghatározása,
és kiíratásuk az osztokszama.txt fájlba

A kiíratás befejeződött.

XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

4) A prímek és a legösszetettebb számok kiválogatása

Az előbb megadott határok közötti prímszámok
és a legtöbb osztóval rendelkező számok meghatározása

A prímszámok:
2017    2027    2029    2039

Az osztók számának maximuma az adott intervallumbam: 36

A(z) 36 db osztóval rendelkező számok:
2016
```

A primfelbontas.txt szövegfájl:

```
2012 = 2*2*503
2013 = 3*11*61
2014 = 2*19*53
2015 = 5*13*31
2016 = 2*2*2*2*2*3*3*7
2017 = 2017
2018 = 2*1009
2019 = 3*673
2020 = 2*2*5*101
2021 = 43*47
2022 = 2*3*337
2023 = 7*17*17
2024 = 2*2*2*11*23
2025 = 3*3*3*3*5*5
2026 = 2*1013
2027 = 2027
2028 = 2*2*3*13*13
2029 = 2029
2030 = 2*5*7*29
2031 = 3*677
2032 = 2*2*2*2*127
2033 = 19*107
2034 = 2*3*3*113
2035 = 5*11*37
2036 = 2*2*509
2037 = 3*7*97
2038 = 2*1019
2039 = 2039
2040 = 2*2*2*3*5*17
```

Az osztokszama.txt szövegfájl:

```
2012 = 2^2*503, osztóinak száma: 6
2013 = 3*11*61, osztóinak száma: 8
2014 = 2*19*53, osztóinak száma: 8
2015 = 5*13*31, osztóinak száma: 8
2016 = 2^5*3^2*7, osztóinak száma: 36
2017 = 2017, osztóinak száma: 2
2018 = 2*1009, osztóinak száma: 4
2019 = 3*673, osztóinak száma: 4
2020 = 2^2*5*101, osztóinak száma: 12
2021 = 43*47, osztóinak száma: 4
2022 = 2*3*337, osztóinak száma: 8
2023 = 7*17^2, osztóinak száma: 6
2024 = 2^3*11*23, osztóinak száma: 16
2025 = 3^4*5^2, osztóinak száma: 15
2026 = 2*1013, osztóinak száma: 4
2027 = 2027, osztóinak száma: 2
2028 = 2^2*3*13^2, osztóinak száma: 18
2029 = 2029, osztóinak száma: 2
2030 = 2*5*7*29, osztóinak száma: 16
2031 = 3*677, osztóinak száma: 4
2032 = 2^4*127, osztóinak száma: 10
2033 = 19*107, osztóinak száma: 4
2034 = 2*3^2*113, osztóinak száma: 12
2035 = 5*11*37, osztóinak száma: 8
2036 = 2^2*509, osztóinak száma: 6
2037 = 3*7*97, osztóinak száma: 8
2038 = 2*1019, osztóinak száma: 4
2039 = 2039, osztóinak száma: 2
2040 = 2^3*3*5*17, osztóinak száma: 32
```

12. Rekurzió: Fibonacci-sorozat, zárójelezés érvényessége

```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Rekurzió:
 * Fibonacci-sorozat
 * Zárójelezés érvényessége
 *
 * @author Klemand
 */

public class BevGyak11 {
	public static void main(String[] args) throws IOException {

		System.out.println("I. A Fibonacci-sorozat tagjainak meghatározása ");
		System.out.println("ciklussal és rekurzióval. \n");

		Scanner sc1 = new Scanner(System.in);

		System.out.print("Kérem a sorozat tagjának indexét (max. 45)! n = ");

		int n = sc1.nextInt();

		System.out.println("");
		System.out.println("A megadott index: " + n +"\n");

		long[] fibonacciCTomb = new long[45];

		System.out.print("1. A(z) " + n + ". tag meghatározása ciklussal: ");

		fibonacciCTomb[0] = 1;
		fibonacciCTomb[1] = 1;

		for (int i = 3; i <= n; i++) {
			fibonacciCTomb[i - 1] = fibonacciCTomb[i - 3] + fibonacciCTomb[i - 2];
		}

		System.out.println(fibonacciCTomb[n - 1]);

		System.out.println("\nFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF\n");

		System.out.print("2. A(z) " + n + ". tag meghatározása rekurzív függvénnyel: ");

		System.out.println(fibonacci(n));

		System.out.println("\nFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF\n");

		System.out.println("3. A Fibonacci sorozat első " + n + " tagjának kiíratása ");
		System.out.println("a fibonacci.txt fájlba\n");

		PrintWriter kiir = new PrintWriter(new FileWriter("fibonacci.txt"));

		for (int i = 1; i <= n; i++) {
			kiir.printf("%2d", i );
			kiir.println(". tag: " + fibonacciCTomb[i - 1]);
		}

		System.out.println("A kiíratás befejeződött.");

		kiir.close();
```

```
                System.out.println("\n{{[]()[]}}[()][]{{[[]{(((())))}]}}{[)(]}{{[]}{{[()[]}}{{{()}}}}\n");


                System.out.println("II. Egy megadott zárójelezés érvényességének eldöntése\n");

                Scanner sc2 = new Scanner(System.in);

                System.out.println("Helyes pl. {{[]()[]}}[()][] vagy {[([]{(((())))})]}");
                System.out.println("Hibás pl. {[)(]}, {([])}, {{[()[]}} vagy {{{()}}}}");
                System.out.print("Kérek egy zárójelsorozatot: ");
                String zj = sc2.nextLine();
                System.out.println("A megadott zárójelezés helyessége: " + ervenyes(zj));

                sc1.close();
                sc2.close();

                System.out.println("");

        }

        // Eljárások, függyvények

        public static long fibonacci(int n) {
                if ((n == 1) || (n == 2)) {
                        return 1;
                } else {
                        return (fibonacci(n - 2) + fibonacci(n - 1));
                }
        }

        public static  boolean ervenyes(String z) {

                if (z.equals("")) {
                        return true;
                } else {
                        String par = "";
                        if (z.contains("()")) {
                                par = "()";
                        } else if (z.contains("[]")) {
                                par = "[]";
                        } else if (z.contains("{}")) {
                                par = "{}";
                        }

                        int i;
                        if (!par.equals("")) {
                                i = z.indexOf(par);
                                return ervenyes(z.substring(0, i) + z.substring(i + 2));
                        } else {
                                return false;
                        }
                }
        }

}
```

```
I. A Fibonacci-sorozat tagjainak meghatározása
ciklussal és rekurzióval.

Kérem a sorozat tagjának indexét (max. 45)! n = 45

A megadott index: 45

1. A(z) 45. tag meghatározása ciklussal: 1134903170

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

2. A(z) 45. tag meghatározása rekurzív függvénnyel: 1134903170

FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF

3. A Fibonacci sorozat első 45 tagjának kiíratása
a fibonacci.txt fájlba

A kiíratás befejeződött.

{{[]()[]}}[()][]{{[[]{(((())))}]}{[)()]}{([])}{{[()[]}}{{{()}}}}

II. Egy megadott zárójelezés érvényességének eldöntése

Helyes pl. {{[]()[]}}[()][] vagy {[([]{(((()))})]}
Hibás pl. {[)(]}, {([])}, {{[()[]}} vagy {{{()}}}}
Kérek egy zárójelsorozatot: {[{{{[(((()))]}}}]}[[[[()]]]]
A megadott zárójelezés helyessége: true
```

A fibonacci.txt szöveges fájl:

```
 1. tag: 1
 2. tag: 1
 3. tag: 2
 4. tag: 3
 5. tag: 5
 6. tag: 8
 7. tag: 13
 8. tag: 21
 9. tag: 34
10. tag: 55
11. tag: 89
12. tag: 144
13. tag: 233
14. tag: 377
15. tag: 610
16. tag: 987
17. tag: 1597
18. tag: 2584
19. tag: 4181
20. tag: 6765
21. tag: 10946
22. tag: 17711
23. tag: 28657
24. tag: 46368
25. tag: 75025
26. tag: 121393
27. tag: 196418
28. tag: 317811
29. tag: 514229
30. tag: 832040
31. tag: 1346269
32. tag: 2178309
33. tag: 3524578
34. tag: 5702887
35. tag: 9227465
36. tag: 14930352
37. tag: 24157817
38. tag: 39088169
39. tag: 63245986
40. tag: 102334155
41. tag: 165580141
42. tag: 267914296
43. tag: 433494437
44. tag: 701408733
45. tag: 1134903170
```

13. Verselemzés: Üllői-úti fák

```java
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Verselemzés: Üllői-úti fák
 *
 * @author Klemand
 */

public class BevGyak13 {
	public static void main(String[] args) throws IOException {

		// A vers beolvasása
		System.out.println("1. feladat");
		System.out.println("A vers szerzőjének, címének és szavainak beolvasása a vers.txt fájlból");
		System.out.println("A szerző és a cím legyenenek nagybetűsek, a szavak kisbetűsek!");

		BufferedReader beolvas = new BufferedReader(new FileReader("vers.txt"));

		String sor;
		String[] daraboltSor;

		sor = beolvas.readLine();
		String szerzo = sor.trim().toUpperCase();
		sor = beolvas.readLine();
		String cim = sor.trim().toUpperCase();

		String[] szavak = new String[1000];
		int db = 0;

		while ((sor = beolvas.readLine()) != null) {
			// Beolvassa a sort és megvizsgálja. Ha nem null, akkor feldolgozza.
			if (sor.length() > 0) { // Az üres sorokat kihagyjuk.

				sor = sor.replace(',', ' ');
				sor = sor.replace('.', ' ');
				sor = sor.replace('!', ' ');
				sor = sor.replace('?', ' ');

				daraboltSor = sor.trim().split("\\s+");
				// Itt több szóköz is előállhat a sor belsejében is!

				// Beolvasás közben kisbetűssé alakítjuk a szavakat.
				int i;
				for (i = 1; i <= daraboltSor.length; i++) {
					db++;
					szavak[db - 1] = daraboltSor[i - 1].toLowerCase();
				}
			}
		}

		beolvas.close();

		System.out.println("A beolvasás megtörtént. A vers " + db + " szóból áll.\n");
```

```
System.out.println("2. feladat");
System.out.println("Írjuk ki a képernyőre a vers szerzőjét és címét,");
System.out.println("a karaktereket ciklikusan hárommal eltolva!");

int i;

for (i = 1; i <= szerzo.length(); i++) {
        System.out.print(eltol(szerzo.substring(i - 1, i).toUpperCase()));
}
System.out.println();

for (i = 1; i <= cim.length(); i++) {
        System.out.print(eltol(cim.substring(i - 1, i).toUpperCase()));
}
System.out.println("\n");

System.out.println("3. feladat");
System.out.println("A verselemzes.txt fájlban listázzuk ki a versben szereplő szavakat,");
System.out.println("mindegyiket csak egyszer! Adjuk meg a szavak előfordulásainak számát,");
System.out.println("a karaktereik számát és az ékezetes betűik számát!");

PrintWriter kiir = new PrintWriter(new FileWriter("verselemzes.txt"));

// Szólista készítése
String[] szoLista = new String[1000];
int[] szoGyak = new int[1000];
int[] ekDb = new int[1000];

int j;
int szoListaDb = 0;
for (i = 1; i <= db; i++) {
        String aktSzo = szavak[i - 1];
        j = 1;
        while (j <= szoListaDb && !(aktSzo.equals(szoLista[j - 1]))) {
                j++;
        }
        if (j <= szoListaDb) {
                szoGyak[j - 1]++;
        } else {
                szoListaDb++;
                szoLista[szoListaDb - 1] = aktSzo;
                szoGyak[szoListaDb - 1] = 1;
                ekDb[szoListaDb - 1] = ekBetuDbMeghat(aktSzo);
        }
}

kiir.println("A versben szereplő szavak és tulajdonságaik az első előfordulásuk sorrendjében:");
for (i = 1; i <= szoListaDb; i++) {
        kiir.print(szoLista[i - 1] + ": " + szoGyak[i - 1] + " alkalommal fordul elő, ");
        kiir.println(szoLista[i - 1].length() + " karakterből áll, ebből ékezetes: " + ekDb[i - 1]);
}
kiir.println("A versben összesen " + szoListaDb + " különböző szó van. \n");
System.out.println("A fájlkiírás befejeződött.\n");
```

```
System.out.println("4. feladat");
System.out.println("Állapítsuk meg, melyik a leghosszabb szó,");
System.out.println("és készítsük el a szavak hossza szerinti statisztikát!");

System.out.print("A leghosszabb szó: ");

int max = 1;
for (i = 1; i <= szoListaDb; i++) {
        if (szoLista[i - 1].length() > szoLista[max - 1].length()) {
                max = i;
        }
}
int maxHossz = szoLista[max - 1].length();
System.out.println(szoLista[max - 1] + ", hossza " + maxHossz);

int[] hosszDb = new int[maxHossz];

for (i = 1; i <= maxHossz; i++) {
        hosszDb[i - 1] = 0;
}

for (i = 1; i <= szoListaDb; i++) {
        hosszDb[szoLista[i - 1].length() - 1]++;
}

System.out.println("\nA szavak hossza szerinti statisztika:");
for (i = 1; i <= maxHossz; i++) {
        System.out.println(i + " karakteres szavak száma: " + hosszDb[i - 1]);
}
System.out.println();

System.out.println("5. feladat");
System.out.println("Határozzuk meg a szólistában az ékezetes szavak százalékos arányát");
System.out.println("két tizedes pontossággal!");

int ekezetesSzoDb = 0;

for (i = 1; i <= szoListaDb; i++) {
        if (ekDb[i - 1] > 0) {
                ekezetesSzoDb++;
        }
}
System.out.print("Az ékezetes szavak aránya a szólistában: ");
System.out.printf("%.2f", (100 * (double) ekezetesSzoDb / szoListaDb));
System.out.println("%. \n");
```

```
System.out.println("6. feladat");
System.out.println("Írjuk ki a képernyőre a vers szavainak egy véletlen permutációját!");
System.out.println("Minden sor 5 szót tartalmazzon!\n");

boolean[] kivalasztott = new boolean[szoListaDb];

for (i = 1; i <= szoListaDb; i++) {
        kivalasztott[i - 1] = false;
}

int n;
i = 0;
do {
        n = (int) (szoListaDb * Math.random()) + 1;
        if (kivalasztott[n - 1] == false) {
                kivalasztott[n - 1] = true;
                System.out.print(szoLista[n - 1] + " ");
                i++;
                if (i % 5 == 0) {
                        System.out.println();
                }
        }
} while (i < szoListaDb);
System.out.println();



System.out.println("7. feladat");
System.out.println("Rendezzük ábécé rendbe a szólistát!");
System.out.println("Az eredményt írassuk ki a verselemzes.txt fájlba!");

// A szólista rendezése abc szerint buborékos rendezéssel
String StringAsztal;
int asztal;

for (i = szoListaDb - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (abcElobbSzo(szoLista[j], szoLista[j - 1])) {

                        StringAsztal = szoLista[j - 1];
                        szoLista[j - 1] = szoLista[j];
                        szoLista[j] = StringAsztal;

                        asztal = szoGyak[j - 1];
                        szoGyak[j - 1] = szoGyak[j];
                        szoGyak[j] = asztal;

                        asztal = ekDb[j - 1];
                        ekDb[j - 1] = ekDb[j];
                        ekDb[j] = asztal;

                }
        }
}

kiir.println();
kiir.println();
kiir.println("A versben szereplő szavak és tulajdonságaik ábécé rendben:");
```

```
for (i = 1; i <= szoListaDb; i++) {
        kiir.print(szoLista[i - 1] + ": " + szoGyak[i - 1] + " alkalommal fordul elő, ");
        kiir.println(szoLista[i - 1].length() + " karakterből áll, ebből ékezetes: " + ekDb[i - 1]);
}
System.out.println("A fájlkiírás befejeződött.\n");

System.out.println("8. feladat");
System.out.println("Határozzuk meg az ábécébe rendezett szólista leghosszabb ");
System.out.println("ékezetes szakaszának kezdőszavát és hosszát!");
System.out.println("Írassuk ki a teljes szakaszt is!");

int aktTkezdet = -1; // irreális kezdőérték
int aktThossz = 0;
int maxTkezdet = -1; // irreális kezdőérték
int maxThossz = 0;

for (i = 1; i <= szoListaDb; i++) {
        if (ekDb[i - 1] > 0) {
                if (aktTkezdet == -1) {
                        aktTkezdet = i;
                }
                aktThossz++;
                if (aktThossz > maxThossz) {
                        maxTkezdet = aktTkezdet;
                        maxThossz = aktThossz;
                }
        } else {
                aktTkezdet = -1;
                aktThossz = 0;
        }
}

if (maxTkezdet > -1) {
        System.out.println("A kezdőszó: " + szoLista[maxTkezdet - 1]
                        + ", a leghosszabb ékezetes szakasz hossza: " + maxThossz);
        System.out.println("Az ékezetes szakasz:");

        for (i = maxTkezdet; i <= maxTkezdet + maxThossz - 1; i++) {
                System.out.println(szoLista[i - 1]);
        }

} else {
        System.out.println("A szólistában nincs ékezetes szó.");
}

System.out.println("");
```

```
        System.out.println("9. feladat");
        System.out.println("Csoportosítsuk a szólistát a szavak hossza szerint,");
        System.out.println("csoportonként megtartva az ábécé rendbe sorolást!");
        System.out.println("Az eredményt írassuk ki a verselemzes.txt fájlba!");

        /*
         * Rendezzük a már ábécébe rendezett szólistát buborékos rendezéssel a szavak
         * hossza szerint, mert az megtartja a korábbi szempont szerinti rendezettséget!
         */
        for (i = szoListaDb - 1; i >= 1; i--) {
                for (j = 1; j <= i; j++) {
                        if (szoLista[j].length() < szoLista[j - 1].length()) {

                                // Minden összetartozó adatot cserélni kell!

                                StringAsztal = szoLista[j - 1];
                                szoLista[j - 1] = szoLista[j];
                                szoLista[j] = StringAsztal;

                                asztal = szoGyak[j - 1];
                                szoGyak[j - 1] = szoGyak[j];
                                szoGyak[j] = asztal;

                                asztal = ekDb[j - 1];
                                ekDb[j - 1] = ekDb[j];
                                ekDb[j] = asztal;

                        }
                }
        }

        kiir.println();
        kiir.println();
        kiir.println("A versben szereplő szavak és tulajdonságaik ");
        kiir.println("a szavak hossza szerint és azon belül ábécé rendben:");
        int betuDb = 0;
        for (i = 1; i <= szoListaDb; i++) {
                if (szoLista[i - 1].length() > betuDb) {
                        betuDb = szoLista[i - 1].length();
                        kiir.println("");
                        kiir.println(betuDb + " karakteres szavak:");
                }
                kiir.print(szoLista[i - 1] + ": " + szoGyak[i - 1] + " alkalommal fordul elő, ");
                kiir.println(" ékezetes karaktereinek száma: " + ekDb[i - 1]);
        }

        System.out.println("A fájlkiírás befejeződött.");
        kiir.close();

}
```

```
	public static int ekBetuDbMeghat(String szo) {

		int db = 0;
		int i;

		for (i = 1; i <= szo.length(); i++) {
			if ("áéíóöőúüű".contains(szo.substring(i - 1, i))) {
				db++;
			}
		}
		return db;
	}

	public static boolean abcElobbSzo(String szo1, String szo2) {
		String abc = "aábcdeéfghiíjklmnoóöőpqrstuúüűvwxyz";

		int i = 1;
		while (i <= szo1.length() && i <= szo2.length()
		       && szo1.substring(i - 1, i).equals(szo2.substring(i - 1, i))) {
			i++;
		}

		if (i <= szo1.length() && i <= szo2.length()) {
			return abc.indexOf(szo1.substring(i - 1, i)) < abc.indexOf(szo2.substring(i - 1, i));
		} else {
			return szo1.length() < szo2.length();
		}
	}

	public static String eltol(String a) {
		String abc = "AÁBCDEÉFGHIÍJKLMNOÓÖŐPQRSTUÚÜŰVWXYZAÁB    ";
		return abc.substring(abc.indexOf(a) + 3, abc.indexOf(a) + 4);
	}

}
```

```
1. feladat
A vers szerzőjének, címének és szavainak beolvasása a vers.txt fájlból
A szerző és a cím legyenenek nagybetűsek, a szavak kisbetűsek!
A beolvasás megtörtént. A vers 84 szóból áll.

2. feladat
Írjuk ki a képernyőre a vers szerzőjét és címét,
a karaktereket ciklikusan hárommal eltolva!
NŐÚBÜŐODÖÁK FGBÚR
WOORKBVÜK IDN

3. feladat
A verselemzes.txt fájlban listázzuk ki a versben szereplő szavakat,
mindegyiket csak egyszer! Adjuk meg a szavak előfordulásainak számát,
a karaktereik számát és az ékezetes betűik számát!
A fájlkiírás befejeződött.

4. feladat
Állapítsuk meg, melyik a leghosszabb szó,
és készítsük el a szavak hossza szerinti statisztikát!
A leghosszabb szó: balzsamost, hossza 10

A szavak hossza szerinti statisztika:
1 karakteres szavak száma: 2
2 karakteres szavak száma: 7
3 karakteres szavak száma: 6
4 karakteres szavak száma: 5
5 karakteres szavak száma: 10
6 karakteres szavak száma: 8
7 karakteres szavak száma: 11
8 karakteres szavak száma: 5
9 karakteres szavak száma: 4
10 karakteres szavak száma: 2

5. feladat
Határozzuk meg a szólistában az ékezetes szavak százalékos arányát
két tizedes pontossággal!
Az ékezetes szavak aránya a szólistában: 58,33%.

6. feladat
Írjuk ki a képernyőre a vers szavainak egy véletlen permutációját!
Minden sor 5 szót tartalmazzon!

kedvem repül édes másoknak illatot
ne ifjúság napja fehér ezer
határ borítsa búsan balzsamost így
ti tivéletek az feleljetek dúdolva
virágos már örök fejetek ciprusát
fák haldoklik bú lombú est
kedvet tusát csírát óráin lássák
megöl legyen nyugszik is jár
lombos virág voltatok szél át
s a szívják higgyék altatót
bús nyíljatok ég sárgult hova
szagos üllői-úti minden adtatok fergeteg

7. feladat
Rendezzük ábécé rendbe a szólistát!
Az eredményt írassuk ki a verselemzes.txt fájlba!
A fájlkiírás befejeződött.
```

```
8. feladat
Határozzuk meg az ábécébe rendezett szólista leghosszabb
ékezetes szakaszának kezdőszavát és hosszát!
Írassuk ki a teljes szakaszt is!
A kezdőszó: borítsa, a leghosszabb ékezetes szakasz hossza: 7
Az ékezetes szakasz:
borítsa
bú
bús
búsan
ciprusát
csírát
dúdolva

9. feladat
Csoportosítsuk a szólistát a szavak hossza szerint,
csoportonként megtartva az ábécé rendbe sorolást!
Az eredményt írassuk ki a verselemzes.txt fájlba!
A fájlkiírás befejeződött.
```

```
A verselemzes.txt szövegfájl:

A versben szereplő szavak és tulajdonságaik az első előfordulásuk sorrendjében:
az: 7 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
ég: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 1
legyen: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
tivéletek: 1 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 1
üllői-úti: 6 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 3
fák: 7 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
borítsa: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
lombos: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
fejetek: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 0
szagos: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
virágos: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
fergeteg: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 0
ezer: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 0
fehér: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
virág: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
ti: 2 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
adtatok: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 0
kedvet: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
tusát: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
voltatok: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 0
ifjúság: 3 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 2
másoknak: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 1
is: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
így: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
nyíljatok: 1 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 1
szívják: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 2
édes: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 1
illatot: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 0
a: 5 alkalommal fordul elő, 1 karakterből áll, ebből ékezetes: 0
balzsamost: 1 alkalommal fordul elő, 10 karakterből áll, ebből ékezetes: 0
altatót: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
est: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 0
óráin: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 2
át: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 1
ne: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
lássák: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 2
bú: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 1
ciprusát: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 1
higgyék: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
örök: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 2
haldoklik: 1 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 0
sárgult: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
határ: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
nyugszik: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 0
kedvem: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
napja: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 0
már: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
szél: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 1
búsan: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
dúdolva: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
jár: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
s: 1 alkalommal fordul elő, 1 karakterből áll, ebből ékezetes: 0
megöl: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
minden: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
csírát: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 2
hova: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 0
repül: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
feleljetek: 1 alkalommal fordul elő, 10 karakterből áll, ebből ékezetes: 0
bús: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
lombú: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
A versben összesen 60 különböző szó van.
```

```
A versben szereplő szavak és tulajdonságaik ábécé rendben:
a: 5 alkalommal fordul elő, 1 karakterből áll, ebből ékezetes: 0
adtatok: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 0
altatót: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
az: 7 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
át: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 1
balzsamost: 1 alkalommal fordul elő, 10 karakterből áll, ebből ékezetes: 0
borítsa: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
bú: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 1
bús: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
búsan: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
ciprusát: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 1
csírát: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 2
dúdolva: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
est: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 0
ezer: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 0
édes: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 1
ég: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 1
fák: 7 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
fehér: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
fejetek: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 0
feleljetek: 1 alkalommal fordul elő, 10 karakterből áll, ebből ékezetes: 0
fergeteg: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 0
haldoklik: 1 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 0
határ: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
higgyék: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
hova: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 0
ifjúság: 3 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 2
illatot: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 0
is: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
így: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
jár: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
kedvem: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
kedvet: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
lássák: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 2
legyen: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
lombos: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
lombú: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
már: 1 alkalommal fordul elő, 3 karakterből áll, ebből ékezetes: 1
másoknak: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 1
megöl: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
minden: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
napja: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 0
ne: 1 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
nyíljatok: 1 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 1
nyugszik: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 0
óráin: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 2
örök: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 2
repül: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
s: 1 alkalommal fordul elő, 1 karakterből áll, ebből ékezetes: 0
sárgult: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
szagos: 1 alkalommal fordul elő, 6 karakterből áll, ebből ékezetes: 0
szél: 1 alkalommal fordul elő, 4 karakterből áll, ebből ékezetes: 1
szívják: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 2
ti: 2 alkalommal fordul elő, 2 karakterből áll, ebből ékezetes: 0
tivéletek: 1 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 1
tusát: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
üllői-úti: 6 alkalommal fordul elő, 9 karakterből áll, ebből ékezetes: 3
virág: 1 alkalommal fordul elő, 5 karakterből áll, ebből ékezetes: 1
virágos: 1 alkalommal fordul elő, 7 karakterből áll, ebből ékezetes: 1
voltatok: 1 alkalommal fordul elő, 8 karakterből áll, ebből ékezetes: 0
```

```
A versben szereplő szavak és tulajdonságaik
a szavak hossza szerint és azon belül ábécé rendben:

1 karakteres szavak:
a: 5 alkalommal fordul elő, ékezetes karaktereinek száma: 0
s: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0

2 karakteres szavak:
az: 7 alkalommal fordul elő, ékezetes karaktereinek száma: 0
át: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
bú: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
ég: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
is: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
ne: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
ti: 2 alkalommal fordul elő, ékezetes karaktereinek száma: 0

3 karakteres szavak:
bús: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
est: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
fák: 7 alkalommal fordul elő, ékezetes karaktereinek száma: 1
így: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
jár: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
már: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1

4 karakteres szavak:
ezer: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
édes: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
hova: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
örök: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 2
szél: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1

5 karakteres szavak:
búsan: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
fehér: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
határ: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
lombú: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
megöl: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
napja: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
óráin: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 2
repül: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
tusát: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
virág: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1

6 karakteres szavak:
csírát: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 2
kedvem: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
kedvet: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
lássák: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 2
legyen: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
lombos: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
minden: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
szagos: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0

7 karakteres szavak:
adtatok: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
altatót: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
borítsa: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
dúdolva: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
fejetek: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
higgyék: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
ifjúság: 3 alkalommal fordul elő, ékezetes karaktereinek száma: 2
illatot: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
sárgult: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
szívják: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 2
virágos: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
```

```
8 karakteres szavak:
ciprusát: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
fergeteg: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
másoknak: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
nyugszik: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
voltatok: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0

9 karakteres szavak:
haldoklik: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
nyíljatok: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
tivéletek: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 1
üllői-úti: 6 alkalommal fordul elő, ékezetes karaktereinek száma: 3

10 karakteres szavak:
balzsamost: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
feleljetek: 1 alkalommal fordul elő, ékezetes karaktereinek száma: 0
```

14. Érettségi mintafeladat: Autók

Az *autok.txt* szövegfájl egy autókereskedés éves beszállításait tartalmazza időrendben. A fájl első sora az év, azután egy sorban ** és szóköz után található a hónap sorszáma, * és szóköz után a napé. A hónapokat csak a hónap első szállítási napja előtt adjuk meg. A nap sorszáma után következő sorokban az aznapi szállítások találhatók. Először a szállító kétszámjegyű kódja, utána szóközzel elválasztva az autó márkája, majd újabb szóközzel elválasztva a típusa. Újabb szóköz után a darabszám, végül az egységár millió Ft-ban 1 tizedesjeggyel, tizedesvesszővel. Max. 1000 szállítás és max. 40 szállító van.

A fájl első néhány sora:

2008

** 02

* 11

77 Skoda Fabia 16 3,8

77 Skoda Octavia 8 5,4

1. Olvasd be a szövegfájt alkalmasan megválasztott adatszerkezetbe!
2. Írasd ki a képernyőre, hogy melyik volt az első és az utolsó szállítási nap!
3. Melyik szállító szállította a legtöbb autót?
4. Mennyi volt az autók összértéke? Ha a tizedesvesszős alakot a program nem ismeri fel számként, oldd meg a problémát!
5. Készíts függvényt, mely megadja az adott évben, hogy egy adott hónap adott napja az év hányadik napja! A számításba jöhető évek között minden néggyel osztható szökőév! Egyébként feltételezzük, hogy ismered, hogy hány naposak a hónapok.
6. Határozd meg, hány napig tartott és mikor kezdődött a leghosszabb időszak az évben, amikor nem volt szállítás!
7. Határod meg, melyik szállító hány napon szállított! Ha egy napon több típust is szállított, azt egy napnak kell tekinteni!
8. Ha volt olyan szállító, amelyik legalább két napon is szállított, akkor melyik szállító két szállítási napja között telt el a legkevesebb nap?
9. Mikor kezdődött és hány napos volt az a leghosszabb időszak, amikor minden nap volt szállítás?
10. Állítsd elő egy véletlen hónap 15 különböző véletlen napját! Add meg 2 tizedes pontossággal, hogy ezeknek a napoknak hány százalékában volt szállítás!
11. Készíts statisztikát, hogy az egyes szállítók hány autótípust szállítottak! Az eredményt típusszám szerint csökkenő sorrendben írasd ki a *szallitok.txt* fájlba, úgy, hogy minden szállító külön sorba kerüljön! Elöl a szállító kódja, tabulátorral elválasztva a szállított típusok száma, majd újabb tabulátor után szóközökkel elválasztva az egyes típusok a hozzájuk tartozó márkával együtt, de minden típus csak egyszer szerepeljen egy sorban!

    Pl.
    32 2 Mercedes C Mercedes A

12. Kérd be egy szállító kódját! Alkalmazott-e az év során áremelést valamely típusnál? Ha igen, add meg, melyik típusnál, és hány %-os volt a legnagyobb arányú áremelése! Sorold fel a többi szállító ennél magasabb áremeléseit a típus és az áremelés mértékének %-os arányával együtt!

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;

/**
 * Érettségi mintafeladat: Autók
 *
 * @author Klemand
 */

public class BevGyak14 {
	public static void main(String[] args) throws IOException {

		System.out.println("1. feladat");

		BufferedReader beolvas = new BufferedReader(new FileReader("autok.txt"));

		String sor;
		String[] daraboltSor, daraboltSzam;

		sor = beolvas.readLine();
		int ev = Integer.parseInt(sor.trim());

		int aktho = -1; // irreális kezdőérték
		int aktnap = -1;
		;

		int[] ho = new int[1000];
		int[] nap = new int[1000];
		int[] szallito = new int[1000];
		int[] darab = new int[1000];
		double[] ar = new double[1000];
		String[] tipus = new String[1000];

		int db = 0;

		while ((sor = beolvas.readLine()) != null) {

			daraboltSor = sor.trim().split(" ");
			if (daraboltSor[0].equals("**")) {
				aktho = Integer.parseInt(daraboltSor[1]);
			} else if (daraboltSor[0].equals("*")) {
				aktnap = Integer.parseInt(daraboltSor[1]);
			} else {
				db++;
				ho[db - 1] = aktho;
				nap[db - 1] = aktnap;
				szallito[db - 1] = Integer.parseInt(daraboltSor[0]);
				tipus[db - 1] = daraboltSor[1] + " " + daraboltSor[2];
				darab[db - 1] = Integer.parseInt(daraboltSor[3]);
				daraboltSzam = daraboltSor[4].split(",");
				ar[db - 1] = Double.parseDouble(daraboltSzam[0]) + Double.parseDouble(daraboltSzam[1]) / 10;

			}
		}

		beolvas.close();
		System.out.println("A beolvasás megtörtént. A fájl " + db + " szállítást tartalmaz.\n");
```

```
System.out.println("2. feladat");
System.out.println("Az első szállítási nap " + ev + ":" + ho[0] + ":" + nap[0] + ",");
System.out.println("az utolsó pedig " + ev + ":" + ho[db - 1] + ":" + nap[db - 1] + " volt.\n");

System.out.println("3. feladat");

int[] szallitokDb = new int[100]; // a szállítók kódja kétjegyű szám
int i;
for (i = 1; i <= 100; i++) {
        szallitokDb[i - 1] = 0;
}

for (i = 1; i <= db; i++) {
        szallitokDb[szallito[i - 1] - 1] += darab[i - 1];
}

int max = 1;
for (i = 2; i <= 100; i++) {
        if (szallitokDb[i - 1] > szallitokDb[max - 1]) {
                max = i;
        }
}

System.out.println("A(z) " + max + " szállító szállította a legtöbb autót, összesen " + szallitokDb[max - 1]
                + " darabot.\n");

System.out.println("4. feladat");

double osszertek = 0;

for (i = 1; i <= db; i++) {
        osszertek += ar[i - 1] * darab[i - 1];
}

System.out.printf("A szállított autók összértéke %,.1f", osszertek);
System.out.println(" millió Ft volt.\n");

System.out.println("5. feladat");
System.out.println("Az evNapja függvény elkészítése");
System.out.println("Az adott évben december 31. az év " + evNapja(ev, 12, 31) + ". napja volt.\n");

System.out.println("6. feladat");

int kezd = 1;
int napDb = evNapja(ev, ho[0], nap[0]) - evNapja(ev, 1, 1);

for (i = 2; i <= db; i++) {
        if (evNapja(ev, ho[i - 1], nap[i - 1]) - evNapja(ev, ho[i - 2], nap[i - 2]) - 1 > napDb) {
                napDb = evNapja(ev, ho[i - 1], nap[i - 1]) - evNapja(ev, ho[i - 2], nap[i - 2]) - 1;
                kezd = evNapja(ev, ho[i - 2], nap[i - 2]) + 1;
        }
}
if (evNapja(ev, 12, 31) - evNapja(ev, ho[db - 1], nap[db - 1]) - 1 > napDb) {
        napDb = evNapja(ev, 12, 31) - evNapja(ev, ho[db - 1], nap[db - 1]) - 1;
        kezd = evNapja(ev, ho[db - 1], nap[db - 1]) + 1;

}
int osszNap = evNapja(ev, 12, 31);
System.out.println("A leghosszabb szállításmentes időszak " + napDb + " napig tartott, ");
System.out.println("kezdőnapja " + datumMeghat(ev, osszNap, kezd) + " volt.\n");
```

```
System.out.println("7. feladat");

String[] szallitokNapok = new String[100];
for (i = 1; i <= 100; i++) {
        szallitokNapok[i - 1] = "";
}

int[] szallitokNapokDb = new int[100];

for (i = 1; i <= db; i++) {
        if (!(szallitokNapok[szallito[i - 1] - 1].contains(ho[i - 1] + ":" + nap[i - 1]))) {
                szallitokNapok[szallito[i - 1] - 1] += (ho[i - 1] + ":" + nap[i - 1] + " ");
        }
}

for (i = 1; i <= 100; i++) {
        if (szallitokNapok[i - 1].equals("")) {
                szallitokNapokDb[i - 1] = 0;
        } else {
                daraboltSor = szallitokNapok[i - 1].trim().split(" ");
                szallitokNapokDb[i - 1] = daraboltSor.length;
                System.out.println("A(z) " + i + " szállító " + szallitokNapokDb[i - 1] + " napon szállított.");
        }
}

System.out.println("\n8. feladat");

int minSzallito = -1; // irreális kezdőérték
int minNap = osszNap + 1; // irreális kezdőérték
String[] daraboltDatum;
int nap1, nap2;
int j;
for (i = 1; i <= 100; i++) {
        if (szallitokNapokDb[i - 1] > 1) {
                daraboltSor = szallitokNapok[i - 1].trim().split(" ");
                daraboltDatum = daraboltSor[0].split(":");
                nap1 = evNapja(ev, Integer.parseInt(daraboltDatum[0]), Integer.parseInt(daraboltDatum[1]));
                for (j = 2; j <= daraboltSor.length; j++) {
                        daraboltDatum = daraboltSor[j - 1].split(":");
                        nap2 = evNapja(ev, Integer.parseInt(daraboltDatum[0]), Integer.parseInt(daraboltDatum[1]));
                        if (nap2 - nap1 < minNap) {
                                minNap = nap2 - nap1;
                                minSzallito = i;
                        }
                        nap1 = nap2;
                }
        }
}

if (minSzallito > 0) {
        System.out.println("A(z) " + minSzallito
        + " szállító két szállítási napja között telt el a legkevesebb nap: " + minNap + "\n");
}
```

```
System.out.println("9. feladat");

boolean[] volt = new boolean[osszNap];
for (i = 1; i <= osszNap; i++) {
        volt[i - 1] = false;
}

for (i = 1; i <= db; i++) {
        volt[evNapja(ev, ho[i - 1], nap[i - 1]) - 1] = true;
}

int aktKezd = -1;
int aktHossz = 0;
int maxKezd = -1;
int maxHossz = 0;

for (i = 1; i <= osszNap; i++) {
        if (volt[i - 1]) {
                if (aktKezd == -1) {
                        aktKezd = i;
                }
                aktHossz++;
                if (aktHossz > maxHossz) {
                        maxHossz = aktHossz;
                        maxKezd = aktKezd;
                        maxKezd = aktKezd;
                }
        } else {
                aktKezd = -1;
                aktHossz = 0;
        }
}
System.out.print("A leghosszabb mindennapos szállítás kezdete: ");
System.out.println(datumMeghat(ev, osszNap, maxKezd) + ", hossza " + maxHossz + " nap volt.\n");

System.out.println("10. feladat");
System.out.print("Egy véletlen hónap előállítása: ");
int velHo = (int) (Math.random() * 12) + 1;
String[] hoNevek = { "január", "február", "március", "április", "május", "június", "július", "augusztus",
                "szeptember", "október", "november", "december" };
System.out.println(hoNevek[velHo - 1]);
int hossz = honapHossz(ev, velHo);
int[] velNapok = new int[15];
boolean[] kivalasztott = new boolean[hossz];

for (i = 1; i <= hossz; i++) {
        kivalasztott[i - 1] = false;
}

int velNap;
i = 0;
do {
        velNap = (int) (Math.random() * hossz) + 1;
        if (!kivalasztott[velNap - 1]) {
                i++;
                /*
                 * velNapok[i-1]=velNap;
                 * Itt is fel lehetne tölteni a kiválasztás sorrendjében,
                 * de a kivalasztott tömbből rendezve is megkaphatjuk a napokat.
                 */
                kivalasztott[velNap - 1] = true;
        }
} while (i < 15);
```

```
j = 0;
System.out.println("A kiválasztott véletlen napok:");
for (i = 1; i <= hossz; i++) {
        if (kivalasztott[i - 1]) {
                j++;
                velNapok[j - 1] = i;
                System.out.print(velNapok[j - 1] + " ");
        }
}
System.out.println();

int szallDb = 0; // a kiválasztott napokon
int aktNap;
for (i = 1; i <= 15; i++) {
        aktNap = evNapja(ev, velHo, velNapok[i - 1]);
        if (volt[aktNap - 1]) {
                szallDb++;
        }
}
System.out.printf("A kiválasztott napok %.2f", (double) 100 * szallDb / 15);
System.out.println("%-ában volt szállítás.\n");

System.out.println("11. feladat");

// A szállítóLista elkészítése

int[] szallitoLista = new int[40];
int[] szallitoTipusDb = new int[40];
String[] szallitoTipusLista = new String[40];
int szallitoListaDb = 0;
int aktSzallito;
String aktTipus;

for (i = 1; i <= db; i++) {
        aktSzallito = szallito[i - 1];
        aktTipus = tipus[i - 1];
        j = 1;
        while (j <= szallitoListaDb && aktSzallito != szallitoLista[j - 1]) {
                j++;
        }
        if (j <= szallitoListaDb) {
                if (!szallitoTipusLista[j - 1].contains(aktTipus)) {
                        szallitoTipusLista[j - 1] += (aktTipus + " ");
                        szallitoTipusDb[j - 1]++;
                }
        } else {
                szallitoListaDb++;
                szallitoLista[szallitoListaDb - 1] = aktSzallito;
                szallitoTipusLista[szallitoListaDb - 1] = (aktTipus + " ");
                szallitoTipusDb[szallitoListaDb - 1] = 1;
        }
}
```

```
// A szállítóLista rendezése típusszám szerint csökkenő sorrendbe
int asztal;
String stringAsztal;

for (i = szallitoListaDb - 1; i >= 1; i--) {
        for (j = 1; j <= i; j++) {
                if (szallitoTipusDb[j - 1] < szallitoTipusDb[j]) {

                        asztal = szallitoLista[j - 1];
                        szallitoLista[j - 1] = szallitoLista[j];
                        szallitoLista[j] = asztal;

                        asztal = szallitoTipusDb[j - 1];
                        szallitoTipusDb[j - 1] = szallitoTipusDb[j];
                        szallitoTipusDb[j] = asztal;

                        stringAsztal = szallitoTipusLista[j - 1];
                        szallitoTipusLista[j - 1] = szallitoTipusLista[j];
                        szallitoTipusLista[j] = stringAsztal;

                }
        }
}

PrintWriter kiir = new PrintWriter(new FileWriter("szallitok.txt"));

for (i = 1; i <= szallitoListaDb; i++) {
        kiir.println(szallitoLista[i - 1] + "\t" + szallitoTipusDb[i - 1] + "\t"
                        + szallitoTipusLista[i - 1]);
}

kiir.close();
System.out.println("A fájlkiírás befejeződött.\n");

System.out.println("12. feladat");

Scanner sc = new Scanner(System.in);

int kod;

System.out.print("Kérem egy szállító kétjegyű kódját: ");
kod = sc.nextInt();
sc.close();
```

```
// A szállítóPluszTípusLista elkészítése

int[] szPTszallito = new int[db];
String[] szPTtipus = new String[db];
double[] szPTkezdoar = new double[db];
double[] szPTvegar = new double[db];
int szPTDb = 0;
double aktAr;
for (i = 1; i <= db; i++) {
        aktSzallito = szallito[i - 1];
        aktTipus = tipus[i - 1];
        aktAr = ar[i - 1];
        j = 1;
        while (j <= szPTDb && !(aktSzallito == szPTszallito[j - 1] && aktTipus.equals(szPTtipus[j - 1]))) {
                j++;
        }
        if (j <= szPTDb) {
                szPTvegar[j - 1] = aktAr;
        } else {
                szPTDb++;
                szPTszallito[szPTDb - 1] = aktSzallito;
                szPTtipus[szPTDb - 1] = (aktTipus);
                szPTkezdoar[j - 1] = aktAr;
                szPTvegar[j - 1] = aktAr;
        }
}

boolean vanKod = false;
boolean vanEmeles = false;
double maxEmeles = 0;
String maxTipus = "";
for (i = 1; i <= szPTDb; i++) {
        if (kod == szPTszallito[i - 1]) {
                vanKod = true;
                if (szPTvegar[i - 1] > szPTkezdoar[i - 1]) {
                        vanEmeles = true;
                        if (szPTvegar[i - 1] / szPTkezdoar[i - 1] > maxEmeles) {
                                maxEmeles = szPTvegar[i - 1] / szPTkezdoar[i - 1];
                                maxTipus = szPTtipus[i - 1];
                        }
                }
        }
}

if (vanKod) {
        System.out.println("A megadott szállító létezik.");
} else {
        System.out.println("A megadott szállító nem létezik.");
}

if (vanEmeles) {
        System.out.printf("A megadott szállító legnagyobb áremelése %.2f", (maxEmeles - 1) * 100);
        System.out.println("% volt a(z) " + maxTipus + " esetén.");
} else {
        System.out.println("A megadott szállító nem emelt árat az év folyamán.");
}
```

```
                if (vanKod) {
                        System.out.println("Ennél nagyobb áremelések:");
                        for (i = 1; i <= szPTDb; i++) {
                                if (kod != szPTszallito[i - 1]) {
                                        if (szPTvegar[i - 1] > szPTkezdoar[i - 1]) {
                                                if (szPTvegar[i - 1] / szPTkezdoar[i - 1] > maxEmeles) {
                                                        System.out.print(szPTszallito[i - 1] + " " + szPTtipus[i - 1]);
                                                        System.out.printf(" %.2f", (szPTvegar[i - 1] /
                                                                        szPTkezdoar[i - 1] - 1) * 100);
                                                        System.out.println("%");
                                                }
                                        }
                                }
                        }
                }

        }

        public static int evNapja(int ev, int ho, int nap) {
                int[] honapok = { 0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30 };
                if (ev % 4 == 0) {
                        honapok[2]++;
                }
                int ossz = 0;
                for (int i = 1; i <= ho; i++) {
                        ossz += honapok[i - 1];
                }
                ossz += nap;
                return ossz;
        }

        public static int honapHossz(int ev, int ho) {

                int[] honapok = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
                if (ev % 4 == 0) {
                        honapok[1]++;
                }
                return honapok[ho - 1];
        }

        public static String datumMeghat(int ev, int ossz, int nap) {

                String datum = "";
                int[] honapok = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
                if (ev % 4 == 0) {
                        honapok[1]++;
                }

                int i = 12;
                while (i > 0 && ossz > nap) {
                        ossz -= honapok[i - 1];
                        i--;
                }
                return datum + ev + ":" + (i + 1) + ":" + (nap - ossz);
        }

}
```

```
1. feladat
A beolvasás megtörtént. A fájl 18 szállítást tartalmaz.

2. feladat
Az első szállítási nap 2008:2:11,
az utolsó pedig 2008:11:30 volt.

3. feladat
A(z) 99 szállító szállította a legtöbb autót, összesen 140 darabot.

4. feladat
A szállított autók összértéke 1 706,0 millió Ft volt.

5. feladat
Az evNapja függvény elkészítése
Az adott évben december 31. az év 366. napja volt.

6. feladat
A leghosszabb szállításmentes időszak 83 napig tartott,
kezdőnapja 2008:7:1 volt.

7. feladat
A(z) 25 szállító 1 napon szállított.
A(z) 32 szállító 2 napon szállított.
A(z) 64 szállító 2 napon szállított.
A(z) 77 szállító 5 napon szállított.
A(z) 88 szállító 2 napon szállított.
A(z) 99 szállító 2 napon szállított.

8. feladat
A(z) 77 szállító két szállítási napja között telt el a legkevesebb nap: 12

9. feladat
A leghosszabb mindennapos szállítás kezdete: 2008:9:22, hossza 3 nap volt.

10. feladat
Egy véletlen hónap előállítása: szeptember
A kiválasztott véletlen napok:
1 2 5 7 9 11 17 19 20 21 22 24 25 29 30
A kiválasztott napok 13,33%-ában volt szállítás.

11. feladat
A fájlkiírás befejeződött.

12. feladat
Kérem egy szállító kétjegyű kódját: 77
A megadott szállító létezik.
A megadott szállító legnagyobb áremelése 10,53% volt a(z) Skoda Fabia esetén.
Ennél nagyobb áremelések:
99 Opel Astra 15,56%
99 Opel Corsa 11,11%
```

A szallitok.txt szövegfájl:

```
77      3       Skoda Fabia Skoda Octavia VW Golf
88      2       Ford KA Ford Fiesta
64      2       Audi A6 Audi A8
32      2       Mercedes C Mercedes A
99      2       Opel Astra Opel Corsa
25      1       BMW 1
```

Folytatás: Klasszikus programozás Java nyelven
II. Az emelt szintű informatika érettségi programozási feladatainak megoldása

Klasszikus programozás

Java nyelven

II.

Az emelt szintű

informatika érettségi

programozási feladatainak

megoldása

Eclipse (Neon, Oxygen, Photon, 2019-06)

Klement András

2016 – 2020