Java SE 2

Java alapok Eclipse kornyezetben

Szabo Daniel
daniel.szabo99@outlook.com

2021. május 29.

Kivonat

Ebben a feladatsorban a Java programozas alapjaival valo ismerkedest fogjuk folytatni az Eclipse integralt fejlesztoi kornyezet (IDE) segitsegevel. A feladatsor az elozo JShell feladatsorban tanult muveletekre epitett ujabb eljarasokat es vezerloszerkezeteket mutat be, a while es for ciklusokat, illetve a tomb adatszerkezetet. A megoldashoz szukseged lesz az Eclipse IDE-re amit mar oran korabban feltelepitetted a szamitogepdre. Ha ez nem tortent meg, az Eclipse hivatalos honlapjan megtalalod a telepitesi utmutatot es szukseges allomanyokat.

Tartalomjegyzék

1. Ciklusok

Bevezetes Ebben a feladatban az if-else vezerlo szerkezet utani legfontosabb vezerlo szerkezet tipussal ismerkedunk meg, a ciklusokkal, illetven annak ket fo fajtajaval, a `while` es a `for` ciklusokkal.

Ciklusokat eljarasok ismetlesere hasznalunk, ahelyett, hogy ugyanazt az eljarast/parancssort tobbszor egymas utan beirnank a forraskodba. Ha egy ciklus kondicioja `true`, azt vegtelen ciklusnak nevezzuk, mert sosem all le, csak akkor ha mi leallitjuk.

1.1. while ciklus

A `while` ciklus addig ismetli a tartalmazott eljarast amig a megadott kondicio teljesul (`true` erteke van). A ciklusbol manualisan `break` kulcsszoval lehet kilepni, illetve a ciklus elejere `continue` kulcsszoval

```
1 int k = 1;
2 while (2717 % k != 0) {
3     k++;
4 }
5 System.out.println("2717␣legkisebb␣osztoja:␣" + k);
```

Kodreszlet 1. Pelda while ciklus 1: 2717 legkisebb osztojanak megkeresese

```
1 int k = 1;
2 while (true) {
3     if(2717 % k == 0) {
4         System.out.println("2717␣legkisebb␣osztoja:␣" + k);
5         break;
6     }
7 }
```

Kodreszlet 2. Pelda while ciklus 2: 2717 legkisebb osztojanak megkeresese

1.2. for ciklus

For ciklusokat olyankor hasznalunk amikor pontosan tudjuk, hanyszor szeretnenk egy eljarast elvegezni, vagy amikor egy adatsor minden elemere szeretnenk egy eljarast lefuttatni, nem pedig egy kiszamithatatlan kondicio alapjan dontjuk el, hogy megegyszer megismeteljuk az eljarast vagy sem. A for es while ciklus egymast teljes mertekbenn tudjak helyettesiteni (lasd: kodreszlet 6), de minden helyzetben egyertelmuen eldontheto, hogy melyikkel lesz gyorsabb es/vagy egyszerubb megoldanunk egy problemat.

```
1 for(int i = 0; i < 10; i++) {
2     System.out.println(i);
3 }
```

Kodreszlet 3. Pelda for ciklus 1: Szamok kiirasa 0-tol 10-ig

```
1 for(int i = 20; i > 10; i -= 2) {
2     System.out.println(i);
3 }
```

Kodreszlet 4. Pelda for ciklus 2: Minden masodik szam kiirasa 20-tol 10-ig

```
1 for(int h = 0; h < 24; h++) {
2     for(int m = 0; m < 60; m++) {
3         System.out.println("Az␣ido:␣" + h + ":" + m);
4     }
5 }
```

Kodreszlet 5. Pelda for ciklus 3: Ido kiirasa precenkent ejfeltol ejfelig

```
1  for(int i = 50; i < 80; i++) {
2      if(i % 3 == 0) {
3          System.out.println(i);
4      }
5  }
6
7  int i = 50;
8  while(i < 80) {
9      if(i % 3 == 0) {
10         System.out.println(i);
11     }
12     i++;
13 }
```

Kodreszlet 6. Pelda for ciklus 4: 50 es 80 kozotti 3-al oszthato szamok kiirasa konzolra while es for ciklussal

1.3. Feladatok

1.3.1. Feladat

Ird ki 0-tol 100-ig a paratlan szamokat `while` es `for` ciklus segitsegevel.

1.3.2. Feladat

Ird ki 0-tol 100-ig 5 szorzatait 3 kulonbozo megoldassal, for ciklus hasznalataval (0, 5, 10 ... 90, 95).

1.3.3. Feladat

Keresd meg es ird ki a konzolra 1142617 egesz osztoit.

1.3.4. Feladat

Mi tortenik, ha

- egy while ciklus kondicioja csak `true` vagy `false`,
- egy while ciklus kondicioja egy valtozo, de megsem vesz fel soha `false` erteket,
- egy vegtelen ciklus utan kodot irsz?

1.3.5. Feladat

Fogalmazd meg, mit csinal a kovetkezo a kovetkezo kodreszlet, majd futtasd le es ellenorizd onmagad.

```
String s = "";
while(s.length() < 100) {
    if(s.length() % 2 == 0) {
        s += "L";
    } else {
        s += "A";
    }
}
System.out.print(s);
```

Kodreszlet 7. Feladat

1.3.6. Feladat

Fogalmazd meg, mit csinal a kovetkezo kodreszlet, majd futtasd le es ellenorizd onmagad.

```
for(int x = 0; x < 100; x++) {
    for(int y = 0; y < 100; y++) {
       System.out.print("\t" + x*y);
    }
    System.out.print("\n");
}
```

Kodreszlet 8. Feladat

2. Tombok

Bevezetes Ebben a feladatban a tombokkel ismerkedunk meg. Gyakran nem egy vagy ket valtozoval dolgozunk, hanem tobb szaz vagy akar tobb millio adattal. Ilyenkor az adatokat ugynevezett adatszerkezetekbe rendezzuk, amikben az adatokat tudjuk kezelni, feldolgozni. A tomb a legegyszerubb adatszerkezet: egy megadott tipusu elemeket tartalmazo, allando hosszusagu rendezett adatsor. Egy tombben az elemek szamozva vannak, az un. indexek 0-tol kezdodnek.

2.1. Tomb letrehozasa

Ha olyan tombot hozunk letre aminek az ertekeit mar most (a program irasakor) ismerjuk, azokat egyszeruen felsorolassal megadhatjuk:

```
1 int[] lottoSzamok = {12, 41, 2, 51, 3, 12};
2
3 String[] nevek = {"Kis␣Juliska", "Nagy␣Jancsika", "Kozepes␣Jolika"};
```

Kodreszlet 9. Tomb letrehozas 1. modszere

```
1 int[] lottoSzamok = new int[] {12, 41, 2, 51, 3, 12};
2
3 String[] nevek = new String[] {"Kis␣Juliska",
4     "Nagy␣Jancsika", "Kozepes␣Jolika"};
```

Kodreszlet 10. Tomb letrehozas 2. modszere

Ha nem szeretnenk vagy nem tudjuk a tomb elemeit rogton megadni es majd csak kesobb szeretnenk feltolteni ertekekkel, megadhatjuk a tomb hosszat es a tomb a tipus alapertekevel lesz feltoltve.

```
1 int[] lottoSzamok = new int[6]; // -> {0, 0, 0, 0, 0, 0}
2
3 String[] nevek = new String[3]; // -> {null, null, null}
```

Kodreszlet 11. Tomb letrehozas 3. modszere

```
1 int[] lottoSzamok = new int[] {12, 41, 2, 51, 3, 12};
2
3 int elsoLottoSzam = lottoSzamok[0];
4 int masodikLottoSzam = lottoSzamok[1];
5 int utolsoLottoSzam = lottoSzamok[5];
```

Kodreszlet 12. Tomb elemeinek elerese indexekkel

```
1 String[] nevsor = new String[] {"Kis␣Juliska",
2     "Nagy␣Jancsika", "Kozepes␣Jolika"};
3
4 int letszam = nevsor.length;
```

Kodreszlet 13. Tomb hosszanak lekerdezese

```
1 String[] nevsor = new String[] {"Kis␣Juliska",
2     "Nagy␣Jancsika", "Kozepes␣Jolika"};
3
4 nevsor[0] = "Kis␣Jucika";
```

Kodreszlet 14. Tomb modositasa

Egy tomb hosszat a `length` valtozo segitsegevel kerhetjuk le. Figyeljunk oda arra, hogy a `String` valtozok eseteben a `length()`, ami megadja a karakterlanc hosszat, az egy fuggveny, es a vegere zarojel kerul: `"szo".length()`. Tombok eseteben viszont nem teszunk zarojelet, mert ott `length` nem egy fuggveny: `nevsor.length`.

```
1 String[] nevsor = new String[] {"Kis␣Juliska",
2     "Nagy␣Jancsika", "Kozepes␣Jolika"};
3
4 String utolsoNev = nevsor[nevsor.length - 1];
```

Kodreszlet 15. Utolso elem indexenek szamolasa

2.2. Feladatok

2.2.1. Feladat

Probalj tombot letrehozni, elemeket megadni, modositani stb. az alabbi adattipusokkal:

- `boolean`
- `String`
- `int`
- `float`

2.2.2. Feladat

Mi tortenik, ha

- 0 hosszu tombot hozunk letre (es ezt milyen modokon tehetjuk?),
- egy `String` tombbe `int`, `int` tombbe `float`, vagy `float` tombbe `int` ertekeket teszunk?

3. Tombok Feldolgozasa Ciklusokkal

Bevezetes Ciklusokat es tomboket nagyon gyakran hasznalunk egyutt, mert sokszor egy egyszeru adatsoron szeretnenk elvegezni egy muveletet minden elemre, peldaul statisztikat szeretnenk szamolni egy szamsorbol. Amikor tombokkel dolgozunk, szinten minden esetben `for` ciklust hasznalunk, mert azt kifejezetten ilyen adatszerkezetek kezelesere hoztak letre.

Egy tomb elemeit bejarhatjuk `for` ciklussal. Hagyomanyos for ciklus segitsegevel rugalmasan kezelhetjuk a bejarast:

```
1 String[] nevek = {"Kis Juliska", "Nagy Jancsika", "Kozepes Jolika"};
2
3 for(int i = 0; i < nevek.length; i++) {
4     System.out.println(nevek[i]);
5 }
```

Kodreszlet 16. Tomb elemeinek megjelenitese

Ha egy tombot az elso elemtol az utolsoig egyesevel szeretnenk bejarni, a `for-each` ciklus is hasznalhato:

```
1 String[] nevek = {"Kis Juliska", "Nagy Jancsika", "Kozepes Jolika"};
2
3 for(String nev : nevek) {
4     System.out.println(nev);
5 }
```

Kodreszlet 17. Tomb elemeinek megjelenitese for-each ciklussal

Az egyik leggyakoribb hasznalata a ciklussal valo tomb bejarasnak a statisztikak szamolasa:

```
1 int[] szamok = {32, 15, 73, 24, 12, 61, 2, 32, 72, 29};
2
3 int osszeg = 0;
4
5 for(int i = 0; i < szamok.length; i++) {
6     osszeg += szamok[i];
7 }
8
9 double atlag = (double) osszeg / szamok.length;
```

Kodreszlet 18. Atlagszamolas for ciklussal

```
int[] szamok = {32, 15, 73, 24, 12, 61, 2, 32, 72, 29};

int osszeg = 0;

for(int szam : szamok) {
    osszeg += szam;
}

double atlag = (double) osszeg / szamok.length;
```

Kodreszlet 19. Atlagszamolas for-each ciklussal

Extrem ertekek keresese egy nagyon gyakori feladat, figyelmesen olvasd el az alabbi peldakat es probalj meg elvonatkoztatni a konkret problemaktol es megfigyelni az extrem ertek kereses altalanos logikajat.

```
int[] szamok = {32, 15, 73, 24, 12, 61, 2, 32, 72, 29};

int max = 0;

for(int szam : szamok) {
    if(szam > max) {
        max = szam;
    }
}
```

Kodreszlet 20. Maximum ertek kereses

```
String[] szavak = {"kutya", "macska", "tej",
        "fa", "teherauto", "alma"};

String legrovidebb = szavak[0];

for(String szo : szavak) {
    if(szo.length() < leghosszabb.length()) {
        leghosszabb = szo;
    }
}
```

Kodreszlet 21. Legrovidebb String kereses

3.1. Feladatok

3.1.1. Feladat

Keszits egy programot, ami kiszamolja az alabbi szamok osszeget:

```
1 int[] szamok = {32, 15, 73, 24, 12, 61, 2, 32, 72, 29};
```

Kodreszlet 22. Feladat

3.1.2. Feladat

Keszits egy programot, ami megszamolja, hany ertek `true` az alabbi tombben:

```
1 boolean[] logikaiErtekek = {true, false, false, true,
2         false, true, false, true};
```

Kodreszlet 23. Feladat

3.1.3. Feladat

Keszits egy programot, ami megkeresi az 50-hez legkozelebb eso szamot az alabbi tombben:

```
1 double[] szamok = {32.2, 15.23, 73.88, 24.2, 12.2,
2         61.12, 2.402, 32.1, 72.02, 29.99};
```

Kodreszlet 24. Feladat

3.1.4. Feladat

A kovetkezo kodreszletben tobb hiba van. Keresd meg a hibakat, gondold at miert nem jo a program ugy ahogy le van irva es javitsd ki oket.

```
1  int szamok = new int[10] {32, 15, 73, 24, 12, 61, 2, 32, 72, 29};
2
3  int a = szamok(0);
4
5  if(szamok.length() > 5) {
6      System.out.println("a␣=␣" + a);
7  }
8
9  for { (i = 1; i <= szamok.length(); i++)
10     System.out.println(szamok(i));
11 }
```

Kodreszlet 25. Feladat