Java bevezető

Kabódi László

Miért Java?

- széleskörben elterjedt
  - Micro Edition - beágyazott rendszerek, régi telefonok
  - Standard Edition - PC, android ezen alapul
  - Enterprise Edition - vállalati programok, web service-ek
- multiplatform
- open source programoknál gyakori
- nincs explicit memóriakezelés, mutatók

Java tulajdonságai

- statikusan, erősen típusos
- virtuális gépen fut - bytecode-ra fordított nyelv
  - nem kell a hardvert ismerni
  - védi az operációs rendszert
- memóriakezeléssel nem kell foglalkozni - garbage collection
- objektum-orientált
  - majdnem minden objektum
  - tömb is objektum, a mérete ismert, használható, kicímzés futásidejű hiba

Primitív típusok és tömbök

- egész típusok
  - byte - 8 bit
  - short - 16 bit
  - int - 32 bit
  - long - 64 bit
- lebegőpontos típusok
  - float -32 bit
  - double - 64 bit
- boolean, char
- tömbök
  - int vektor[] = new int[3];
  - double matrix[][] = new double[4][4];
  - String szoveg = "kisróka";
  - szoveg = "Falra hányt borsó.";
- van kezdeti értékük

Osztályok

- láthatóság
  - private - senki
  - protected - leszármazott
  - public - mindenki
- minden metódus virtuális - felüldefiniálás
- csak egyszeres öröklés - extends
- több interface is lehet - implements

Vezérlési szerkezetek - if

```
if(szam < 4) {
    System.out.println("A szám kisebb, mint 4");
}
else {
    System.out.println("A szám legalább 4");
}
```

Vezérlési szerkezetek - while

```
int i=1;
while(i < 11){
    System.out.println(i + ". iteráció");
    ++i;
}
```

Vezérlési szerkezetek - for

```
for(int i = 1; i < 11; ++i){
    System.out.println(i + ". iteráció");
}
```

Vezérlési szerkezetek - for each

```
int[] tomb={25, 42, 6, 21, 10};
for(int i : tomb){
    System.out.println(i);
}
```

Példa - Hello, World!

```
public class helloworld {

    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }

}
```

Példa - osztály

```
public abstract class medve{
    protected int jollakottsag;
    public boolean ragadozo;
    public String szorszin;

    public abstract void eszik(int mennyit);
}
```

Példa - öröklés

```
public class oriaspanda extends medve{
    public oriaspanda(){
        jollakottsag = 0;
        ragadozo = false;
        szorszin = "Fekete és fehér";
    }
    public void eszik(int mennyit){
        jollakottsag += mennyit;
        System.out.println("Már "+jollakottsag+
            " egységet evett.");
        if(jollakottsag > 10){
            System.out.println("Jóllakott.");
        }
    }
}
```

Példa - öröklés

```
public class vorospanda extends medve{
    private int csikok_szama;
    public vorospanda(int csikok){
        jollakottsag = 0;
        ragadozo = false;
        szorszin = "Vörös";
        csikok_szama = csikok;
    }
    public void eszik(int mennyit){
        jollakottsag += mennyit;
    }
    public int get_csikok_szama(){
        return csikok_szama;
    }
}
```

Példa - osztályok használata

```
public class pelda{
    public static void Main(String[] args){
        pelda prog = new pelda();
        prog.run();
    }
    public void run(){
        oriaspanda panda = new oriaspanda();
        panda.eszik(5);
        panda.eszik(10);
        vorospanda pandi = new vorospanda(2);
        System.out.println(pandi.get_csikok_szama());
    }
}
```

Input/output műveletek

- java.io csomagot (vagy megfelelő részét) kell importálni
- stream alapú
- FileReader/BufferedReader
- FileWriter/PrintWriter

Példa - IO

```
import java.io.*; // ideális esetben csak ami kell
public class pelda{
    public static void Main(String[] args){
        try{
            FileReader fr=new FileReader("x.txt");
            BufferedReader r=new BufferedReader(fr);
            String s;
            while((s=r.readline())!=null)
                System.out.println(s);
            r.close();
        }
        catch(Exception e){} // külön a hibákra
    }
}
```

Megjegyzések az előzőhöz

- csak azt importáljuk, amit használunk
  - import java.io.FileReader;
  - import java.io.BufferedReader;
- a hibákat külön kapjuk el, és kezeljük
  - FileNotFoundException
  - IOException
- nem kell külön FileReader
- BufferedReader r=new BufferedReader(new FileReader("x.txt");

Példa - IO 2

```
import java.io.*;
public class pelda{
    public static void Main(String[] args){
        BufferedReader r=new BufferedReader(
            new InputStreamReader(System.in));
        try{
            PrintWriter w=new PrintWriter(
                new FileWriter("x.txt"));
            String s;
            while((s=r.readLine())!=null)
                w.println(s);
            w.close();
        }
        catch(IOException e){}
    }
```

Collections

- egyféle típusú adatból több példány tárolására
- List: ArrayList, LinkedList
- Set: HashSet, LinkedHashSet, TreeSet
- Map: HashMap, LinkedHashMap, TreeMap
- különböző függvények a Collections osztályban
  - binarySearch, sort
  - min, max
  - swap
  - reverse, rotate, shuffle
- java.util csomagban

Példa - Collections

```
import java.util.*;
public class pelda{
    public static void Main(String[] args){
        List<String> medve = new ArrayList<String>();
        medve.add("óriáspanda");
        medve.add("vörös panda");
        medve.add("barna medve");
        medve.add("fekete medve");
        Collections.sort(medve);
        Collections.swap(medve, 0, 3);
        System.out.println(medve.get(2));
    }
}
```