Programozás I. – Első ZH segédlet

Ezen az oldalon: kiírás az alapértelmezett (hiba) kimenetre, sztring konkatenáció, primitív típusok, osztály létrehozás, példányosítás, adattagok, metódusok

Kiíratás alapértelmezett kimenetre:

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

Kiíratás alapértelmezett hibakimenetre:

```
System.err.println("Error világ :(");
```

Stringek összefűzése:

```
String szoveg1 = "Hello";
String szoveg2 = "World!";

System.out.println("Szövegkonstans " + szoveg1 + szoveg2);
//Eredménye: Szövegkonstans HelloWorld!
```

Primitív típusok

```
boolean bo = true;      // logikai tipus
char c1 = 'a';          // karakter tipus
char c2 = '\u0054'; // ket bajtos, unicode karaktereket tarol!
byte b1 = 127;          // 8 bites egesz tipus
byte b2 = -128;   // minden egesz tipus elojeles!
short s = 1024;   // 16 bites egesz
int i = 0x7fffffff; // 32 bites egesz
long l = 0x7fffffffffffffffL; // 64 bites egesz
float f = 123.123f; // 32 bites lebegopontos tipus
double d = 5.0; // 64 bites lebegopontos
```

Osztály létrehozása

```
public class OsztalyNeve {
  ...
}
```

Példányosítás

```
OsztalyNeve objektumNeve = new OsztalyNeve();
```

Adattagok (általában private láthatóságúak)

```
private int elsoAttributum;
private String masodikAttr;
private boolean harmadikAttr;
```

Metódusok (operációk, függvények)

```
public boolean firstMethod(String param1, Object param2) {
      ...
      return true;
}
```

Ezen az oldalon: main függvény, parameters/default konstruktor, getter/setter metódusok, láthatóság, this/super kulcsszó

Main függvény – a program belépési pontja

```
public static void main(String[] args) {
	...
}
```

Konstruktor – Objektum inicializálása, az objektum létrejöttekor fut le

```
//Default konstruktor
public OsztalyNeve() {
	...
}

//Paraméteres konstruktor
public OsztalyNeve(String param1, int param2) {
	...
}
```

Getter

```
int number;
public int getNumber() {
	return number;
}
```

Setter

```
int number;
public void setNumber(int anotherNumber){
	number = anotherNumber;
}
```

Láthatóság

public:	minden külső osztály számára látható, szabadon használható
protected:	az adott osztály és a gyerekosztályai használhatják
nincs:	package-private- csomagon belül public, csomagon kívül private
private:	az osztályon kívül nem érhető el, csak az adott osztályon belül, ahol létrehozták

This kulcsszó

Hivatkozás az aktuális objektumra. (this.attributum)

Super kulcsszó

Hivatkozás az ősobjektumre.

```
super.method();			//Ősosztálybeli metódus hívása
```

Ősosztály konstruktorának hívása:

```
super();			//Default konstruktor
super(10,"Szoveg");		//Paraméteres konstruktor
```

Ezen az oldalon: Vezérlési szerkezetek (for, if, while, switch), Összehasonlítás, Szöveg konvertálása számmá

Vezérlési szerkezetek

```
//While ciklus
while( szam < 10 ) {
   ...
}

//For ciklus
for(int i = 0; i < 20; i++) {
    ...
}

//Egyszerű if-else
int szam = 10;

if( szam == 5 ) {
    ...
} else if ( szam == 6 ) {
    ...
} else {
    ...
}

//switch
int a = 8;
String szoveg;

switch(a) {
    case 5 : szoveg="ot"; break;
    case 6 : szoveg="hat"; break;
    default : szoveg="nincs"; break;
}
```

Összehasonlítás

```
int elso=4;
int masodik=100;

//Primitív típusoknál lehet használni az ==, !=, <, >, stb..
if(elso == masodik) {
      ...
}

String str = "Szoveg";

//Objektumoknal .equals() metodust kell hasznalni
if(str.equals("Szoveg")){
      ...
}
```

String konvertálása:

```
String str = "20";
int iszam = Integer.parseInt(str);
double dszam = Double.parseDouble(str);
```

Ezen az oldalon: overloading, overriding, öröklődés, static, final kulcsszó, konstans létrehozás

Overloading

```
void osszeAd(int a, int b) {
      ...
}

void osszeAd(int a, int b, int c){
      ...
}

void osszeAd(int a, double b){
      ...
}
```

Overriding

Már meglévő metódus felülírása egy gyerekosztályban

Azonos név **és** paraméterlista

Öröklődés: extends kulcsszóval

```
class OsOsztaly {
    public void foo() {
        // mukodes
    }
}
class GyerekOsztaly extends OsOsztaly {
    public void foo() {
        // valami mas mukodes
    }
}
```

Static

Adattag esetén: ugyanazon a helyen tárolódik az összes példány. Emiatt ugyanaz lesz az értéke bármely objektumpéldány esetén

Metódus esetén: A metódus példányosítás nélkül is meghívható

Final

Adattag esetén: az adattag kezdeti értékét nem változtathatjuk meg (adni kell kezdőértéket!)

Metódus esetén: a gyerekosztályban nem lehet felüldefiniálni (minden private metódus impiciten final)

Osztály esetén: nem lehet gyerekosztálya

Konstans létrehozás

Konstans változó a final kulcsszóval.

Valódi konstans létrehozása a static és final kulcsszavakkal.

Ezen az oldalon: toString() felüldefiniálása, tömb létrehozás, tömb elemszáma, csomagba helyezés,csomag importálás, teljes példa, parancssori argumentumok

toString() metódus felüldefiniálása

```
public String toString() {
      return "Szoveg";
}
```

Parancssori argumentumok kezelése

```
public class MainOsztaly {
    public static void main(String args[]){
        int[] szamTomb = new int[args.length];
        //double[] doubleTomb = new double[args.length];

        for(int i = 0; i < args.length; i++) {
            szamTomb[i] = Integer.parseInt(args[i]);
            //doubleTomb[i] = Double.parseDouble(args[i]);
        }

        for(int i = 0; i < szamTomb.length; i++) {
            // Itt már bármit tehetünk a tömb elemeivel, hiszen számok.
        }
    }

}
```

Tömb létrehozása

```
//n elemű tömb létrehozása

//Primitív adattípus
int[] tomb = new int[n];

//Saját típus
OszalyNeve[] sajatTomb = new OsztalyNeve[n];
```

Tömb elemszáma:

```
int[] tomb = new int[n];
int tombHossza = tomb.length;
```

Csomag elnevezés: fordított domain

`prog1.inf.u-szeged.hu` → `hu.u_szeged.inf.prog1`

Melynek mappaszerkezete: `hu/u_szeged/inf/prog1`

Csomagba helyezés (Legelső nem komment sor)

```
package csomagnév;
```

Csomag importálása (ha használni szeretnénk valamit egy másik csomagból)

```
import csomagnév.OsztályNév;
import csomagnév.*;
```

Ezen az oldalon: teljes példa

Teljes példa (Zh előtti órai Bor, Aszu, BorMain osztályok)

```
//
//Bor.java
//
package ital;
public class Bor {

	private String fajta;
	private int evjarat;

	public Bor(String fajta, int evjarat) {
		this.fajta = fajta;
		this.evjarat = evjarat;
	}

	public String getFajta() {
		return fajta;
	}

	public void setFajta(String fajta) {
		this.fajta = fajta;
	}

	public int getEvjarat() {
		return evjarat;
	}

	public void setEvjarat(int evjarat) {
		this.evjarat = evjarat;
	}

	@Override
	public String toString() {
		return "Bor [fajta=" + fajta + ", evjarat=" + evjarat + "]";
	}
}
```

```
//
//Aszu.java
//
package ital;
public class Aszu extends Bor{

	private int puttony;

	public Aszu(int evjarat, int puttony) {
		super("aszu", evjarat);
		this.puttony = puttony;
	}

	public int getPuttony() {
		return puttony;
	}
```

```
	public void setPuttony(int puttony) {
		this.puttony = puttony;
	}

	public String toString() {
		return "Aszu [evjarat=" + getEvjarat() + ", puttony=" +
			this.puttony + "]";
	}
}
```

```
//
//Bor Main
//
// meghivni: java BorMain Aszu 4 1999 Bikaver 2009 Bikaver 2011 Aszu 6 2011
import ital.Aszu;
import ital.Bor;

public class BorMain {

    public static void kiirBor(Bor bor){
        System.out.println(bor);
        //System.out.println(bor.toString());
    }

    public static void main(String[] args) {
        for (int i = 0; i < args.length; i++) {
            if(args[i].equals("Aszu")){
                int puttony = Integer.parseInt(args[i+1]);
                int evjarat = Integer.parseInt(args[i+2]);
                Bor aszu = new Aszu(puttony, evjarat);
                kiirBor(aszu);
                i +=2;
            } else {
                int evjarat = Integer.parseInt(args[i+1]);
                Bor bor = new Bor(args[i], evjarat);
                kiirBor(bor);
                i+=1;
            }
        }
    }
}
```

A segédanyagot készítette: Antal Gábor. A segédanyagban előfordulhatnak esetlegesen hibák, elírások, ha valaki talál, akkor kérem jelezze felém az *antal@inf.u-szeged.hu* címen.

Mindenkinek sok sikert kívánok a zh-hoz.