02-06: Javában programozunk

Primitív típusok

- `int, double, boolean, char`
- egyetlen értéket tárolnak

Alap OOP adattípusok

- `Integer, Double, Boolean, Char, String`
- egyetlen értéket tárolnak, de *null* is lehet az értékük

Objektum **osztály** (class)

- az osztály egy saját adattípus amit mi készítünk el
- több adatot és műveletet is tartalmazhat
- adatok: **attribútumok**
- műveletek: **metódusok**
- konstruktor: speciális metódus ami akkor fut le, ha egy példányt létrehoznak az osztályból
- mindig NagyBetűs
- olyan, mint egy dolog "tervrajza"
- osztály pld. az Ember

Objektum **példány** (instance)

- a példány egy "változó", típusa: Integer, String, Ember...
- minden példány egymástól független
- mindig kisBetűs
- olyan, mint egy dolog "legyártása" a tervrajz szerint
- a new operátorral a konstruktort meghívva gyártódik le egy újabb példány
- példány pld. `Ember john=new Ember()`

Egységbe zárás

- **(Encapsulation)**
- Adatok (attribútumok) és a hozzájuk tartozó műveletek (metódusok) egy helyen van

```
public class Ember {
   String name;  // adat
    void igyal(Innivalo innivalo) { // művelet
```

Öröklés

- **(Inheritance)**
- Egy leszármazott osztály megörökli az ős minden metódusát és attribútumát

```
public class Bor extends Innivalo {
```

Többalakúság

- **(Polymorphism - polimorfizmus)**
- Egy leszármazott osztály megváltoztathajta a megörökölt metódusok működését, így ugyanaz a metódus máskét viselkedik

```
@Override
public String toString() {
    return name; // saját toString
}
```

02-06: Javában programozunk

Nyomi programok

- Konzolról beolvas, kiszámol, kiír
- Általában egyszerű matematikai problémák
- Nem igényel tervezést
- *Nem OOP*
- Nehezen bővíthető (1000 kör területe?)

```
import java.util.Scanner;

public class Area {
  public static void main(String[] args) {
    Scanner in=new Scanner(System.in);
    System.out.print("Sugár? ");
    double radius=in.nextDouble();
    System.out.println("Terület="
        +(radius*radius*Math.PI));
    System.out.println("Kerület="
        +(radius*21*Math.PI);
  }
}
```

Alap OOP programok

- Általában modellez valamit
  - Szereplők fajtái: **osztályok**
  - Szereplők tulajdonságai: **attribútumok**
  - Szereplők tevékenységei: **metódusok**
  - Szereplők: **példányok**
- Meg van tervezve OOP-sen
- Az objektum-példányok egymással kölcsönhatásban állnak, mint a való életben (john.iszik(bor))
- Az öröklés miatt egy dolgot csak egyszer kell leírni
- Könnyen bővíthető

Ős-osztály

- Minden olyan tulajdonság és kód ami minden Innivalóra egyformán kelleni fog.

```
public class Innivalo {
  private double felszolgaltMeret=5; // dl
  private double alkoholTartalom=4; // %
  private String nev;
  protected Innivalo(String nev,
    double felszolgaltMeret,
    double alkoholTartalom) {
      this.nev=nev;
      this.felszolgaltMeret=felszolgaltMeret;
      this.alkoholTartalomSzazalek=alkoholTartalom;
  }
}
```

Leszármazott osztály

- Csak annyi plusz kód, amennyivel a Bor különbözik bármely innivalótól.

```
public class Bor extends Innivalo {
    public Bor() {
          super("Kannás bor",1,12); // 1 dl, 12%
    }
}
```

Objektum példányok kölcsönhatása

- Jack egy Ember példány, aki megiszik egy Bor példányt.

```
Ember jack=new Ember("Jack",1.2);
jack.igyal(new Bor());
```

02-06: Javában programozunk

Ellenőrző-lista OOP tervezéshez

- Definiáld a problémát
  - Közös nyelv – találd ki, hogy hogyan fogod nevezni a dolgokat, és ragaszkodj hozzá.
- Kik a szereplők?
  - A szereplőknek milyen tulajdonságai vannak? – **attribútumok**
  - A szereplők milyen műveleteket végeznek? – **metódusok**
  - Mik azok az attribútumok és metódusok, amik különféle szereplőknél közösek? (Pld. minden Innivalónak van neve.) – **ős-osztályok**
  - Miben különböznek a szereplők? – **leszármazott osztályok**, további attribútumok, metódusok, örökölt metódusok **override**-olása
- Milyen lehetséges hibák fordulhatnak elő a metódusokban?
  - Dobj **exception**-t ha valami nem jó!
- Ki változtathatja az attribútumokat, ki használhatja a metódusokat?
  - Állítsd be a láthatóságot – **private, protected, public**
- Egy osztály igyekszik elrejteni a részleteket – minden attribútumot is. Használj **getter**eket és **setter**eket ha valamelyik attribútumot mégis kívülről is látni kell.
- **JavaDoc kommentek** minden public és protected attribútumra és metódusra
- Csinálj saját **toString()**-et!
- Sok egyforma objektum példány kezelésére használj konténereket, pld. **List<xxx>**-t és **for-each** ciklust!

Jó OOP program

- Nem látszanak benne érdektelen részletek (pld. matematikai számítások) – ezeket metódusokba rejtettük
- Simán olvasható, hiszen objektum példányok közti kölcsönhatást ír le
- Könnyen bővíthető: mind mennyiségben (400 emberes party) mind minőségben (mosdószünet-kiszámolása 6 sorból)

"Egy programozási nyelv akkor alacsony szintű, ha arra kényszeríti az embert, hogy az érdektelen dolgokra is odafigyeljen." – Alan J. Perlis

A Java magasszintű nyelv, használjuk magas szinten!