Budapesti Műszaki és Gazdaságtudományi Egyetem

Közlekedés- és Járműirányítási Tanszék

Számítástechnika I.

BMEKOKAA152

BMEKOKAA119

Infokommunikáció I.

BMEKOKAA606

Dr. Bécsi Tamás

9. előadás

AZ OOP alapelvei

- **Egységbezárás**(Encapsulation)
  - Az adatokat es a hozzájuk tartozó eljárásokat egyetlen egységben (osztályban) kezeljük. Az osztály adatmezői tároljak az adatokat, a metódusok kommunikálnak a külvilággal.
- **Öröklés**(Inheritance)
  - Az osztály továbbfejlesztése. Ennek során a származtatott osztály örökli az ősosztálytól az összes attribútumot, metódust. Ezeket azonban újakkal is kibővíthetjük, ill. bizonyos szabályok mellett az örökölt metódusokat is megváltoztathatjuk.
- **Sokalakúság**(Polymorphism)
  - Ugyanarra a metódusra a különböző objektumok különbözőképpen reagáljanak. A származtatás során az ősosztályok metódusai képesek legyenek az új, átdefiniált metódusok használatára újraírás nélkül is.
  - Ezt virtuális (vagy dinamikus) metódusokkal érhetjük el.

Osztályok


```
class osztálynév [: szülő, …]
{
   • Osztálytaglista
      • Konstruktorok
      • Destruktorok
      • Adattagok
      • Metódusok
};
```

- Minden osztály egy közös ősből, az objectosztályból származik

Hozzáférési szintek osztályok esetében

1. A **private** tagokat csak az adott osztályon belülről érhetjük el.
2. Az osztályok publikus **public** mezőit bárhonnan elérhetjük, módosíthatjuk.
3. A **protected** mezők az osztályon kívüliek számára nem elérhetőek, míg az utódosztályból igen.
4. Az **internal** mezőket a készülő program osztályaiból érhetjük el.
5. A **protected internal** elérés valójában egy egyszerű vagy kapcsolattal megadott hozzáférési engedély. A mező elérhető a programon belülről, vagy az osztály utódosztályából! (Egy osztályból természetesen tudunk úgy utódosztályt származtatni, hogy ez nem tartozik az eredeti programhoz.)

Egyszerű példa
Adattagok


```
class Ember
    {
        private int eletkor;
        public string nev;
    }
……
Ember e = new Ember(); //automatikus konstruktor
e.nev = "Béla"; //működik
e.eletkor= 12; //nem működik
```

Egyszerű példa
Metódusok


```
 public int getEletkor()
        {
            return eletkor;
        }

public void oregszik()
        {
            eletkor++;
        }

public void kiir()
        {
            Console.WriteLine("{0} {1}", nev, getEletkor());
        }
```

Konstruktorok

- Osztályok esetén ez a kezdőérték-adás nem biztos, hogy olyan egyszerű, mint volt elemi típusok esetén, ezért ebben az esetben egy függvény kapja meg az osztály inicializálásával járó feladatot.
- Ez a függvény az osztály „születésének” pillanatában automatikusan végrehajtódik, és konstruktornak vagy konstruktor függvénynek nevezzük.
- A konstruktor neve mindig az osztály nevével azonos.
- Ha ilyet nem definiálunk, a keretrendszer egy paraméter nélküli automatikus konstruktort definiál az osztály számára.
- Az osztály referencia típusú változó, egy osztálypéldány létrehozásához kötelező a **new** operátort használni, ami egyúttal a konstruktor függvény meghívását végzi el.

Konstruktorok Példa


```
public Ember(int peletkor)
{
     eletkor = peletkor;
}

public Ember(Ember e)
{
    eletkor = e.eletkor;
    nev = e.nev;
}
```