



# ***Programozási nyelvek II. (Java)***

*Dancs Sándor*  
*Nyíregyházi Egyetem*  
*Matematika és Informatika Intézet*





# 1. Konzultáció





# Bemutakozás, a tananyag beosztásának áttekintése és a követelmények ismertetése

## Kurzusinformáció





# Az előző anyag (OIN1101 Programozási nyelvek 1. (C)) rendszerező összefoglalása



# Algoritmusok

# Programok





# A programozás alapjai

- Változók
- Típusok
- Kifejezések



# Vezérlési szerkezetek

- Szekvencia
- Elágazás
- Ciklus



# Programozási tételek

- Eldöntés
- Keresés
- Kiválasztás
- Kiválogatás
- Másolás



# Programozási tételek

- Megszámlálás
- Összegzés
- Rendezés
- Szélsőérték keresés (minimum, maximum)
- Sorozatszámítás



# Tömbök

- Inicializálás
- Indexelés (túlindexelés)
- Programozási tételek megvalósítása



# Főprogram Alprogram (függvény)

- Deklarálás, definiálás
- Függvényhívás
- Paraméterek
- Érték és mellékhatás



# Struktúrák

- Definíció és példányosítás
- Inicializálás
- Értékadás
- Operátorok



# Mutatók (pointerek)

- Indirekció
- Operátorok
- Paraméterátadás
- Címaritmetika
- NULL pointer



# Sztringek

- Inicializálás
- Másolás
- Paraméterátadás
- Sztringkezelő függvények



# Rendezések

- Buborékrendezés
- Közvetlen kiválasztással
- Kertitörpe-rendezés
- Összefésülés
- Gyorsrendezés



# Rekurzió

- Rekurzív hívás (verem)
- Leállási feltétel (báziskritérium)



# Operátorok

- Precedencia
- Asszociativitás
- Polimorfizmus (többalakúság)
- Konverzió (cast)
- Érték és mellékhatás
- Rövidzár kiértékelés



# Dinamikus memóriakezelés

- Függvények (malloc() és a free())
- Dinamikus tömb
- Memóriaterületek (data, stack, heap)



# Fájlkezelés

- Megnyitási módok
- Fájlkezelő függvények



# Modulok

- Főprogram és modulok
- Forrásállományok
- Deklarációk és definíciók
- Fejlécfájlok



# Programok életciklusa

- Specifikáció
- Fejlesztés
- Továbbfejlesztés



# Dokumentáció

- Fejlesztői
- Forráskód
- Felhasználói
- Tesztelési



# Láncolt listák

- Listaműveletek
- Duplán láncolt listák
- Listák alkalmazásai



# Bináris fák

- Bejárások
- Műveletek
- Keresőfa
- Fák alkalmazásai
- Kettős indirekció
- Hash táblák



# Állapotgépek

- Tervezés
- Leképezések



# Reguláris kifejezések

- Szintaxis
- Használat



# Generikus algoritmusok

- Függvénytípus
- Predikátum
- Programozási tételek generikus változatai
- Generikus algoritmusok tömbökön



# Visszalépő keresés (backtrack)

- Általános megfogalmazása
- Alkalmazási lehetőségek



# Objektum-orientált programozás alapfogalmai, elvei, objektum fogalma. Osztály, egységbezárás, láthatóság és információrejtés fogalma





# Objektumorientált paradigma

- Objektum
  - Állapot (tulajdonság(ok), azaz változó(k))
  - Viselkedés (metódus(ok), azaz függvény(ek))
- Osztály
  - Hasonló objektumok közös változóit és metódusait leíró sablon

Az objektum egy osztály példánya!

Az osztály az objektum típusa!



# Egységbe zárás (encapsulation)

A változók és a metódusok osztályba való becsomagolása

# Információrejtés

Az objektum belső működésének elrejtése a  
külvilág előtt

Ennek megfelelően az osztály két részből áll:

- Publikus interfész (mire jó, hogyan használható)
- Implementáció (hogyan működik)



# Felelősség

Minden osztálynak egyetlen felelősséget kell ellátnia (SRP, Single responsibility principle, Egyetlen felelősség elve)

# Öröklődés

Osztályok közötti kapcsolat, egy szülő (ős) és gyerek (leszármazott) között

A származtatott osztály mindent örököl az őstől (tulajdonságokat, viselkedést és kapcsolatokat), de az örökölt metódusokat felüldefiniálhatja, illetve kiegészítheti újabbakkal is

(LSP, Liskov Substitution Principle, Liskov helyettesítési elve)

# Öröklődés

- Többszörös öröklődés

Egy osztály nem csak egy, hanem több őszosztálytól is örököl

- Egyszeres öröklődés

Egy osztálynak nem lehet több őszosztálya, de több leszármazott osztálya igen

A Java nyelvben csak egyszeres öröklődés van!

# Láthatóság

| Módosító        | Osztály | Csomag | Leszármazott | Mindenki |
|-----------------|---------|--------|--------------|----------|
| private         | Igen    | Nem    |              |          |
| nincs (default) | Igen    |        | Nem          |          |
| protected       | Igen    |        |              | Nem      |
| public          | Igen    |        |              |          |

|          |                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| static   | Osztályszintű tulajdonság vagy metódus                                                                                                          |
| final    | Változó (egyszer kaphat értéket)<br>Metódus (leszármazott osztály nem definiálhatja felül)                                                      |
| abstract | Metódus (nincs definiálva, azaz nincs törzse)<br>Osztály (abstract, ha abstract metódusokat tartalmaz)<br>Abstract osztály nem példányosítható! |



# Tagváltozók és tagfüggvények

## A Java nyelv kialakulása



# Tagváltozók és tagfüggvények

- Osztálytagok
  - Nem az objektumokhoz, hanem az osztályhoz tartoznak
  - Szerepel előttük a static módosító
  - Az osztály példányosítása nélkül is használhatók
- Példánytagok
  - Nem az osztályhoz, hanem az objektumokhoz köthetők
  - Nem szerepel előttük a static módosító



# Osztálytagok

- Osztályváltozók
  - Az objektumok egyező tulajdonságait tárolják
  - Egy példányban vannak jelen a memóriában
  - Minden objektum eltudja érni őket
- Osztálymetódusok
  - Az osztály példányosítása nélkül is hívhatók
  - Közvetlenül csak osztálymetódust hívhatnak
  - Közvetlenül csak osztályváltozót érhetnek el

# Példánytagok

- Példányváltozók
  - Az objektumok állapotait írják le
  - Minden objektum egyedi, az állapotuk egyezésekor is
  - Minden objektumnak (állapot) külön memóriaterülete van
- Példánymetódusok
  - Az objektumok viselkedéseit írják le
  - Az osztály- és példánytagokat egyaránt elérik
  - Paraméter: this referencia

# A Java nyelv kialakulása

- Tisztán objektum orientált nyelv, eredetileg a C++ leváltása tervezték, de a programok lassabban futnak
- Már meglévő nyelvek (C/C++, Smalltalk), jobb tulajdonságait vette át, a rosszabbak mellőzésével
- A komplexitás csökkentése volt a cél
- Nyelvi szinten támogatott lett a hibakezelés
- Szempont volt az egyszerű használat és szintaxis (C)
- Jelenleg az API (Application Programming Interface, Alkalmazásprogramozási Felület) támogatás óriási



# A Java nyelv jellemzői

- Objektorientált
- Előfordított
- Értelmezett
- Semleges architektúrájú
- Hordozható
- Egyszerű
- Robusztus
- Biztonságos
- Többszálú
- Dinamikus



Köszönöm a figyelmet!





## 2. Konzultáció





# A Java környezet és technológia alapelemei

## Változók, típusok, vezérlési szerkezetek, eljárások





# A Java környezet és technológia alapelemei

# Alkalmazás létrehozása és futtatása

Forrásállomány -> Fordító -> JVM (Java Virtual Machine, Java Virtuális Gép) -> Operációs rendszerek

Pl.:

```
javac HelloWorld.java -> HelloWorld.class
```

```
java HelloWorld.class
```



# Java technológia

A Java technológia egyaránt programozási nyelv és platform

## Programok

- Asztali alkalmazások
  - Beágyazott programok: applet, servlet, midlet
- 

# Java programozási nyelv

Fordítás és értelmezés:

Forrásállomány -> Fordító (javac) -> Bájtkód -> JVM  
(Java Virtual Machine, Java Virtuális Gép) (java) ->  
gépi kód -> Operációs rendszer

JIT (Just in time) fordító

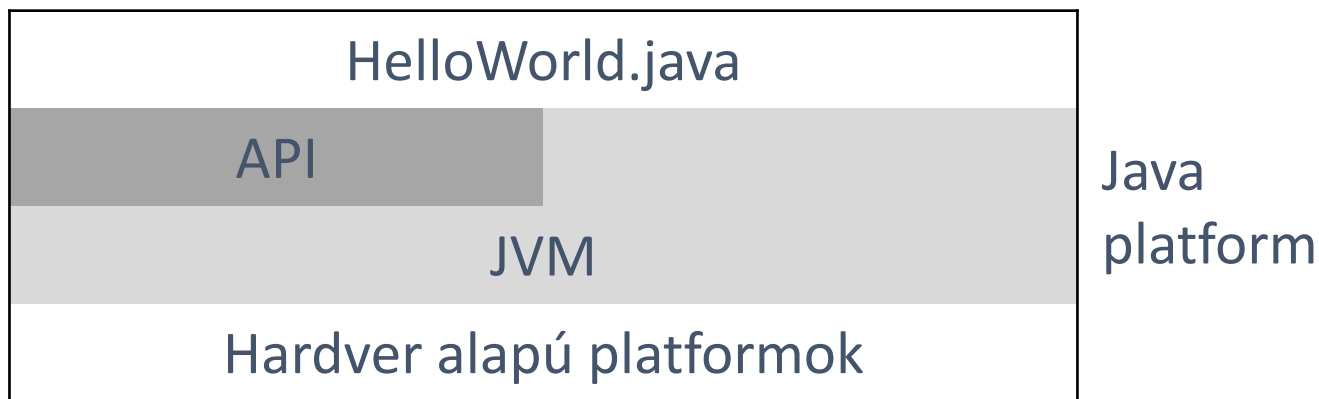


# Java platform (teljesen szoftverplatform)

Komponensek:

- Java Virtual Machine
  - Java Application Programming Interface  
(csomagokba szervezett osztályok és interfészek)
- 

# A Java platform működése





# HelloWorld Java program

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```



# Megjegyzések

`/*megjegyzés*/`

`//megjegyzés`

`/**dokumentáció*/`

`javadoc -> dokumentáció (HTML, XML, RTF, PDF, TeX, ...)`

# Osztálydefiníció

```
public class HelloWorld {  
    //Jelenleg nincsenek változóink és csak  
    egyetlen metódusunk (main) van  
    public static void main(String[] args) {  
        System.out.println("Hello World!");  
    }  
}
```



# Változók, típusok, vezérlési szerkezetek, eljárások



# Objektum orientált programozás Java nyelven

## Osztályok, példányok, konstruktorok





A Java objektum orientált  
programozási nyelv, nem a kódoláson  
van a hangsúly, hanem a tervezésen!



A tervezés során, folyamatosan fejlődő  
modell alapján adjuk meg a szoftver  
megvalósításának módját!



# UML

(Unified Modeling Language,  
Egységes Modellezési Nyelv)

[uml.org](http://uml.org)



# Adattagok, tagfüggvények

kötőjel: private  
pluszjel: public  
kettőskereszt: protected  
hullámvonal: package private  
aláhúzás: static

| Osztály neve (dőlt ha, absztrakt)                                                                  |
|----------------------------------------------------------------------------------------------------|
| - <u>adattag: Típus</u><br># <u>adattag: Típus</u>                                                 |
| + <u>függvény(paraméter: Típus, ...): Típus</u><br>~ <u>függvény(paraméter: Típus, ...): Típus</u> |



Asszociáció



Aggregáció



Kompozíció



Öröklődés

Függőség

# Konstruktor

- Megfelelő értékekkel inicializálja az objektumok adattagjait
- Neve mindig megegyezik az osztály nevével
- Mindig meghívásra kerül, amikor az allokáció (new) megtörténik
- Default konstruktor
- Több konstruktor az osztályban (overloading, kiterjesztés)



Köszönöm a figyelmet!

