

Java-Beginner

- Algoritmus, program
- Gyakori algoritmusok
- Adat, változó, metódus
- Tömb és használata
- Vezérlési szerkezetek
- Objektumorientált alapfogalmak
- Java nyelvi alapok
- Java osztályok készítése és használata

Java története

Háttér

- Szoftverkrízis
- Új programozási paradigmák: objektumorientált programozás
- 1991 Sun titkos projektje, James Gosling
- Első java: Mosaic böngészőben

Célok

- Egyszerű, objektumorientált
- Robusztus, biztonságos
- Architektúra-semleges, hordozható
- Nagy teljesítményű
- Interpretált, többszálú és dinamikus

Jelen

- Oracle
- Főleg nagyvállalati backend rendszerek
- Mobil: Android
- Kliens oldal: Swing, JavaFX
- 10 millió Java fejlesztő, 15 milliárd eszköz

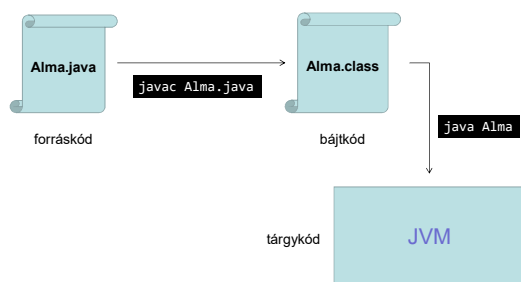
Verziószámok

- Java SE 8u172 (1.8.0_172-b11)
- Java 8
- Már a Java 10-nél járunk

Mi kell a Javához?

- Futtatáshoz:
 - JRE (Java Runtime Environment)
- Fejlesztéshez:
 - JDK (Java Development Kit)
 - JRE + Tools (pl. compiler)
 - Szövegszerkesztő
 - Jó ha van: IDE

Fejlesztés menete



Java program fordítása és futtatása

- Fordítás:

```
javac Osztalynev.java
```

Csak akkor fordítható, ha a fájl és a benne található osztály neve megegyezik!
- Futtatás:

```
java Osztalynev [parameterek]
```

Csak akkor futtatható, ha van benne main metódus!

```
public static void main(String[] args)
```

Build

Projekt: sok egymással együttműködő
class fájl → kellene 1 futtatható
állomány → build

Build eszközök: Ant, Maven, Gradle

Build folyamat

- Forrás állományok fordítása
- Erőforrás állományok kezelése
- Teszt esetek futtatása
- Alkalmazás összecsomagolása (.jar)

Java nyelv alapjai IDE használata

Maven

- pom.xml: projektleíró állomány
- Mappaszerkezet:
 - src/main/java
 - src/main/resources
 - src/test/java
 - src/test/resources
 - target/classes

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.training360.yellowroad</groupId>
  <artifactId>intramaven</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>${project.artifactId}</name>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.8</maven.compiler.source>
    <maven.compiler.target>1.8</maven.compiler.target>
  </properties>
</project>
```

Java nyelv alapjai

- Csomag
- Osztály
 - attribútum
 - konstruktor
 - metódus
- Megjegyzések

Java nyelv alapjai II.

- Karakterkészlete: Unicode
- Kis- és nagybetű érzékeny
- CamelCase használata az azonosítókban
- Kódolási konvenciók:
<http://www.oracle.com/technetwork/java/codeconvtoc-136057.html>

Változó, konstans, literál

- Minden adatot változóban tárolunk
- Lehet osztályban deklarált: attribútum (mező, osztály szintű változó)
`public int number;`
- Lokális: metóduson vagy blokkon belül
`String name;`
- Nincs konstans, de van `final` módosító
- Literálok: `1`, `"Szöveg"`, `'c'`, `true`, `null`

Adattípusok I.

- Primitív típusok
 - `byte`
 - `short`
 - `int`
 - `long`
 - `char`
 - `boolean`
 - `float`
 - `double`

Adattípusok II.

- Összetett típusok (osztály!)
 - tömb
 - String
 - Java API kollekciók
 - Java API osztályok

Kifejezések és operátorok

- Kifejezés: 1 érték kiszámítására operandusok és operátorok sorozata
- Operátorok:
 - + , - , * , / , %
 - new
 - < , > , <= , >= , == , !=
 - ! , && , || , & , |
 - ++ , --
 - ? :
 - =

Vezérlési szerkezetek

- Szekvencia
- Szelekció
- Iteráció

Szelekció (elágazás) I

```
if(feltétel) {  
    utasítás;  
}  
  
if(feltétel) {  
    utasítás1;  
} else {  
    utasítás2;  
}
```

Szelekció (elágazás) II

```
if(feltétel1) {  
    utasítás1;  
} else if (feltétel2){  
    utasítás2;  
} ...  
} else {  
    utasításn;  
}
```

Szelekció (elágazás) III

```
switch(kifejezés) {  
    case érték1:  
        utasítás1; break;  
    case érték2:  
        utasítás2; break;  
    ...  
    default:  
        utasításn;  
}
```

Iteráció (ciklus) I

```
for(init; feltétel; léptetés) {  
    utasítás;  
}  
  
while(feltétel) {  
    utasítás;  
}
```

Iteráció (ciklus) II

```
do {  
    utasítás;  
} while(feltétel);
```

break – a ciklust követő utasításra ugrik (kivétel a címke használata)

continue – a következő végrehajtási ciklusba ugrik

Típuskonverzió

- Implicit (automatikus)
byte age = 5;
int age2 = age;
- Explicit
 - Csomagoló osztályok:
parseXXX(String)
 - Osztályokban: toString()
 - Típuskényszerítés (cast)

Tömb I

- Több ugyanolyan típusú adat sorozata, amelyben minden elem elérhető közvetlenül a sorszámaival (index)
- Index 0-tól kezdődik
- Deklaráció: `int[] numberArray;`
- Létrehozás: `numberArray = new int[10];`

Tömb II

- Literálja: {elem1, elem2, elem3, ...}
- Hossza: `length`
- Arrays osztály:
 - `toString()`
 - `sort()`
 - `copyOf()`

Gyakori algoritmusok

Sorozatszámítás I

```
eredmény := kezdőérték  
ciklus amíg van elem  
    elem := következő elem  
    eredmény := f(eredmény, elem)  
ciklus vége
```

Sorozatszámítás II

Például összegzés Javában:

```
sum = 0;  
for(Element element: array) {  
    sum = sum + element;  
}
```

Megszámlálás I

```
db := 0  
ciklus amíg van elem  
    ha megfelelő(elem), akkor db := db + 1  
ciklus vége
```

Megszámlálás II

Példa megszámlálásra Javában:

```
count = 0;
for(Element element: array) {
    if(searched(element) {
        count++;
    }
}
```

Keresés I

eredmény := üres

ciklus amíg van elem és nem
megfelelő(elem)

elem := következő elem

ciklus vége

ha van elem, akkor eredmény := elem

Az első megfelelőt találja meg, ha van.

Keresés II

```
found = null; //???
```

```
int i = 0;
```

```
while(i < array.length &&  
!searched(array[i])) {
```

```
    i++;
```

```
}
```

```
if(i < array.length){
```

```
    found = array[i];
```

```
}
```

Keresés III

Hatékonyabban:

```
found = null;  
for(Element element: array) {  
    if(searched(element)) {  
        found = element;  
        break;  
    }  
}
```

Szélsőérték kiválasztás I

max := első elem

min := első elem

ciklus amíg van elem

 elem := következő elem

 ha elem > max, akkor max := elem

 ha elem < min, akkor min := elem

ciklus vége

Szélsőérték kiválasztás II

Előfeltétel: a sorozat nem üres, és az
elemeire létezik rendezési reláció

max = array[0]; min = array[0];

```
for (Element element: array) {
```

```
    if(element > max)
```

```
        max = element;
```

```
    if(element < min)
```

```
        min = element;
```

```
}
```

Kiválogatás I

eredmény = üres sorozat
ciklus amíg van elem
 elem := következő elem
 ha(megfelelő(elem)), akkor
 eredménybe(elem)
ciklus vége

Kiválogatás II

```
result = new ArrayList<>();  
for(Element element: array) {  
    if(searched(element)) {  
        result.add(element);  
    }  
}
```

Rendezések I

Egyszerű cserés rendezés
N := sorozat hossza
ciklus i := 0-tól N-2-ig
 ciklus j := i+1-től N-1-ig
 ha sorozat(i) > sorozat(j), akkor
 csere(sorozat(i), sorozat(j))
 ciklus vége
ciklus vége

Rendezések II

Buborékos rendezés

$N := \text{sorozat hossza}$

ciklus $i := N-1$ -től 1 -ig

 ciklus $j := 0$ -tól $i-1$ -ig

 ha $\text{sorozat}(j) > \text{sorozat}(j+1)$

 csere($\text{sorozat}(j)$, $\text{sorozat}(j+1)$)

 ciklus vége

ciklus vége

Rendezések III

Java-ban:

`Arrays.sort(array)`

`Collections.sort(collection)`

Mindkettőhöz definiálhatunk rendezési szempontot
vagy az elemek természetes rendezettségűek
lesznek.

Helyben rendez.


Osztály, objektum, interfész

Osztályok

- Az osztály a világ egyedeinek modellje
- Modell = tulajdonságok + viselkedés
- Az egyed állapota a leíró tulajdonságainak összessége
- Java nyelvben a legkisebb fordítási egység

Java osztály

```
public class Car {  
    private String registrationNumber;  
    private int speed;  
    private int positionX;  
    private int positionY;  
  
    public void accelerate() {  
        speed++;  
    }  
    public void move(int deltaX, int deltaY) {  
        positionX += deltaX;  
        positionY += deltaY;  
    }  
}
```



Láthatóság módosítók

- Osztályra
 - public
 - (package private)
- Tagra (attribútum, metódus)
 - public
 - protected
 - (package private)
 - private

Objektum

- Osztály: terv – hogyan néz ki általában egy egyed
- Objektum, példány: a terv egy megvalósítása, konkrét állapota van
- Objektum létrehozása: konstruktor
Feladata a példány legyártása és a tulajdonságainak inicializálása.

```
public class Car {  
    private String registrationNumber;  
    private int speed;  
    private int positionX;  
    private int positionY;  
  
    public Car(String registrationNumber) {  
        this.registrationNumber = registrationNumber;  
    }  
}
```

Default konstruktor

- Konstruktor mindig van!
- Ha mi nem készítünk → paraméter nélküli ún. default konstruktor jön létre.
- Láthatósága megegyezik az osztályéval

Metódusok

láthatóság
visszatérési típus
formális paraméterek
kivételek

```
public final void nap(int minutes) throws InterruptedException {
    //take a nap
}
```

egyéb módosító
azonosító
törzs

- módosítók: final, abstract, static
- paraméterek kiértékelése
 - sorrendi kötés
 - számbeli egyeztetés (varargs)
 - típus egyeztetés

TRAINING 360
GET FLOW NOT SKILLS
Osztály, objektum, interfész
52

Metódusok

- Visszatérési típus
 - void – nincs visszatérési értéke
 - return – a visszaadott érték megadása, azonnal kilép a metódusból
- Csak érték szerinti paraméterátadás van
- Speciális metódusok: getter/setter

TRAINING 360
GET FLOW NOT SKILLS
Osztály, objektum, interfész
53

Metódus túlterhelés

- Ugyanolyan funkciójú metódus ugyanolyan névvel, de különböző paraméterekkel
- A visszatérési érték is eltérhet
- Gyakori konstruktoroknál
- Gyakori általános célú metódusoknál

TRAINING 360
GET FLOW NOT SKILLS
Osztály, objektum, interfész
54

Destruktor

- Garbage Collector
- objektum megsemmisítése előtt fut le a finalize metódus, csak nem tudjuk, mikor lesz megsemmisítve ;)

OOP alapelvek I

- A-PIE
 - Abstraction
Csak a legszükségesebb részlet látszódik kifelé. (láthatóság)
 - Polymorphism
Az objektum többféle formában megjelenhet (method overriding, method overloading)

OOP alapelvek II

- Inheritance
Új osztály írása egy már meglévő kiterjesztésével.
- Encapsulation
Az adatok és a rajtuk dolgozó metódusok egységbe zárása, az adatrejtés elvével.

OOP alapelvek III

- S.O.L.I.D
 - Single responsibility principle
Egy osztálynak csak egy feladata lehet.
 - Open-closed principle
Nyitott a kiterjesztésre, de zárt a módosításra.
 - Liskov substitution principle
Helyettesíthetőség elve (leszármazott átveheti az ős szerepét).

OOP alapelvek IV

- Interface segregation principle
Egy osztály sose kényszerüljön olyan interfész vagy metódus implementálására, amire nincs szüksége.
- Dependency Inversion Principle
Az osztályoknak az absztrakciótól és nem a megvalósítástól kell függniük. (loose coupling) (interfészre programozunk)

Statikus tagok

- Attribútum: az osztályban tárolva, minden objektum ugyanazon a változón dolgozik.
- Metódus: nem az objektum attribútumain dolgozik, működése nem függ az objektum állapotától.

Nagy rendszerek

- Sok objektum együttműködése
- Osztályok csomagokba szervezhetők
- Osztályok közötti kapcsolatok
 - Öröklődés (is-a)
 - Tartalmazási (has-a)
 - Kompozíció
 - Aggregáció

Referenciák

- Cay S. Horstmann, Gary Cornell:
Core Java Vol. I. Fundamentals
- <https://docs.oracle.com/javase/tutorial/getStarted/index.html>
- <https://docs.oracle.com/javase/tutorial/java/nutsandbolts/index.html>
