Hello World

Óbudai Egyetem, Java Standard Edition
Mérnök Informatikus szak, BSc
Labor 1

Bedők Dávid
2016.10.07.
v0.3

Információ

Hivatalos tárgyi weboldal: http://users.nik.uni-obuda.hu/java/

- Prezentációk
- Minta zárthelyik
- Követelményrendszer

Tárgyfelelős: **Szénási Sándor** (szenasi.sandor@nik.uni-obuda.hu)

Labor weboldal: http://users.nik.uni-obuda.hu/bedok.david/jse.html

- Labor prezentációk (vmajor.minor + yyyy.mm.dd.)
- Programok, eszközök, library-k
- Forráskódok

Oktató: **Bedők Dávid** (bedok.david@nik.uni-obuda.hu)

Előkövetelmény:

- **Programozás I/II (C#) alapos** ismeret
- **XML**, **XHTML** ismeret
- Open-source szemlélel
- Java ecosystemre való nyitottság

Követelmény

- Laborlátogatások TVSZ szerint
- 2 zárthelyi dolgozat
  - I. Objektum-orientált programozás Java nyelven
  - II. Egyszerű webalkalmazás fejlesztése Java nyelven
- Pót zárthelyi dolgozat (várhatóan összevont)
- Gyakorlati jegy pótlása a vizsgaidőszak első 10 napjában

Tematika

Elsősorban: Hivatalos oldalon megtalálható követelményrendszer szerint.

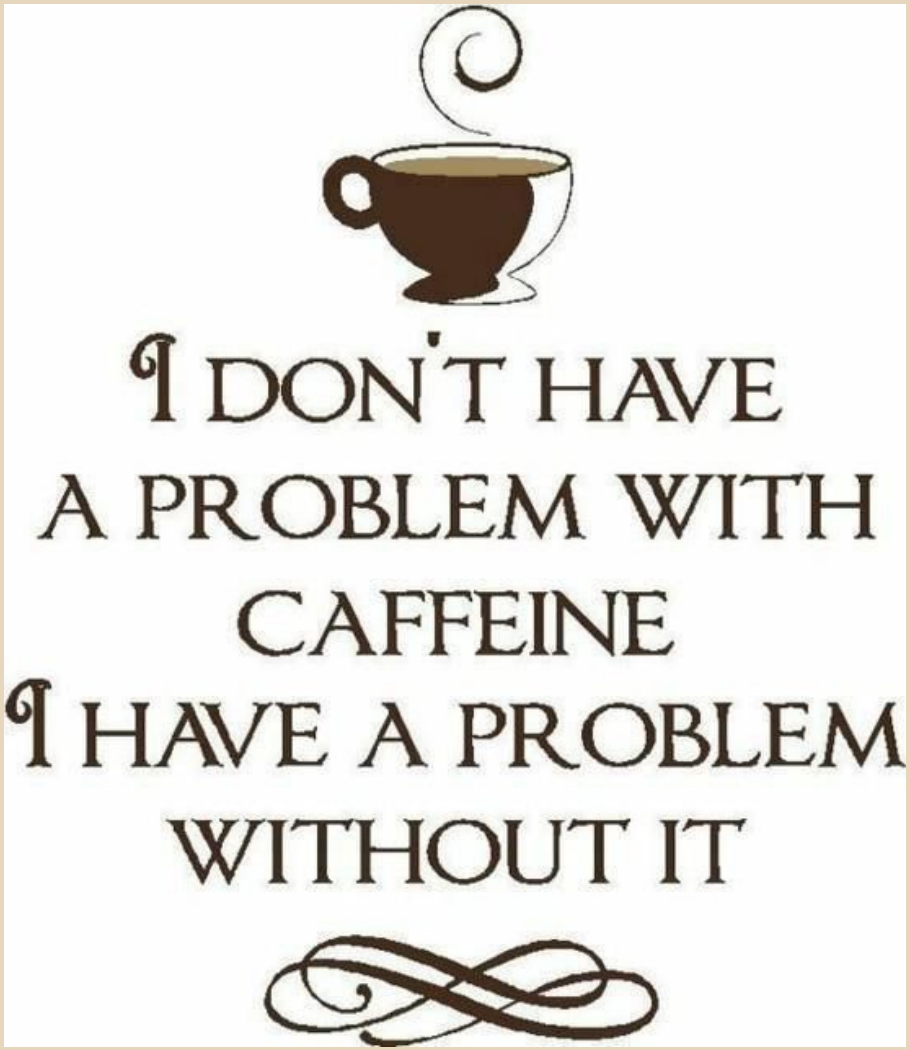
Laborfoglalkozás hangsúlya: A Java nyelv alapvető megismerése, a C# nyelven tanult objektum-orientált ismeretek átültetése (**Java SE 7**).

A Webfejlesztés alapjaival való megismerkedés és a “framework nélküli” Java webalkalmazás fejlesztés elsajátítása (**Servlets**, **JSP**, esetleg **JSTL**, **TLD**, stb.).

Minimális fejlesztői környezet építési tapasztalat megszerzése (**Eclipse IDE**, **Apache Tomcat**).

A tárgy fontos előkövetelménye a “Szerveroldali Java Programozás” címen futó Java Enterprise Edition laborgyakorlatnak, mely egy szakmai választható tárgy.

Java szó eredete



Java Standard Edition - Történet

1991. **Oak**, később **Green**
- *SUN* (Stanford University Network)
- **Dr. James A. Gosling**
- Mike Sheridan
- Patrick Naughton

1996.01.23 Java 1.0 [AWT]
1997.02.19. Java 1.1 [Inner class, JDBC, RMI, Reflection API]
1998.12.08. Java 1.2 **Playground**
2000.05.08. Java 1.3 **Kestrel** [Java Sound, JNDI API]
2002.02.06. Java 1.4 **Merlin** [regexp, exception chain, Image IO, Pref. API]
2004.09.30. Java 5 **Tiger** [autoboxing, generic types]
2006.12.11. Java 6 **Mustang**
2007. GPL, nyílt forráskódú szabad software lett
2009. *Oracle* felvásárlás
2011.07.28. Java 7 **Dolphin**
2014.03.18. Java 8 **Spider** [lambda expression]
2017.03.?? Java 9 (money, currency API, better nativ code integration, ..)
2018.??.?? Java 10 (primitív típusok kivétele?)

Java vs. JavaScript

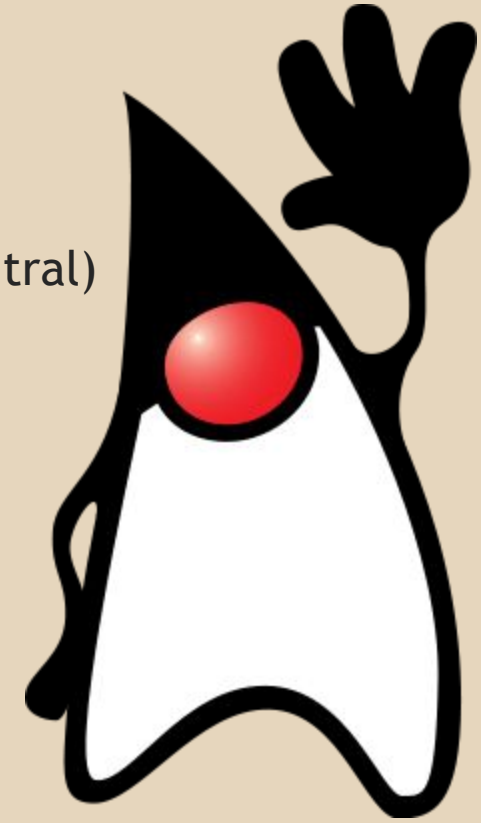
"Not to be confused with JavaScript." (wikipedia)
"Java and JavaScript are Still Two Different Animals" (htmlgoodies)

Kis túlzással a JavaScript és a Java közös tulajdonságaik kimerülnek a nevükben található 4 ASCII karakterben...

Koncepció

Write **O**nce **R**un **A**nywhere ("írd meg egyszer, futtasd bárhol")
- WODA "írd meg egyszer, keress hibát mindenhol"

Koncepció
- egyszerű, objektum-orientált és ismerős (szintaxis)
- robosztus és *biztonságos*
- concurrent computing (parallel execution, szálkezelés)
- platformfüggetlenség (*hordozhatóság*, architecture-neutral)
- hálózati programozás kiemelt támogatása
- biztonságos távoli futtatás (applet korszak)
- *kiemelkedő teljesítmény*



Főbb tulajdonságok
- általános célú
- objektum-orienált
- interpretált (futás közben fordul)
- ~ C/C++ szintaxis

Változatok

- **Java Card**
  - smartcard-ok számára
- **Java Platform Micro Edition** (Java ME, korábban* J2ME)
  - csökkentett erőforrások, mobil eszközök számára
- **Java Platform Standard Edition** (Java SE, korábban* J2SE)
  - workstation-ök számára
  - általános felhasználás, kliens gépek
- **Java Platform Enterprise Edition** (Java EE, korábban* J2EE)
  - elosztott vállalati környezetben, avagy széles(ebb) spektrum igény esetén

*: *2006 előtt más néven voltak hivatkozva, de ez zavaró volt*

Java Virtual Machine (JVM)

- JVM részei
  - osztálybetöltő (class loader)
  - szemétgyüjtő (garbage collector)
  - végrehajtó motor (execution engine)
- **HotSpot**: Oracle-Sun felvásárlást követően az Oracle által karbantartott JVM neve
  - Java 1.2-ban még modul, Java 1.3 óta ez az alapértelmezett JVM
  - C++/assembly-ban írták, és valójában két virtuális gépet takar (különböző célú optimalizálás):
    - **Client** (gyors fordítási sebesség, rövid életű kód)
    - **Server** (erősen optimalizált, hosszú életű kód)
  - Kapcsolók segítségével konfigurálható
  - Windows (x86/64), UNIX (x86/64) és Solaris (SPARC is) támogatás
- Vannak más JVM implementációk is: Kaffe, IBM J9 (ezek ún. *clean room* implementációk), vagy a BEA/Oracle JRockit (pl. WebLogic AS használta, ma Oracle Fusion Middleware)

SDK, JRE, JDK, TLA...

Java Development Kit (JDK)

- korábban hívták Software Development Kit (SDK)-nek is
- Java programok fejlesztéséhez, hibakereséséhez
  - Verziózás: JDK1.7, JDK1.8, stb. (+build number)
  - javac.exe, java compiler
  - java.exe, policytool.exe, jar.exe, jarsigner.exe, javadoc.exe, jconsole.exe, keytool.exe, wsgen.exe, wsimport.exe, stb.

Java Runtime Environment (JRE)

- A JDK egy részhalmaza
- Java programok futtatásához használható
  - Verziózás: JRE7, JRE8, stb.
  - java.exe, java runtime luncher
  - policytool.exe, keytool.exe, stb.

Javasolt további Java alap komponens

- *Java Cryptography Extension* (JCE) Unlimited Strength Jurisdiction Policy Files
- US security policy miatt nem lehet része a JDK-nak

Natív, byte és Java kód

- **Natív kód**
  - Win OS:: ~ *.exe
  - architektúra, platform függő
- **(Java) byte-kód** (bytecode)
  - *.class
  - platformfüggetlen
  - JVM értelmezi, natív kódot készít belőle
- **Java kód**
  - *.java
  - legtöbbször hordozható, platformfüggetlen
  - fordítást követően Java byte-kód készül belőle
- Nem kizárólag Java nyelven készülnek Java byte-kódok, vagyis JVM platformra más nyelven is lehet fejleszteni
  - Scala
  - JRuby
  - Groovy script
  - JavaFX script
  - AspectJ
  - stb.

Hello World

HelloWorld.java

```
package hu.qwaevisz.demo;

public class HelloWorld {

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

}
```

Az állományt kötelezően **HelloWorld.java** néven (case sensitive) kell elmenteni a **package**-nek megfelelő könyvtár struktúrában:

- hu
  - qwaevisz
    - demo
      - HelloWorld.java

Minden állományban egyetlen nyilvános osztály (**public class**) lehet.
A "program" belépési pontja a **main** metódus, ennek első utasításával indul (számunkra) a végrehajtás.
A **System.out.println()** feloldása: *System* osztály, *out* ezen osztály statikus *PrintStream* típusú nyilvános mezője, a void *println()* metódus pedig értelemszerűen ezen *PrintStream* osztály példánymetódusa.

Fordítás és futtatás

```
>mkdir bin
>javac -d ./bin ./src/hu/qwaevisz/demo/HelloWorld.java
```

-d [path]: output directory
minden ami a paraméterek mögött van: forrásállományok elérési útja (tetszőleges számú lista, az elérési útban forráskönyvtárak (src) és csomagkönyvtárak is lehetnek (hu, qwaevisz, demo), de értelemszerűen csak ebben a sorrendben).

```
>java -cp ./bin hu.qwaevisz.demo.HelloWorld
```

-cp: classpath (ahol a class állományok vannak)
utolsó paraméter: belépési osztály full qualified neve

Teljesítmény

- Általában a Java-ban írt programok lassabbak és több memóriát igényelnek mint a C-ben írt változatuk.
- A Java 1.1-ben bemutatták a JIT (Just-in-time) compilation-t (futásidejű fordítás*), amivel elindult egy közelítési folyamat
- 2012. februári adat szerint csupán ~1.5x lassabb a Java a C-nél
- szerver környezetben ez a sebesség eltérés általában lényegtelen

*: a byte-kódot futási időben fordítja natív kóddá, dynamic compilation-nek is nevezik

Kritikák

- generikus típusok megvalósítása (ez nézőpont kérdése)
- Java sebessége (nézőpont kérdése)
- előjel nélküli primitív típusok hiánya
- lebegőpontos aritmetikai számítások kezelése
- HotSpot korai változatainak biztonsága, támadhatósága
  - pl. Double converting bug (karakterláncból valós szám konverziója végtelen ciklust okozott) JDK 1.6.10-ben még benne van.. (1.6.27-ben már nincs)

Típusok

- **Primitív típusok** (pontos méretük előre definiált. Pl. C++ban nem így van, a nagyobb optimalitás miatt különböző platformokon különböző méretű pl. az egész típus) (C# value types..)
- **Összetett típusok** (osztályok, tömbök, minden más)

típus	leírás
byte	8 bites előjeles egész
short	16 bites előjeles egész
int	32 bites előjeles egész
long	64 bites előjeles egész
float	32 bites egyszeres lebegőpontosságú
double	64 bites kétszeres lebegőpontosságú
char	16 bites Unicode-karakter
boolean	logikai érték (igaz / hamis)

Literálok

"A literal is a source code representation of a value." (MSDN Library)

Integer literal

42 (int)	42L (long)
0x4a (hexadicima int)	042L (octal long)

Character literal

'c'	'\n' (new line)
'\t' (tab)	'\\' (backslash)

'\u**d**' ahol d hexadec. 4 jegyű szám (unicode karakter) pl.: '\u0041' (A)

Boolean literal

true	false

Floating-point literal

3.14f (float)

3.14d (double) - d nélkül is double (alapértelmezett)

.123 (double) (0.123)

10. (double) (ha csak 10-et írunk, akkor int)

String literal (speciális, mivel nem primitív típus)

"Hello World" "c"

Null literal (minden szempontból speciális)

null

Tömbök

```
int[] data;
data = new int[3];
data[0] = 42;
data[1] = 30;
data[2] = 0;
```

Tömb literál (tömb inicializáció, speciális, mivel nem primitív típus)
`{}` (üres tömb)
`{1, 2, 3}` (3 elemű int tömb)
`{'a', 'b', 'c', 'd'}` (4 elemű char tömb)

```
int[] data = { 3, 2, 5, 7, 9, 2, 10 };

for (int i = 0; i < data.length; i++) {
    System.out.println(data[i]);
}
```

C# vs. Java: alapvető programozási tételek átírhatóak C#-ról Java nyelvre "legtöbbször" a `Length length`-re átírásával (vagyis az L-et kell kisbetűsíteni). Azonban: Java-ban a **`length`** egy nyilvános (történelmi okokból) és végleges (itt `final`, C# `readonly`) mező, addig C#-ban a **`Length`** egy nyilvános `property`. Java-ban nincsenek ilyen jellegű `property`-k.