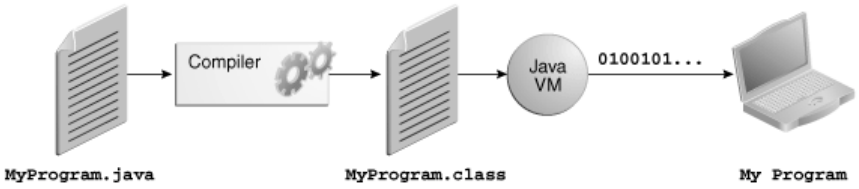
Programozás III

JAVA ALAPOK

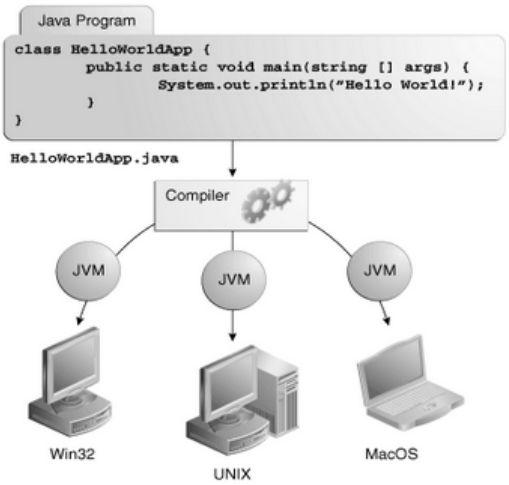
A JAVA TECHNOLÓGIA LÉNYEGE

Többlépcsős fordítás



A JAVA TECHNOLÓGIA LÉNYEGE

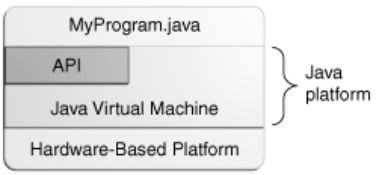
Platformfüggetlenség



JAVA PLATFORM

Két komponense:

- Java Virtual Machine (JVM)
- Java Application Programming Interface (API)



Kicsit lassúbb, mint a natív kód futtatása.

KITÉRŐ: JAVA vs C++ vs C#

http://www.developer.com/java/article.php/3856906/Java-vs-C-The-Performance-Showdown.htm

http://www.25hoursaday.com/CsharpVsJava.html

http://www.harding.edu/fmccown/java_csharp_comparison.html

http://slashdot.org/topic/cloud/java-vs-c-which-performs-better-in-the-real-world/

http://shootout.alioth.debian.org/

http://stackoverflow.com/questions/1049004/java-vs-c-are-there-any-studies-that-compare-their-execution-speed

Google

A JAVA TECHNOLÓGIA LÉNYEGE

Néhány tutoriál:

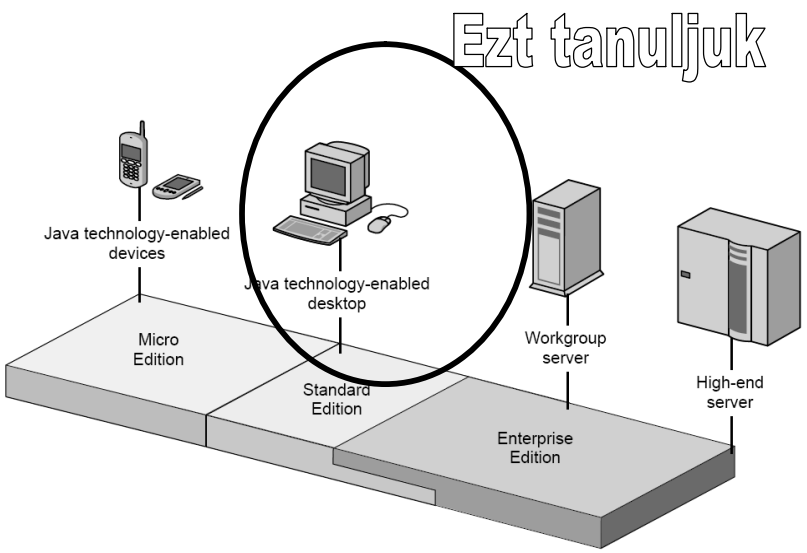
http://docs.oracle.com/javase/tutorial/

http://www.oracle.com/technetwork/java/langenv-140151.html

http://www.java2s.com/Tutorial/Java/CatalogJava.htm

+ Google és Java fórumok.

JAVA PLATFORMOK



KÖRNYEZETEK

Java Futtató Környezet

JRE: Java Runtime Environment

(tartalmazza a JVM-et (java parancs))

Ezt töltjük le a gépünkre otthon, és ez fut a mobiltelefonon is

KÖRNYEZETEK

Java Fejlesztő Környezet

JDK: Java Development Kit
ez fejlesztő eszköz, de tartalmazza a JRE-t is!
(Ebben található a javac parancs)

Verzió: 1.8, 144-es update.

(Java SDK (Software Development Kit))

FEJLESZTŐESZKÖZÖK

A JDK fapados módszert biztosít (notepad + parancsmód)

Ezért IDE-ket használunk

Integrated Development Kit
Fordítók, grafikus szövegszerkesztők,
syntax highlighting…stb

Javasolt fejlesztőeszközök:

NetBeans

Eclipse

Egyéb…

A JAVA TELEPÍTÉSE

1.lépés
Szoftverek letöltése

2.lépés
JDK telepítése (ez a JRE-t is telepíti)

3.lépés
IDE telepítése: Eclipse, NetBeans, stb

(A NetBeans + JDK egyben is letölthető, telepíthető)

OTTHONI GÉPEN

Figyelni kell a környezeti változók beállításaira:

PATH → JDK bin könyvtára

CLASSPATH → az a könyvtár, ahol dolgozunk
(„.” az aktuális könyvtár)

(Ez csak a „fapados” változatnál érdekes, a fejlesztőkörnyezetek helyes telepítés esetén tudják kezelni a környezeti változókat.)

ELSŐ JAVA PROGRAM

```
/*
 * Elso program
 */

public class Hajra{
   public static void main(String args[]){
       System.out.println("Jó tanulást kívánok!");
   }
}
```

Fájlnév: Hajra.java

FÁJLNÉV = OSZTÁLYNÉV!!!

FORDÍTÁS ÉS FUTTATÁS

Fordítás:

javac Hajra.java

⇒ Hajra.class (bájtkód)

Futtatás:

java Hajra (a bájtkódot futtatjuk)

Integrált fejlesztőkörnyezetben ezek egy gombnyomással elintézhetők ☺

ELSŐ (PRIMITÍV) PROGRAMJAINK VÁZLATA

```
public class OsztályNév
{
    public static void main(String args[])
    {
        //Főprogram törzse
    }
}
```

Ez még nem objektumorientált gondolkozás, hiszen egyetlen osztály alkotja a teljes programot.

Az utasítások csak metóduson belül lehetnek – egyszerűbb feladatoknál csak a main metódus létezik, ezen belül vannak az utasítások.

Az osztályon belül több metódus is lehet.

TÍPUSOK, VÁLTOZÓK, DEKLARÁLÁSOK

Erősen típusos nyelv:
Deklarálni kell a változókat a használat előtt

Típusok
int, float, char, double, boolean ← *egyszerű*
és **S**tring típus (fontos a nagy „S”) ← *referencia*

Egyszerű típus: azonosítójával közvetlenül hivatkozunk a változó memóriahelyére. Ezt a helyet a rendszer a deklaráló utasítás végrehajtásakor foglalja le.
Referencia típus: A referencia típusú változók objektumokra mutatnak. Egy referencia típusú változó azonosítójával közvetve hivatkozunk az objektum memóriahelyére. (Maga a hivatkozás rejtve marad.) Deklaráláskor csak a referencia részére foglalunk tárterületet, maga az objektum a példányosítás során jön létre.

KIFEJEZÉSEK, MŰVELETEK

A SZOKÁSOS ☺

Aritmetikai kifejezések (+, -, *, /, %)

Inkrementálás, dekrementálás (prefix és postfix alakok egyaránt)

Összehasonlító operátorok (==, <, >, !=…stb)

Logikai operátotok (ÉS: && VAGY: || NEM: !)

Stb…

PROGRAMSZERVEZŐ UTASÍTÁSOK

Pontosan úgy, mint C-ben

Szelekciók (if…else, switch…case…)

Iterációk

- Elöltesztelő ciklus (while)
- Hátultesztelő ciklus (do…while)
- Növekményes ciklus (for)

Egyebek

- break, continue

Utasítás végén pontosvessző!

METÓDUSOK

Javaban nem használatos kifejezés:
- eljárás
- függvény

helyette **METÓDUS**

Metódusok általános alakja:

visszatérési_érték *metódusnév* *(paraméterlista*) {

// törzs

}

PARAMÉTEREK HASZNÁLATA

A main metódus
- String tömbje → **args[]**

A konzolról Stringeket vesz át!!!

- args[0] → első paraméter
- …
- args[n] → n. paraméter

Például:

java ElsoProgram egy 2 3.7

args[0]=egy ; args[1]=2 ; args[2]=3.7

FONTOS!!!

Fájl neve == Osztály neve

A fájl nevének pontosan meg kell egyeznie a főosztály (publikus osztály) nevével!!!

(case sensitive módon!!)

Még egy: mit jelent a main metódus fejében a static kulcsszó?

public **static** void main

PROGRAMOZÁSI ALAPOK

Programírással kapcsolatos néhány alapfogalom:

- **Memóriakezelés**
  - A változóhoz a memóriaterület hozzárendelése (*allokálás*-a)
    - automatikus, a definíció kiértékelésekor
    - a programozó rendelkezik róla
  - a változó által lefoglalt terület felszabadítása
    - automatikus,
    - a programozó hatáskörébe tartozik.
- **Élettartam**

  A változókhoz a szükséges memóriaterület lefoglalása (allokálása) és annak felszabadítása közötti időt a változó *élettartam*ának nevezzük.

PROGRAMOZÁSI ALAPOK

- **statikus változó:**
  - élettartama a program egész működése idejére kiterjed
  - mindig az ún. statikus (main) memória-területen helyezkedik el
  - a statikus terület egyszer, a program betöltésekor kerül lefoglalásra
- **dinamikus változó**:
  - a program explicit módon foglal le területet, a dinamikus (heap) memória-területen, amire a címével, ún. pointer-rel lehet hivatkozni.
  - egyes nyelvek tartalmaznak utasítást a dinamikus területek felszabadítására is.

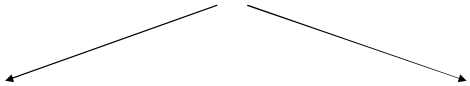
ÚJABB PÉLDA

```
/* faktoriális számítás */
public class Faktorialis {

    public static void main(String[] args) {
        double fakt = 1;
        int n = 10;
        System.out.println("A faktoriálisok:");
        for (int i = 1; i <= n; i++) {
            fakt *= i;
            System.out.println(i + "! = " + fakt);
        }
    }
}
```

Mi hiányzik? beolvasás

BEOLVASÁS BILLENTYŰZETRŐL



Biztonságos,
de kicsit bonyolult

Bufferelt beolvasás
kötelező kivételkezeléssel

Egyszerű,
de kevésbé biztonságos

Scanner osztály, nem
kötelező a kivételkezelés

SCANNER OSZTÁLY

Használata:

Példányosítani kell:

```
Scanner scan = new Scanner(System.in);
```

Ehhez importálni kell a java.util.Scanner osztályt:

```
import java.util.Scanner;
```

(automatikusan generálható)

A scan példány metódusai segítségével olvashatunk.

SCANNER OSZTÁLY

Metódusok:

scan.next() (vagy nextLine()) ← String beolvasása

scan.nextInt() ← integer beolvasása

scan.nextFloat() ← float beolvasása

stb.

A PÉLDA JAVÍTOTT VÁLTOZATA

```
/* faktoriális számítás */
public class Faktorialis {

    public static void main(String[] args) {
        double fakt = 1;
        int n;
        Scanner scan = new Scanner(System.in);

        System.out.print("Meddig számoljak: ");
        n = scan.nextInt();
        System.out.println("A faktoriálisok:");
        for (int i = 1; i <= n; i++) {
            fakt *= i;
            System.out.println(i + "! = " + fakt);
        }
    }
}
```

APROPÓ: REKURZIÓVAL IS MEGOLDHATÓ ☺

```
public class RekurzivFakt{

    public static void main(String args[]){
        System.out.print("mekkora szám faktoriálisát számojam? :");
        int n;
        Scanner scan = new Scanner(System.in);

        System.out.print("Meddig számoljak: ");
        n = scan.nextInt();
        System.out.printf("%d faktoriálisa: %15.0f", szam, fakt(szam));
    }

    static double fakt(int n){
        if (n<=1)return 1;
        else return n*fakt(n-1);
    }
}
```

TÖMBÖK

Igazi referencia típusok

Indexelés 0-tól

Szintaktika:

Deklarálás: **int n = 10;**
int tomb[ ]=new int[n];

Hivatkozás: **elem=tomb[i];**

Hossz: tomb.length;

(A deklarált méretet jelenti, nem a feltöltött elemek számát!
A tömb létrehozásától (deklarálásától) kezdve állandó.)

MÁTRIXOK

Nincs kétdimenziós tömb, csak tömbök tömbje.

Pl.:

```
int sor = 3;
int oszlop = 5;
float matrix [ ][ ]=new float[sor][oszlop];
```

(utolsó elem: matrix[2][4])

NÉHÁNY EGYSZERŰ METÓDUS

A **Math osztály** sok gyakran használatos metódust tartalmaz. Például:

Négyzetgyök művelet **Math.sqrt(…)**

Használata:

```
int szam;
int gyoke = Math.sqrt(szam);
```

Véletlenszám generátor

A Math osztály random() metódusa egy 0 és 1 közé eső double értéket ad ($0 \leq$ veletlenSzam < 1)

Használata:

```
veletlenSzam = Math.random();
```

MEGJEGYZÉS

Véletlenszám generátor

az (alsó,felső) intervallumba eső véletlen egész szám:
(int)((felső-alsó)*Math.random()) + alsó + 1;
az [alsó, felső] intervallumba eső:
(int)((felső+1-alsó)*Math.random()) + alsó;

DE pl. a (0,10) intervallumba eső véletlen szám **NEM**
(10 - 0)*Math.random() + 0 + 1
HANEM
10*Math.random() + 1 !!!

NÉHÁNY EGYSZERŰ METÓDUS: KONVERZIÓK

String → integer konverzió
Integer (csomagoló) osztály parseInt metódusa…
pl.:

```
String szam=”5”;
int ertek=Integer.parseInt(szam);
```

Egyéb konverziók, pl.:

```
String szam = ”5”;
float fertek = Float.parseFloat(szam);
double dertek = Double.parseDouble(szam);
String vissza = Double.toString(dertek);
float d_to_f = (float) dertek;
```

MEGJEGYZÉS

Mindegyik típus esetén létezik egy valueOf metódus is:

```
String szam = ”5”;
int iertek = Integer.valueOf(szam);
float fertek = Float. valueOf(szam);
double dertek = Double. valueOf(szam);
String visszad = String.valueOf(dertek);
String visszai = String.valueOf(iertek);
```

A két módszer között van elvi eltérés, de gyakorlatilag egyformán használhatóak.

FORMÁZOTT KIIRATÁS

System.out.***printf(…)***

pl.:

```
System.out.printf("%d faktoriálisa: %15.0f", szam, fakt(szam));
```

További részletek: HELP

FONTOS!!!

HELP!

http://download.oracle.com/javase/7/docs/api/
vagy
http://download.oracle.com/javase/8/docs/api/

vagy: google ☺

HÁZI FELADAT

- Fejlesző környezet létrehozása az otthoni gépen
- C nyelv átismétlése
- Beadandó: ld. a witch-en, neptunban vagy fb-n.

TUDNIVALÓK

Bár a Java dokumentáció szerint a Java egyik tulajdonsága az egyszerűsége, mégis ez egy nehéz nyelv!

Folyamatos otthoni munka nélkül nem lehet megtanulni!

TUDNIVALÓK

A folyamatos otthoni munka hatása: