Java programozási nyelv 2007-2008/ősz
1. óra

A Java programozási nyelv

Bevezetés, alapfogalmak
Változók, egyszerű típusok, tömb
Vezérlési szerkezetek

Standard I/O

legradi.gabor@nik.bmf.hu szenasi.sandor@nik.bmf.hu

Java programozási nyelv

Témakörök

Bevezetés, a Java kialakulása

Alapfogalmak

Alapvető változótípusok

Tömbök

Vezérlési szerkezetek

Standard I/O

Bevezetés, a Java kialakulása

- Általános célú, objektum-orientált alapú programozási nyelv
- Kialakulása:
  - SUN (Stanford University Network) nevű cég,
  - 80-as évek vége,
  - első publikált változatot a 90-es évek közepén
  - több változáson ment keresztül a nyelv azóta,
  - C, illetve C++ alapokról kiindulva (bonyolultabb dolgok elhagyásával, egyszerűsítés alkalmazásával),
- Tervezési célok:
  - biztonság;
  - OOP támogatása;
  - átvihetőség (portabilitás) nagymértékű támogatása;
  - átgondolt objektumkönyvtár;

(Megjegyzés: Azért még a Java nyelv sem tökéletes!)

Java programozási nyelv

Témakörök

Bevezetés, a Java kialakulása

Alapfogalmak

Alapvető változótípusok

Tömbök

Vezérlési szerkezetek

Standard I/O

Fogalmak

- **JVM (Java Virtual Machine):**
  Olyan szoftverréteg („szoftveres processzor”), amely képes lefordított Java programokat, azaz bájtkódot végrehajtani.
- **JRE (Java Runtime Environment):**
  JVM + osztálykönyvtár
  Akkor érdemes használni, ha csak a Java környezet, azaz Java programok futtatása a cél (felhasználás).
- **JDK (Java Development Kit):**
  JVM + osztálykönyvtár + fordító eszközök
  Akkor érdemes használni, amikor a Java programok fordítása és futtatása a cél (fejlesztés + felhasználás)

Fogalmak (2.)

- **ME (Micro Edition):**
  Olyan Java változat, amely kisebb erőforrású gépeken (mobiltelefon, PDA) működő programok fejlesztését támogatja.
- **SE (Standard Edition):**
  Olyan Java változat, amely általános célú gépeken (PC) működő programok fejlesztését támogatja.

  A Java kiadások közül ez a változat felel meg az egyéb általános célú programozási környezeteknél eddig megismerteknek.
- **EE (Enterprise Edition):**
  Olyan Java változat, amely üzleti célú programok fejlesztését támogatja (pl. komponensek, szerveroldali Java támogatás, JSP, adatbázisok elérése).

Fogalmak (3.)

- **alkalmazás (application):**
  Olyan Java program, amelynek futtatásához a JRE szükséges.
- **applet („kisalkalmazás”):**
  Olyan Java program, amelyet jellemzően HTML nyelvű oldalakba ágyaznak, s futtatásukhoz a böngészőben levő JVM, vagy a JDK-ban található appletviewer program használható.
- **szervlet:**
  Olyan Java program, amelyet egy szerveren futó JVM hajt végre (pl. adatbázishoz kapcsolódik, s munkájának eredményét HTML/XML formában szolgáltatja). Szervletek futtatásához az Enterprise Editon környezet szükséges.

Fogalmak (4.)

- **natív kód:**
  Olyan programkód, amelyet a processzor módosítás vagy átalakítás nélkül is azonnal végre tud hajtani.
- **bájtkód (bytecode):**
  Olyan programkód, amelyet a processzoron futó alkalmazás (pl. maga JVM) módosítás vagy átalakítás nélkül is azonnal végre tud hajtani. A bájtkód tulajdonképpen a JVM natív kódjaként is felfogható.

  A Java programok (a bájtkódok) átvihetők több, akár különböző operációs rendszeren levő azonos verziójú JVM között is („write once, run anywhere”).

  A .NET keretrendszernél megismertekkel ellentétben tehát itt a bájtkódot közvetlenül futtatja egy „virtuális” processzor

Fogalmak (5.)

- **JVM felépítése:**
  - hardver (a legalsó szint):
    maga a működő számítógép,
  - operációs rendszer (középső szint):
    szoftver, amely a gép erőforrásait kezeli, s futtatja a programokat,
  - JVM (a legfelső szint):
    a bájtkódot ellenőrzi és futtatja, valamint az egyes programlépésekhez szükséges operációs rendszerbeli funkciókat meghívja,
- **Java nyelv – JVM kapcsolata:**

  A nyelv és a futtatási környezet egymástól függetlenek. Elvben más programozási nyelveken írt forrásokból is fordítható JVM bájtkód, illetve elképzelhető natív kódot készítő Java fordító is.

Fogalmak (6.)

- Java programok fordítása parancssorból

  ```
  javac javaprogram.java
  ```

- Java programok futtatása parancssorból

  ```
  java javaprogram
  ```

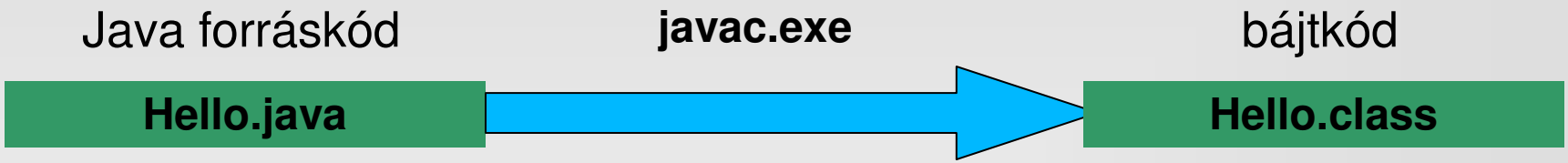
- A parancssori használathoz a PATH nevű rendszerváltozót be kell állítanunk!

  (Sajátgép Tulajdonságok -> Speciális -> Környezeti változók -> Felhasználói változók -> adjuk hozzá a PATH változóhoz a JDK által szolgáltatott java.exe és javac.exe programokat tartalmazó alkönyvtár útvonalát!)

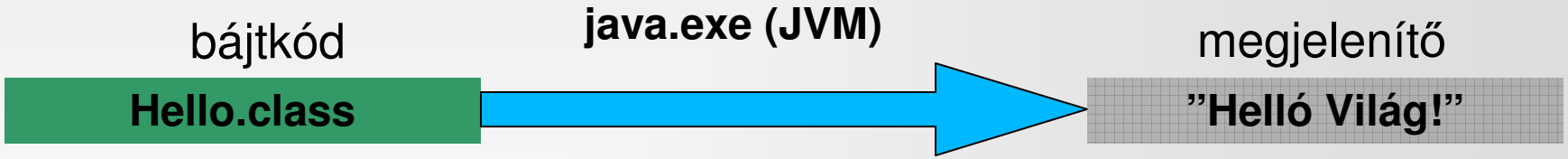
- Amennyiben fejlesztőkörnyezetet használunk (mint pl. NetBeans), akkor a fordítást és futtatást a környezet segítségével tudjuk elvégezni.

Fogalmak (7.)

A Java program fordításának folyamata:



A Java program futtatásának folyamata:



Feladat

1.1. feladat: Készítsük el azt a programot, amely kiírja a ”Helló világ!” szöveget a képernyőre!

1.1.a. feladat: A program készüljön a parancssori eszközök felhasználásával!

1.1.b. feladat: A program készüljön a fejlesztőeszköz felhasználásával!

(Kiegészítés: derítsük ki azt, hogy az elkészült programok pontosan milyen fájlokat és alkönyvtárakat jelentenek a két megoldás esetén!)

Feladat

1.1.a feladat: (Megoldás)

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

Megjegyzések:

- A publikus Hello osztályt kötelezően Hello.java fájlban kell elhelyezni
- main metódus kötelező szignatúrája:
  public statis void main(String[] args)
  (kisbetű, paraméterek, publikus)

Témakörök

- Bevezetés, a Java kialakulása
- Alapfogalmak
- Alapvető változótípusok
- Tömbök
- Vezérlési szerkezetek
- Standard I/O

Alapvető típusok a Java-ban

Primitív típus	Osztály	Leírás
boolean	Boolean	logikai típus
char	Character	16 bites UNICODE
byte	Integer	8 bites előjeles egész
short	Integer	16 bites előjeles egész
int	Integer	32 bites előjeles egész
long	Integer	64 bites előjeles egész
float	Float	32 bites lebegőpontos szám
double	Double	64 bites lebegőpontos szám

Változó létrehozása: `típusnév változónév;`

vagy `típusnév változónév = alapérték;`

Alapvető típusok a Java-ban (2.)

- Primitív típus
  A változóban a konkrét érték tárolódik. A Java nyelvben a változótípusok mind előjelesek!
- Osztály
  Az érték helyett egy adott típusú objektum referenciája jön létre, a változó csak a referenciát tárolja.
- Természetesen a más nyelvekben megszokott operátorok és műveleti jelek itt is használhatók.
- A == (egyenlőségvizsgálat) operátor objektumreferenciák esetén a referenciák egyenlőségét, nem pedig a referenciákkal jellemzett objektumok egyezését vizsgálja (arra az equals() metódus szolgál).

  Primitív típusok esetén tartalmat hasonlít össze.

Alapvető típusok a Java-ban (3.)

Művelet típusa	Műveleti jel
Posztfix	`kif++ kif--`
Unáris	`++kif --kif +kif -kif ~ !`
Multiplikatív	`* / %`
Additív	`+ -`
Shift	`<< >> >>>`
Relációs	`< > <= >= instanceof`
Egyenlőség vizsgálat	`== !=`
Bitenkénti ÉS	`&`
Bitkitenkénti kizáró VAGY	`^`
Bitenkénti megengedő VAGY	`|`
Logikai ÉS	`&&`
Logikai VAGY	`||`
Ternáris	`? :`
Értékadás	`= += -= *= /= %= &= ^= |= <<= >>= >>>=`

Alapvető típusok a Java-ban (4.)

- A Java nyelv nem rendelkezik primitív string típussal, helyette a *java.lang.String* osztály használatos. Ez nem azonos a karakterekből képezhető tömbbel, működése mindenben megegyezik az átlagos referencia típusokkal
- Mivel a nyelv nem ad lehetőséget operátor felülírásra (és a fordító sem kezeli a *String*et speciális esetként), különösen ügyelni kell az ilyen típusú objektumok összehasonlításánál, mivel gyakori hibaforrás:
  - *== operátor esetén*: referencia szerinti összehasonlítás (két különböző *String* objektum esetén, azok tartalmától függetlenül értéke mindig hamis lesz)
  - *equals(String) esetén*: két *String* objektum esetén azok tartalmát hasonlítja össze
- A *String* példányosítása a literál alapján automatikus:
  `"valami" → new String("valami")`

Alapvető típusok a Java-ban (5.)

- A *String* típusú objektumok nem változtathatók, így minden *String* típusra vonatkozó művelet egy új *String* típusú objektumot hoz létre (ennek elkerülése érdekében lásd *StringBuffer* osztályt)!

```
String s1 = new String(”Helló”);
s1 = s1 + ” Világ!”;
  /* s1 most más objektumot címez!*/
```

- A *String* osztály tartalmaz számos metódust ami a szövegek feldolgozásával kapcsolatos (egyenlőség vizsgálat, sorrend, konverziók, karaketerkhez hozzáférés)
- A *String* típus azért is nagy jelentőséggel bír, mert már az *Object* ősosztályban létezik egy olyan toString() nevű metódus, amellyel az adott objektum *String* típusúvá alakítható

Témakörök

Bevezetés, a Java kialakulása

Alapfogalmak

Alapvető változótípusok

Tömbök

Vezérlési szerkezetek

Standard I/O

Típusok használata (tömb típus)

- Egy névvel ellátott, indexkifejezéssel címzett folyamatos memóriaterület, mely primitív típusú adatokat/objektumreferenciákat tárol.
- Létrehozása:

```
int[] itomb = new int[12];
```

  Ez létrehoz és kezdőértékkel feltölt egy 12 elemű egydimenziós tömböt.
- Használata:

```
itomb[1]=123;
itomb[2]=itomb[1]+134;
```

- Többdimenziós tömbök is használhatók:

```
int[][] iitomb = new int [2][3];
```

  (mivel ez tömbök tömbje, lehet fűrészfogas is)

Típusok használata (tömb típus, 2.)

- A tömb méretét a length operátorral kérdezhetjük le:

```
int i = itomb.length;
```

- A tömb elemeinek indexe mindig 0-tól indul!

Feladat

1.2. feladat: Készítsük el azt a programot, amely 5 elemű egész tömb elemeit feltölti, majd pedig a képernyőre kiírja!

1.2.a. feladat: A feladatot ciklus utasítás nélkül oldjuk meg!

1.2.b. feladat: A feladatot ciklus utasítás segítségével oldjuk meg! Használjuk ki az a tényt, hogy a tömb „ismeri” a saját elemszámát!

Típusok használata (tömb típus, 3.)

- Van lehetőség objektumokat is tömbbe tenni, pl.

```
String[] s2 = new String[3];
```

- Létrehoz egy három String referenciát tartalmazó tömböt (kezdeti értéke három null referecia). Feltöltése:

```
s2[0] = ”Hahó”;
s2[1] = ”Java”;
S2[2] = ”SUN”;
```

- Megjegyzés: Az előbbiekben tárgyaltak statikus tömbök, a program futása során méretük nem változtatható

  Dinamikus viselkedésű tömböt a gyűjteményosztályok segítségével lehet létrehozni, például a java.util.Vector segítségével. Mivel ezek a tárolást az Object ősosztály referenciákkal oldják meg, tetszőleges típust képesek tárolni (egyidőben akár különböző típusú objektumokat is) Erről a megoldásról egy későbbi alkalommal lesz bővebben szó.

Feladat

1.3. feladat: Készítsük el azt a programot, amely 5 elemű String tömb elemeit feltölti, majd pedig a képernyőre kiírja!

1.3.a. feladat: A feladatot ciklus utasítás nélkül oldjuk meg!

1.3.b. feladat: A feladatot ciklus utasítás segítségével oldjuk meg! Használjuk ki az a tényt, hogy a tömb „ismeri” a saját elemszámát!

Témakörök

- Bevezetés, a Java kialakulása
- Alapfogalmak
- Alapvető változótípusok
- Tömbök
- Vezérlési szerkezetek
- Standard I/O

Vezérlési szerkezetek

Alapvető vezérlési szerkezetek:

- Szekvencia (utasítások egymás után történő vérehajtása)

```
utasítás1;
utasítás2;
```

- Elágazás (szelekció)
  - if utasítás

```
if(logikai feltétel)
utasítás;
```

vagy

```
if(logikai feltétel)
  utasítás1;
else
  utasítás2;
```

Vezérlési szerkezetek (2.)

– switch utasítás

```
switch(kifejezés){
case eset1: utasítások
case eset2: utasítások
                  break;
case eset3: utasítások
                  break;
default:  utasítások;
                  break;
}
```

A kifejezés byte, short, char vagy int típusú lehet.

Vezérlési szerkezetek (3.)

- Ciklus (iteráció):
  - Előltesztelő típusú (while):

```
while (logikai_feltétel)
  utasítás;
```

  - Hátultesztelős típus (do while)

```
do {
  utasítások
} while (logikai_feltétel);
```

  - Számláló ciklus (for)

```
for(kezdőérték;kilépési_feltétel;ciklus
 változó_módosítás)
  utasítás;
```

Feladat

1.4.a. feladat: Készítsük el azt a programot, amely megjeleníti a szorzótáblát! A program „szépen” adja a táblát, tehát az eredmények legyenek csoportosítva (pl. 10x10-es mátrix formájában)!

1.4.b. feladat: Módosítsuk az előző feladatot úgy, hogy a táblázat azokon a helyeken ahol nem prímszám szerepel, csak egy ‘*’ karaktert tartalmazzon!

Témakörök

- Bevezetés, a Java kialakulása
- Alapfogalmak
- Alapvető változótípusok
- Tömbök
- Vezérlési szerkezetek
- Standard I/O

Standard I/O

- A Java konzolos típusú I/O lehetőséggel:
  - Beolvasás (Stringet olvas be, más típusra át kell alakítani):
    `String System.console().readLine()`

Ezt az új lehetőséget az 1.6 verzió vezette be, ezért a fejlesztői eszközök (a NetBeans-t is beleértve) még nem támogatják: a fejlesztői környezeten belüli futtatáskor nem működik! Helyette:

- Fordítás után a futtatást parancssorból lehet elvégezni
- A grafikus felületet lehet használni
- A System.in streamen keresztül a szabványos bemenetet olvasni:

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
try {
        br.readLine()
} catch (IOException e) { }
```

  - Kiírás (Stringet ír ki):
    `System.out.println(String)`
    `System.out.print(String)`

Feladat

1.5. feladat: Készítsük el azt a programot, amely a parancssorból olvas be paramétereket, s azokat a képernyőre kiírja!

Megjegyzés: a parancssori paramétereket tartalmazó String tömb a main függvény egyetlen paramétere, s mindig kötelezően szerepel.

Az programot készítsük el konzolos megoldással, valamint a fejlesztő környezettel is!

Ez utóbbihoz ki kell derítenünk, hogy a környezet hol tartalmazza a parancssor elemeit!

Ajánlott irodalom

Az óra anyagához kapcsolódó irodalom

- Nyékyné Gaizler Judit: Java 2 útikalauz programozóknak 1.3 I. III.;
  ELTE TTK Hallgatói alapítvány, Budapest
  20-29. o.
  20-35. o.
  47-48. o.