Programozási Ismeretek

Dr. Gergely Tamás
Dr. Ferenc Rudolf, Dr. Jász Judit, Dr. Kiss Ákos

Szegedi Tudományegyetem
Informatikai Intézet
Szoftverfejlesztés Tanszék

2024

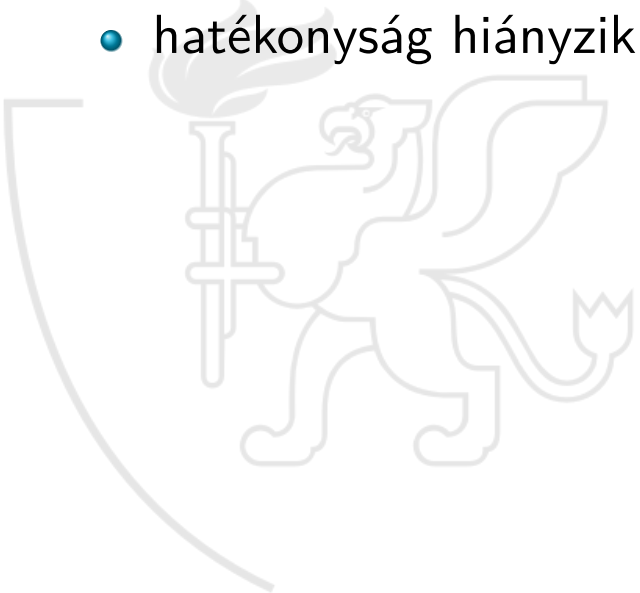
(v0214)

Tartalom

Tartalom

Java alkalmazások

- A Java megoldást ad a World Wide Web programozására (kliens/szerver elv)
- Önálló programok írására is alkalmas
- Előnyei
  - portabilitás
  - gyorsan írhatók megbízható, robusztus programok
- Hátrányai
  - hatékonyság hiányzik (több oka is van)



Miért ennyire sikeres?

- Fő célkitűzés: **Produktivitás**
- Alapoktól indulva lett megtervezve, az addigi programozási nyelvek tapasztalatait felhasználva
- OOP
- Jó osztálykönyvtárak
- Hibakezelés nyelvi szinten (kivételek)
- Nagy rendszerek írására is alkalmas
- Könnyű elsajátítani egy olyan szinten amivel már megbízható program írható
  - A C++ többet enged, nehezebb és ezért kevesebb a jó C++ programozó

Java vagy C++?

- Régebben állították, hogy a Java leváltja majd a C++-t
- Már nem így van
- A két nyelv más-más területeken előnyös
- Ha web programozás: Java
- Ha új nyelvet kell megtanulni, a Java könnyebb
- Első interpretált Java 20-50-szer lassabb volt mint a C/C++! Mára már jobb a helyzet, de drámai áttörés nem történt
  - JIT – Just in time
  - Sun hotspot
  - natív fordítók

Tartalom

Minden objektum . . .

- A Java nyelv feltételezi, hogy objektum orientáltan fognak programozni benne
- A programozónak objektum orientáltan kell gondolkodnia
- Egy Java program alapvető komponenseit tekintjük át, és látni fogjuk, hogy minden objektum, még maga a Java program is

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

Direkt tárolás

Indirekt tárolás

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

Direkt tárolás

```
int x = 6;
```

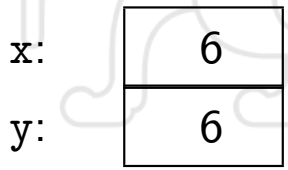
Indirekt tárolás

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

Direkt tárolás

```
int x = 6;
int y = x;
```



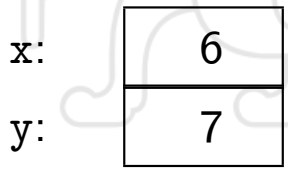
Indirekt tárolás

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

Direkt tárolás

```
int x = 6;
int y = x;
y += 1;
```



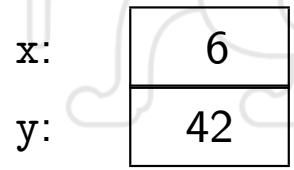
Indirekt tárolás

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

Direkt tárolás

```
int x = 6;
int y = x;
y += 1;
y *= x;
```



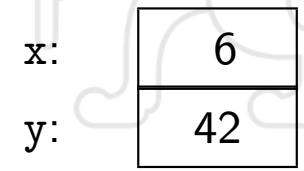
Indirekt tárolás

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

Direkt tárolás

```
int x = 6;
int y = x;
y += 1;
y *= x;
```



Indirekt tárolás

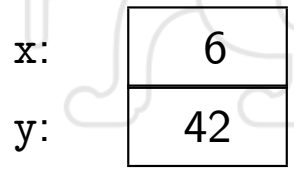
```
Integer x = new Integer(6);
```

Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

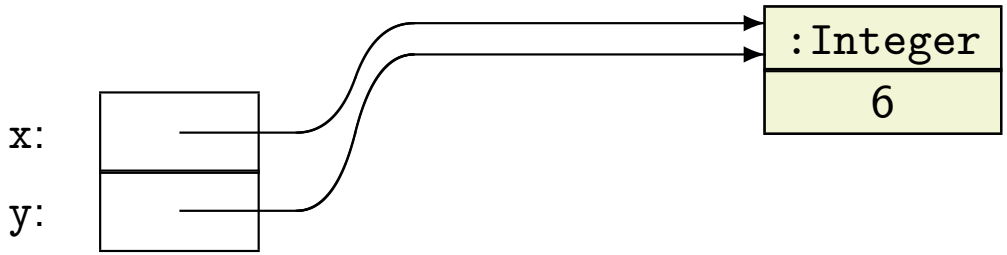
Direkt tárolás

```
int x = 6;
int y = x;
y += 1;
y *= x;
```



Indirekt tárolás

```
Integer x = new Integer(6);
Integer y = x;
```

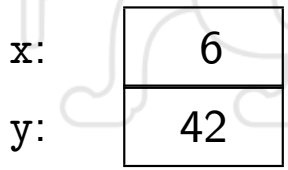


Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

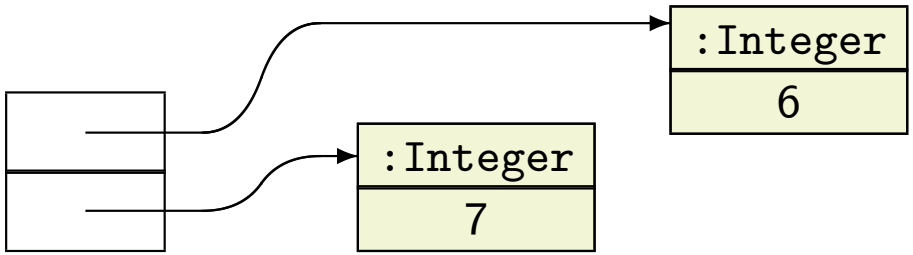
Direkt tárolás

```
int x = 6;
int y = x;
y += 1;
y *= x;
```



Indirekt tárolás

```
Integer x = new Integer(6);
Integer y = x;
y = new Integer(y.intValue() + 1);
```

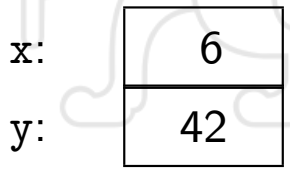


Objektum referenciák

- Adatok (objektumok) manipulálása lehet
  - Direkt (pl. C-ben egy változó)
  - Indirekt (pl. C/C++-ban pointer; C++ és Java-ban referencia)
- Java-ban minden objektum, és mindig referenciával hivatkozunk rájuk (kivéve a primitív típusokat)

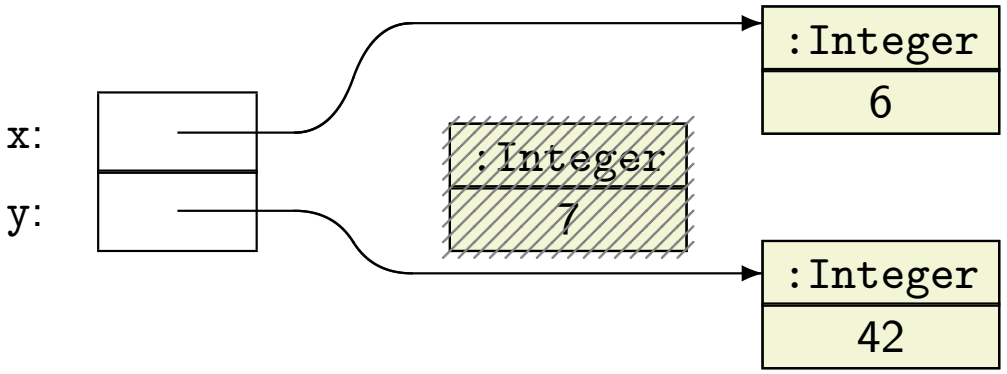
Direkt tárolás

```
int x = 6;
int y = x;
y += 1;
y *= x;
```



Indirekt tárolás

```
Integer x = new Integer(6);
Integer y = x;
y = new Integer(y.intValue() + 1);
y = new Integer(
        y.intValue() * x.intValue());
```



Tartalom

Primitív típusok

- A primitív típusok (hasonlóan a C/C++-hoz) a stack-en keletkeznek és direkt elérésűek (nem referencia által)
- Primitív típusok (fix bitszélesség):

Prim. típ.	Méret	Min	Max	Wrapper
boolean	–	–	–	Boolean
char	16 bit	Unicode `0`	Unicode `0xffff`	Character
byte	8 bit	-128	$+127$	Byte
short	16 bit	-2^{15}	$+2^{15}-1$	Short
int	32 bit	-2^{31}	$+2^{31}-1$	Integer
long	64 bit	-2^{63}	$+2^{63}-1$	Long
float	32 bit	IEEE-754	IEEE-754	Float
double	64 bit	IEEE-754	IEEE-754	Double
void	–	–	–	Void

Primitív típusok (folyt.)

- `boolean` értéke: `true` vagy `false`
- Wrapper: objektumba csomagolja a primitív típust

```
char c = 'x';
Character C = new Character('x');
```

- `BigInteger`: akármekkora egész szám
- `BigDecimal`: akármekkora fix pontos szám
- Tömbök
  - referenciákat tárol ill. primitív típusnál értékeket
  - automatikusan inicializálódnak (`null`, `0` vagy `0.0`)
  - indexhatár ellenőrzés van futáskor (biztonságos, de lassabb)

Új típusok létrehozása

- Új, egy osztályba tartozó objektumokhoz típus hozzárendelése: `class` kulcsszóval, pl.

```
class UjTipus {
    /* az osztály törzse */
}
// ...
UjTipus ut = new UjTipus();
```

- Önmagában még nem sok mindenre jó
  - csak létre lehet hozni,
  - de nem tud fogadni semmilyen üzenetet,
  - és nem tárol semmi hasznosat.
- Személyre kell szabni!

Attribútumok

- Osztály attribútuma (mezője, adattagja) lehet
  - másik osztály típusú (referenciát tárol), létre kell hozni `new`-val (inicializálás)
  - primitív típusú (értéket tárol, inicializálódik)
- A tagok elérése a „`.`” operátor segítségével
- Ez így önmagában még csak adattárolásra használható (mint egy C `struct`)

```
class UjTipus {
    int i;
    float f;
    Boolean b;
}
```

```
UjTipus ut = new UjTipus();
// ...
ut.i = 47;
ut.f = 1.1f;
ut.b = new Boolean(false);
```

Operációk

- Funkcionalitást biztosít az objektumoknak, meghatározza, hogy milyen üzeneteket fogadhat
- Részei: név, paraméterek, visszatérési típus, törzs

```
visszateresiTipus operacioNev( /* paraméter lista */ ) {
  /* metódus törzs */
}
```

- Csak osztályoknak lehetnek metódusai
- Metódus hívás = üzenet küldése az objektumnak

```
class UjTipus {
    int i;
    float f;
    Boolean b;
    int m(boolean b) {
        /*...*/
    }
}
```

```
UjTipus ut = new UjTipus();
// ...
ut.m(true);
```

Osztályok használata

- Egy fájlon belül
  - egyszerűen használni kell, az sem baj, ha csak később lesz definiálva
- Külső osztályok (pl. Java osztálykönyvtárak)
  - `import` kulcsszóval behúzhatunk egy osztályt/csomagot
  - pl. használni szeretném a láncolt listát:

```
import java.util.LinkedList;
```

  - pl. használni szeretnék mindent az `util` csomagból:

```
import java.util.*;
```

Tartalom

Az első Java program

- Ha önálló programot írunk Java-ban, akkor
  - az adott `.java` forrásfájlban kell lennie egy osztálynak, aminek ugyanaz a neve, mint a fájlnak, és
  - ez az osztály kell, hogy tartalmazzon egy statikus `main` metódust, a fenti paraméterezéssel.
- A `java.lang.System.out` egy statikus `PrintStream` típusú objektum, a `println()` pedig ennek egy metódusa
- A `java.util` csomag kell a `Date`-hez, a `java.lang` importja implicit

```
import java.util.*;

public class HelloDate {
    public static void main(String[] args) {
        System.out.println("Hello, a datum: ");
        System.out.println(new Date());
    }
}
```

Fordítás és futtatás

- Fordítás: forrás (.java) → bytecode (.class)
  - Kell egy Java fejlesztői csomag (JDK)
  - Feltételezzük „*A Java*” JDK-t (`https://www.oracle.com/technetwork/java/index.html`)

```
javac FORRÁSFÁJL(OK)
```

- Futtatás: a program végrehajtása interpreter segítségével
  - Kell egy Java futtató környezet (JRE)
  - A JDK tartalmazza a JRE-t is

```
java OSZTÁLYNÉV
```

- A `HelloDate` fordítása és futtatása:

```
javac HelloDate.java
java HelloDate
```

- Szokás valamilyen build-rendszert használni

Megjegyzések

- C/C++-szerű többsoros (blokk)

```
/* Ez egy
   több soros
   megjegyzés */
```

- C/C++-szerű egysoros

```
// Ez egy megjegyzés a sor végéig
```

Dokumentáció

- Egy Java forráshoz hozzátartozik annak dokumentációja speciális megjegyzések formájában: „/** ... */” vagy „/// ...”
- `javadoc`: program, ami összegyűjti ezeket a megjegyzéseket és készít egy HTML dokumentumot

```
/**
 * Megjegyzés a soron következő osztályhoz ...
 */
class A {
    /// ... de lehet attribútumhoz ...
    int i;
    /** vagy metódushoz is */
    float m() {
    ...
```

- HTML-t is be lehet ágyazni a dokumentációba, ezeket a javadoc átveszi

Dokumentáció
(folyt.)

- A dokumentációban használható (a `javadoc` által felismert) Tag-ek:
  - `@see`: más osztályokra hivatkozás – linket szúr be
  - `@version`: verzió
  - `@author`: szerző
  - `@param`: metódus paraméter szerepe
  - `@return`: metódus visszatérési értékének leírása
  - `@throws`: metódus milyen kivételeket dobhat

```
/**
 * @param T Hőmérséklet, Kelvin-fokban megadva
 * @return Az anyag térfogata T hőmérsékleten
 * @throws NegativErtekException Ha T negatív
 */
double volume(double T) {
...
```

Kódolási stílus

- Nem hivatalos Java standard
- Osztály nevek
  - nagy kezdőbetűvel kezdődnek
  - CamelCase: ha összetett szó, akkor nincs elválasztó karakter, hanem minden szó nagy betűvel kezdődjön

```
EgyJoOsztalyNev
```

- Metódus- és mező nevek:
  - kisbetűvel kezdődnek
  - CamelCase

```
egyJoTagNev
```

- {} zárójelezés mint a fóliákon

Tartalom

Operátorok

- Értékadás `=`
  - primitív típusoknál értékmásolás
  - objektum típusoknál csak **referencia másolás**
    (valódi objektum másolás a `clone()` metódussal végezhető)
- Matematikai `+, -, *, /, %`
- Egy operandusú `+, -`
- Inkrementáló/dekrementáló `++, --`
- Logikai bitműveletek `&, |, ^` (xor), `~` (not)
- Biteltolás `<<, >>, >>>` (unsigned)
- Értékadással összevont műv. `+=, -=, *=, /=, %=, &=, |=, ^=, <<=, >>=, >>>=`
- Relációk `==, !=, <, >, <=, >=`
  - primitív típusoknál érték összehasonlítás
  - objektum típusoknál csak **referencia összehasonlítás**
    (objektum érték összehasonlítás az `equals()` metódussal végezhető)

Operátorok
(folyt.)

- Logikai `&&`, `||`, `!`
  - C/C++-szal ellentétben csak `boolean` értékekre használható
  - összetett kifejezés csak addig értékelődik ki, amíg ki nem derül egyértelműen az értéke
- Háromoperandusú if-else `bool-expr?val0:val1`
- Típuskonverzió `(type)value`
  - primitív típusok között használható korlátok nélkül (kivéve a `boolean`-t)
  - osztályok között csak egy öröklődési fán belül engdélyezett
- **Nincs** `,` (vessző) operátor
  - Vessző van (pl. `for` ciklusban vagy metódushívásnál), de operátorként nem használható
- **Nincs** `sizeof()`

Precedencia

Operátor típus	Operátorok
Post inc/dec	`x++`, `x--`
Egyoperandusú	`+`, `-`, `++x`, `--x`, `~`, `!`
Típuskonverzió	`(type)x`
Multiplikatív	`*`, `/`, `%`
Additív	`+`, `-`
Bitléptetés	`<<`, `>>`, `>>>`
Relációs	`<`, `>`, `<=`, `>=`
Egyenlőség	`==`, `!=`
Bitenkénti és	`&`
Bitenkénti kizáró vagy	`^`
Bitenkénti vagy	`|`
Logikai és	`&&`
Logikai vagy	`||`
Feltételes	`c?t:f`
Értékadó	`=`, `+=`, `-=`, `*=`, `/=`, `%=`, `&=`, `|=`, `^=`, `<<=`, `>>=`, `>>>=`

Tartalom

Szelekció

`if else`, `switch`

- A Java támogat minden C vezérlési szerkezetet, kivéve a `goto`-t
- Egyszerű szelekció
  - csak logikai kifejezéssel

```
if (condition)
    statement;
else
    statement;
```

- Esetkiválasztásos szelekció
  - egész értékű primitív típusokkal és wrapper osztályaikkal, `String`-gel és `enum` típusokkal

```
switch (selectorExpression) {
    case constVal: statement; break;
    // ...
    default: statement; break;
}
```

Ismétléses vezérlések

`while`, `do while`

- Kezdőfeltételes ismétlés
  - csak logikai kifejezéssel

```
while (condition)
    statement;
```

- Végfeltételes ismétlés
  - csak logikai kifejezéssel

```
do
    statement;
while (condition);
```

Ismétléses vezérlések

for

- Számlálásos ismétlés
  - csak logikai típusú feltétel-kifejezéssel

```
for (initialization; condition; increment)
    statement;
```

- Diszkrét ismétlés
  - Iterable típusokra (ilyen C-ben nem volt)

```
for (variable : collection)
    statement;
```

Vezérlésátadás

`break`, `continue`, `return`

- Ciklus (és `switch` vagy cimkézett blokk) megszakítása
  - megszakítja az aktuális (vagy a cimkézett) vezérlési blokkot és az azt követő utasítással folytatja a programot

```
break;
break labelOfBreakable;
```

- Ciklus folytatása
  - az aktuális (vagy a cimkézett) ciklus következővel iterációval folytatja a programot

```
continue;
continue labelOfLoop;
```

- Visszatérés metódushívásból

```
return value;
return;
```