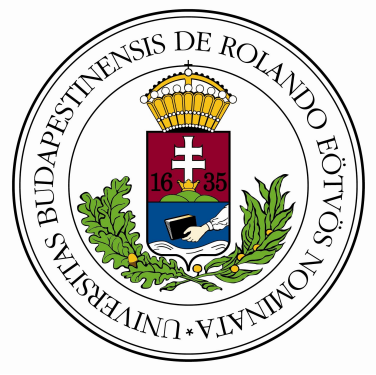
Programozási nyelvek – Java

Kifejezések, utasítások



Kozsik Tamás

ELTE Eötvös Loránd Tudományegyetem

Forráskód felépítése

- fordítási egységek
- típusdefiníciók
- metódusok
- utasítások
- kifejezések
- lexikális elemek
- karakterek

Outline

Karakterkódolási szabványok

character encodings

- Bacon's cipher, 1605 (Francis Bacon)
- Baudot-code, 1874
- BCDIC, 1928 (Binary Coded Decimal Interchange Code)
- EBCDIC, 1963 (Extended ...)
- ASCII, 1963 (American Standard Code for Information Interchange)
- ISO/IEC 8859 (Latin-1, Latin-2,...)
- Windows 1250 (Cp1250)
- Unicode (UTF-8, UTF-16, UTF-32)

lásd: `iconv` (Unix/Linux)

Fordítás: karakterkódolás

- Legyen `Main.java` magyar Windowsos karakterkódolású.
- Legyen a gépemen más, pl. UTF-8 karakterkódolás.

Fordítás: karakterkódolás

- Legyen `Main.java` magyar Windowsos karakterkódolású.
- Legyen a gépemen más, pl. UTF-8 karakterkódolás.

Fordítás alapértelmezett karakterkódolással

```
$ javac Main.java
Main.java:2: error: unmappable character (0xE1) for encoding UTF-8
    String hib?s;
```

Fordítás: karakterkódolás

- Legyen `Main.java` magyar Windowsos karakterkódolású.
- Legyen a gépemen más, pl. UTF-8 karakterkódolás.

Fordítás alapértelmezett karakterkódolással

```
$ javac Main.java
Main.java:2: error: unmappable character (0xE1) for encoding UTF-8
    String hib?s;
```

Átváltás magyar Windowsos karakterkódolásra

```
$ javac -encoding Cp1250 Main.java
```

Outline

Lexikális elemek

- Kulcsszavak
- Azonosítók
- Operátorok
- Literálok
- Zárójelek: `(.) [.] {.} <.>`
- Speciális jelek: `. , : ; -> | ... :: @`
- Megjegyzések (egysoros, többsoros, „dokumentációs'')

Ebben a félévben tanulandó kulcsszavak és foglalt szavak

Utasítások: `if else switch case default while do for break continue return try catch finally throw assert yield`

Programszerkezet: `package import class enum interface extends implements`

Deklarációk: `public protected private abstract static final throws`

Típusok: `boolean char byte short int long float double void`

Speciális változók, konstruktorok: `this super`

Operátorok: `instanceof new`

Literálok: `true false null`

További kulcsszavak és foglalt szavak

Típuskikövetkeztetés

`var`

Deklarációkban

`synchronized volatile transient strictfp native`

Nem használt foglalt szavak

`_ const goto`

Moduldeklarációkban használt kulcsszavak

`module exports open opens provides requires uses with to transitive`

Azonosítók

- betűk, számjegyek, _ és $
- Unicode betűk, akár szökőév vagy ε

Konvenciók

```
package java.lang;
public final class Integer ... {
    ...
    public static final int MAX_VALUE = 2147483647;
    public int intValue(){ ... }
    ...
}
```

Literálok

- Logikai ~: `true` és `false`
- Karakter-~: `'c'`, `'\t'`, `'\''`, `'\\'`, `'\uBABE'`
- Szöveg-~: `"this is a string\ncontaining \u0032 lines"`
- Egész ~
  - `int` típusú: `1984`, `9_772_756`, `0123`, `0XBee`, `0xCAFE_BABE`, `0b1010101`
  - `long` típusú: `1984L`, `1984l`, `0xDEAD_BEEF_ADDED_C00L`
- Lebegőpontos ~
  - `double` típusú: `3.14159`, `.000_001`, `1E-6`, `6.022140857e23`, `3.`, `3D`, `3.14d`, `0x1.Bp-2` = `(1+11./16)/4`, `0X1DE.1P0D`
  - `float` típusú: `3.14159F`, `.000_001f`…

Outline

Kifejezések

- szintaxis: operátorok arítása, fixitása; zárójelezés
- kiértékelés
  - precedencia (`A + B * C`)
  - asszociativitás (`A - B - C`)
  - operandusok/paraméterek kiértékelési sorrendje (`A + B`, `f(A,B)`)
  - lustaság (`A ? B : C`)
  - mellékhatás (++x)

Példa mellékhatásos kifejezésre: *olvasás EOF-ig* idiómája

Legyen `in` egy bemeneti adatfolyam, pl. megnyitott fájl.

```
int v;
while( (v=in.read()) != -1 ){
    ...
}
```

Példa mellékhatásos kifejezésre: *olvasás EOF-ig* idiómája

Legyen `in` egy bemeneti adatfolyam, pl. megnyitott fájl.

```
int v;
while( (v=in.read()) != -1 ){
    ...
}
```

Két mellékhatás a ciklus feltételében!

Logikai műveletek kiértékelése

- Lusta: `A && B`, `A || B`

Logikai műveletek kiértékelése

- Lusta: `A && B`, `A || B`
- Mohó: `A & B`, `A | B`

(`A` és `B` típusa `boolean`)

Lusta és mohó művelettábla

Jelölje $\uparrow$, $\downarrow$, $\bot$ és ∞ a négy lehetséges eredményt egy logikai kifejezés kiértékeléséhez: igaz, hamis, kivétel, nem terminálló számítás. Az $\alpha \wedge \beta$ kifejezés értéke az α és β értékének függvényében (a mellékhatásoktól itt eltekintünk):

α && β	$\beta = \uparrow$	$\beta = \downarrow$	$\beta = \bot$	$\beta = \infty$
$\alpha = \uparrow$	$\uparrow$	$\downarrow$	$\bot$	∞
$\alpha = \downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$
$\alpha = \bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\alpha = \infty$	∞	∞	∞	∞

α & β	$\beta = \uparrow$	$\beta = \downarrow$	$\beta = \bot$	$\beta = \infty$
$\alpha = \uparrow$	$\uparrow$	$\downarrow$	$\bot$	∞
$\alpha = \downarrow$	$\downarrow$	$\downarrow$	$\bot$	∞
$\alpha = \bot$	$\bot$	$\bot$	$\bot$	$\bot$
$\alpha = \infty$	∞	∞	∞	∞

Példa mohó logikai operátorra

```
int v1, v2;
while( ((v1 = in1.read()) != -1) | ((v2 = in2.read()) != -1) ){
    if( v1 == -1 ){
        out.write( v2 );
    } else if( v2 == -1 ){
        out.write( v1 );
    } else {
        out.write( v1+v2 );
    }
}
```

Bitműveletek

- Bitenkénti *és*elés és *vagy*olás: `A & B`, `A | B`
  (`A` és `B` típusa `int` vagy `long`)

Bitműveletek

- Bitenkénti *és*elés és *vagy*olás: `A & B`, `A | B`
  (`A` és `B` típusa `int` vagy `long`)
- XOR: `A ^ B`

Bitműveletek

- Bitenkénti *é*selés és *vagy*olás: `A & B`, `A | B` (`A` és `B` típusa `int` vagy `long`)
- XOR: `A ^ B`
- Bitenkénti ellentett: `~A`

Bitműveletek

- Bitenkénti *és*elés és *vagy*olás: `A & B`, `A | B` (`A` és `B` típusa `int` vagy `long`)
- XOR: `A ^ B`
- Bitenkénti ellentett: `~A`
- Léptetés: `A << B`, `A >> B`, `A >>> B`

Operátorok túlterhelése, új operátorok definiálása

- Csak a beépített operátorok (nem lehet újat definiálni)
- Csak a beépített jelentéssel (nem lehet túlterhelni)
  - Beépített túlterhelés: pl. + vagy &

Outline

Utasítások

- Kifejezéskiértékelő utasítás
  - Értékadások
  - Metódushívás
- `return`-utasítás és `yield`-utasítás
- Elágazások (`if`, `switch`)
- Ciklusok (`while`, `do`–`while`, `for`)
- Nem strukturált: `break`, `continue`
- Blokk-utasítás
- Deklaráció (pl. változó~)
- Kivételkezelő és -kiváltó utasítások
- `assert`-utasítás

Puzzle 22: Dupe of URL (Bloch, Gafter: Java Puzzlers)

```
class Main {
    public static void main( String[] args ){
        https://jdk.java.net/
        System.out.println();
    }
}
```

Puzzle 22: címkézett utasítás

```
class Main {
    public static void main( String[] args ){
        https://jdk.java.net/
        System.out.println();
    }
}
```

switch-utasítás

- egész típusokra
- felsorolási típusokra
- `String` típusra

switch-utasítás felsorolási típusra

```
static int workingHours( Day day ){
    switch( day ){
        case SUN:
        case SAT: return 0;
        case FRI: return 6;
        default:  return 8;
    }
}
```

switch-utasítás Stringre

```
static int workingHours( String day ){
    switch( day ){
        case "SUN":
        case "SAT": return 0;
        case "FRI": return 6;
        default:    return 8;
    }
}
```

Hagyományos switch-utasítás

```
String name;
switch( dayOf( new java.util.Date() ) ){
    case  0: name =    "Sunday"; break;
    case  1: name =    "Monday"; break;
    case  2: name =   "Tuesday"; break;
    case  3: name = "Wednesday"; break;
    case  4: name =  "Thursday"; break;
    case  5: name =    "Friday"; break;
    case  6: name =  "Saturday"; break;
    default: throw new Exception("illegal value");
}
```

Biztonságosabb switch-utasítás

```
String name;
switch( dayOf( new java.util.Date() ) ){
    case  0 -> name =    "Sunday";
    case  1 -> name =    "Monday";
    case  2 -> name =   "Tuesday";
    case  3 -> name = "Wednesday";
    case  4 -> name =  "Thursday";
    case  5 -> name =    "Friday";
    case  6 -> name =  "Saturday";
    default -> throw new Exception("illegal value");
}
```

switch-kifejezés

```
String name = switch( dayOf( new java.util.Date() ) ){
                    case  0 ->    "Sunday";
                    case  1 ->    "Monday";
                    case  2 ->   "Tuesday";
                    case  3 -> "Wednesday";
                    case  4 ->  "Thursday";
                    case  5 ->    "Friday";
                    case  6 ->  "Saturday";
                    default -> throw new Exception("illegal value");
                };
```

Túlcsorgás

```
switch(month){
  case  4:
  case  6:
  case  9:
  case 11: days = 30;
           break;
  case  2: days = 28 + leap;
           break;
  default: days = 31;
}
```

```
days = switch(month){
         case 4, 6, 9, 11 -> 30;
         case 2  -> 28 + leap;
         default -> 31;
       };
```

yield-utasítás

```
int days = switch(month){
               case 4, 6, 9, 11 -> 30;
               case 2 -> { int leap = 0;
                           if( year % 4 == 0 ) leap = 1;
                           if( year % 100 == 0 ) leap = 0;
                           if( year % 400 == 0 ) leap = 1;
                           yield 28 + leap;
                         }
               default -> 31;
           };
```

Nem triviális túlcsorgás

```
enum States {RED, AMBER, GREEN};
...
switch ( trafficlight ){
    case RED:    stop();
                 break;
    case AMBER:  if( canSafelyStop() ){
                     stop();
                     break;
                 }
    case GREEN:  go();
}
```

Javában nem, de C-ben ilyen is írható

```
switch ( trafficlight ){
    case AMBER:  if( canSafelyStop() ){    // not valid in Java
    case RED:        stop();
                     break;
                 }
    case GREEN:  go();
}
```