Imperatív és procedurális programozás a Javában

Kozsik Tamás

kto@elte.hu
http://kto.web.elte.hu/

Eötvös Loránd Tudományegyetem
Programozási Nyelvek és Fordítóprogramok Tanszék

2008.

Tartalom

Tartalom

Áttekintés

- Olcsó programfejlesztés
- Biztonságra való törekvés
  - Safety
  - Security
- Memóriakezelés
- Szintaktikai szabályok – megörökölte a C betegségeit
- Statikus szemantikai szabályok
- Dinamikus szemantikai szabályok
- `http://java.sun.com/`

Típusok

- Primitív típusok
- Referencia típusok
  - Osztályok (konkrét és absztrakt)
    - Tömb típusok
    - Felsorolási típusok
  - Interfészek
    - Annotációk

Változók típusa:

csak primitív vagy referencia

Primitív típusok

- 8 db beépített típus
  - `boolean`
  - `char`
  - `byte`, `short`, `int`, `long`
  - `float`, `double`
- Rögzített ábrázolás
- Nincs előjel nélküli egész típus
- A `char` típus 2 bájtos Unicode
- A logikai típus önálló (vezérlési szerk., relációk)
- Altípusosság; konverzió: automatikusan csak bővítő (!)
- Osztályosítás: csomagoló osztályokkal
  - Auto-(un)boxing
- Gyatra aritmetika
- `BigDecimal`, `BigInteger`
- `strictfp`, IEEE 754

Referenciák

- A vermen csak primitív típusú adatok és referenciák
- Az összetett adatok mindig dinamikusak
- Felszabadító utasítás nincs
- Automatikus szemétgyűjtés
- Destruktor – `finalize`

Tömbök

- Referenciák
  - Nem lehet deklarációval létrehozni
  - Minden tömb a heap-en van
- Speciális osztályok, öröklődés
- Biztonságos használat
  - `length` attribútum
  - Futási idejű indexellenőrzés

Használat

```
int[] t = new int[100];
for( int i = 0; i<t.length; ++i ){ t[i] = i; }
```

Szövegek

- Nincs `char*`
- Elsődlegesen a `String` osztály
  - Pl. idézőjelek
  - Konkatenáció: túlterhelt `+`
- Emellett `StringBuffer` és `StringBuilder`
- Ritkán `char[]`

Felsorolási típusok

A legegyszerűbb eset

```
enum Color { RED, GREEN, BLUE, YELLOW }
```

- Nem `int`
- Speciális osztályok
- Definiálhatók hozzájuk mezők és műveletek

Hiányosságok

- Alprogram típus
- Primitív típusok és referenciák összeférhetetlensége
- Primitívből és tömbből származtatás hiánya
- Fixpontos, moduló

Vezérlési szerkezetek

- Elágazás: `if` és `switch–case`
- Ciklus: `for`, `while` és `do–while`
  - Iterálás adatszerkezeten: enhanced for-loop
- Blokk utasítás
- Nem strukturált lehetőségek

Iterálás tömbön

```
int[] t = new int[100]; ...
int sum = 0;
for( int elem: t ){ sum += elem; }
```

Strukturált vezérlési szerkezetek

- `if` és `while` esetén: logikai típusú feltétel

Többágú elágazás tördelése

```
if( <feltétel-1> ){
   <utasítások>
} else if( <feltétel-2> ){
   <utasítások>
} else {
   <utasítások>
}
```

- `switch` esetén diszkrét vagy felsorolási típusú diszkrimináns
  - A minták fordítási idejű konstansok
  - `break`!!!

A switch és a break

```
switch (billentyű) {
  case 'w': y++; break;
  case 'z': y--; break;
  case 'a': x--; break;
  case 's': x++; break;
  default : System.out.println("Hiba!");
}
```

Nem strukturált vezérlés

- Címkézhető `break` és `continue`
- `return`
- Nincs `goto`

Hatékonyság növelése

```
int i=0, j=0;
külső: for (; i<10; ++i)
  for (j=0; j<10; ++j)
    if (tömb[i][j] == 0)
      break külső;
```

Deklaráció utasítások

- Típus-, alprogram- és változódeklarációk
- Konvenció: azonosítók neve
- Azonosítók lexikális szabályai
  - Unicode
  - _ és $
- Egy deklarációban több változó
- Inicializáció
- Módosítószavak
- A léptetős ciklusnak lehet lokális változója, de a többi utasításnak nem!
- Konvenció: használjuk a kapcsos zárójeleket, tördelés
- Típuskifejezésekhez nem deklarálható név

Kifejezés, mint utasítás

- Kifejezések értéke figyelmen kívül hagyható
- Mellékhatás
- Értékadás(ok)
- Függvényhívás
  - `void` metódus
  - Kivételek
  - Sorozás

Blokkok

- Egyszerűsített hatóköri/láthatósági szabályok
- Egymásba ágyazott blokkok lokális változói
- Elfedés: csak tagok esetében
- Lokális változók inicializálása

Megjegyzések

- Egysoros és többsoros
- Dokumentációs megjegyzés
  - `/**` és `*/` között
  - Kódegység elé írható
  - `javadoc`
  - Tipikusan HTML, de programozható (doclet)
  - Kiegészítő információk (`@return`, `@param`, `@see` stb.)

Problémák

- Utasítások és kifejezések keveredése
- A szintaxis bénasága, túlzott tömörség

```
int sum = 0, i = 1;
while( i<10 );
    sum += i++;
```

- `switch` utasítás nem intuitív

Csellengő else

```
if (a==1)
  if (b==2)
    c = 1;
else
  c = 2;
```

Problémák

- Utasítások és kifejezések keveredése
- A szintaxis bénasága, túlzott tömörség

```
int sum = 0, i = 1;
while( i<10 );
    sum += i++;
```

- `switch` utasítás nem intuitív

Csellengő else

```
if (a==1)                if (a==1)
  if (b==2)                if (b==2)
    c = 1;                   c = 1;
else                     else
  c = 2;                   c = 2;
```

Problémák

- Utasítások és kifejezések keveredése
- A szintaxis bénasága, túlzott tömörség

```
int sum = 0, i = 1;
while( i<10 );
    sum += i++;
```

- `switch` utasítás nem intuitív

Csellengő else

```
if (a==1)
  if (b==2)
    c = 1;
else
  c = 2;
```

```
if (a==1)
  if (b==2)
    c = 1;
  else
    c = 2;
```

```
if (a==1){
  if (b==2)
    c = 1;
} else
    c = 2;
```

Kifejezések

Felépítés:

- literál
- változó
- operátor
- metódushívás (függvény)
- zárójelek
- tömbindexelés, szelekció

Kiértékelési sorrend

- Teljesen definiált
- Operátorok precedenciája
- Operátorok asszociativitása
- Zárójelezés
- Argumentumok kiértékelési sorrendje

Operátorok

- Arítás
- Fixitás
- Precedencia, asszociativitás
- Túlterhelt operátorok
- Operátorokat nem terhelhet túl a programozó
- Új operátorokat sem definiálhat

Java operátorok

- C-ből megszokottak
  - Aritmetikai, relációs, értékadó, bitmanipuláló
  - Elágazó
  - Értékadások, sorozás, hatékonyság
- Van `instanceof`, valamint `>>>` és `>>>=`
- A `+` operátor jelentései
- Lusta és mohó logikai operátorok

Literálok

- Egész literálok
- Lebegőpontos literálok
- Karakterliterálok
- Stringliterálok
- Logikai típusértékek: `true`, `false`
- `null`
- `class`-literál

Egész literálok

- Alapból `int` típusú
  - Decimális
  - Hexadecimális
  - Oktális
  - Nincs bináris vagy egyéb!
- `long` típusú
- `short` vagy `byte` típusú
- Automatikus szűkítő konverzió

Lebegőpontos literálok

- Alapból `double` típusú
- `float` típusú
- Van `0.0` és `-0.0`
- Konstansok, pl: `Float.POSITIVE_INFINITY`, `Double.NaN`

Szöktetés

- Karakter- és stringliterálokban
- A \ karakterrel
- Speciális karakterek
  - Tabulátor, kocsivissza, újsor, lapdobás...
  - Rep-jel
  - Idézőjelek
- Karakterek Unicode megadása
  - `\u000a` és `\u000d` kizárva!
- ASCII karakterek oktális megadása

Konstansok

- `final` változók
  - primitív típus
  - referencia
- Üres konstans
- Fordítási idejű konstans
- Feltételes fordítás

Programegységek

- Típusdefiníciók
  - Mezők (példány- és osztály-)
  - Metódusok (példány- és osztály-)
  - Típusdefiníciók (példány- és osztály-)
  - Konstruktorok
  - Inicializátor blokkok (példány- és osztály-)
- Nincsenek globális alprogramok
  - Példány- vagy osztálymetódusok
  - Procedurális programozás imitálása
- Nincsenek globális változók sem!
- Csomagok

Alprogramok

- Függvény és eljárás összemosása: metódus
- Visszatérési érték
- Verem, aktivációs rekord, automatikus változók
- Paraméterátadás (call-by-value, call-by-sharing)
- Nem adható formális paraméternek alapértelmezett érték
- Túlterhelés
- Inline-osítás
- Felüldefiniálás, dinamikus kötés
- Hívási lehetőség: invokevirtual, invokestatic, invokespecial
- Változó hosszúságú paraméterlista
  - `printf`