Java VIII.
Az interface és az `instanceof` operátor

Krizsán Zoltán

Ficsor Lajos

Miskolci Egyetem

Általános Informatikai Tanszék

Utolsó módosítás: 2005. 10. 24.

Az interfészről általában

- Egy osztály interfészén a nyilvános elemeinek összességét értjük, ami a használatához szükséges.
- Az információ rejtés elve miatt, általában csak metódusokból áll.
- Specifikáció, tervezés során készül el.
- Kényszeríti az osztály készítőjét a megfelelő elemek implementálására.
- „Protokollok” (a kommunikáció szabályai) az objektumok között.

Interfészek JAVA-ban

- Mivel nincs globális változó, konstans, ezért lehetnek adat részei is.
- Szintaktikája hasonló az osztályhoz, de a **`class`** kulcsszó helyett **`interface`.**
- Kiterjeszthet más interfészeket (**`extends`**), létezik többszörös interfész öröklés **.**
- Az interfész nem tartalmaz végrehajtható kódot, azt a megvalósító osztályban (**`implements`**)kell megadni.
- Megállapodás, hogy „I”-vel kezdődik a neve.

„Tiszta absztrakt osztály”

A "legabsztraktabb osztály", mivel tartalma:

- Csak a metódusok prototípusa, nincs implementáció
  - Név
  - Paraméter szignatúra
  - Visszatérési érték
- Minden metódus implicite
  - **`public`**
  - **`abstract`**

„Tiszta absztrakt osztály”

- Konstans definíciókat is tartalmazhat, azok minősítése alapértelmezés szerint
  - **`public`**
  - **`static`**
  - **`final`**
- Nincs statikus inicializáló blokkja, csak a deklaráció során inicializálható.
- Értékét be lehet állítani:
  - Futás időben, ha az inicializálás csak akkor kiértékelhető elemeket tartalmaz
  - Fordítási időben, ilyenkor hagyományos konstans.

Szintaktika

```
[módosító] interface lazonosito
  [extends Ios1[,Ios2]]
{
[Elemek deklarációi]
}
```

Módosító lehet:

- **`public`**
- **`abstract`** (elavult, mert minden interfész alapértelmezés szerint absztrakt)

Interfészek kiterjesztése

- Minden interfész kiterjeszthet **egy vagy több** interfészt
- Fontos, hogy önmagát nem terjesztheti ki sem közvetve, sem közvetlen. Nem alakulhat ki körkörös lánc.

Interfészek láthatósága

- **`public`**: nyilvános
  Ebben az esetben az interfészt azonos nevű fájlba kell írni, 1 interfész 1 fájl.
- `-`: csomagszintű
  Ilyenkor csak abban a csomagban lehet használni, de több is lehet egy forrásfájlban.

Új osztály szintaktika

[módosító] class osztálynév [**extends** ősosztály] [**implements lint1**[,**lint2**[,...]]]

{

[Elemek deklarációi]

}

Szabályok

- Ha egy osztály implementál egy interfészt, akkor köteles annak minden metódusát implementálni!
- Az implementált elemeket nem módosíthatja.
- Metódusok esetében a fejlécnek teljesen egyeznie kell!
- Azonos nevű elemet az öröklődés során és az implementálás során nem kaphat meg (fordítási hiba).
  - Azaz például az ős osztályban és valamelyik implementált interface-ben nem lehet azonos nevű metódus.

Interfész használata

- Egy interfész új referencia típust vezet be => mindenhol használható, ahol egy osztály.
- Változó deklarációban szerepelhet (megadhatja annak statikus típusát).
- Bármelyik osztállyal helyettesíthető, amely implementálja (megadva ezzel a dinamikus típust).

Forgatókönyv I.

- Hozzuk létre az interfészt, ha szükséges örököltessük más(ok)ból

```
interface Itest{
  boolean jo = true;
  boolean rossz = false;
  boolean joE();
}
```

Forgatókönyv II

- Implementáljuk az interfészt:

```
class Munkadarab implements Itest{
   int hossz;
   boolean joE() {
         if (hossz > 100)
             return jo;
         else
             return rossz;
     }
   }
```

Forgatókönyv III

- Használjuk az objektumot az interfész referenciáján keresztül:

```
class tarolo{

...

Vector m_munkadarabok;

public void ujDarab(munkadarab
  ujdb){
          m_munkadarab.add(ujdb);
    }
```

Forgatókönyv IV

```
public kiirSelejtSorszam(){
// univerzális kód, bármely Itest-et
// implementáló osztályra működik
for(int i=0; i < m_munkadarabok.size(); i++){
if((!(Itest)m_munkadarabok.elementAt(i)).joE())
{
System.out.println(i);
}
…
}// class tarolo vege
```

Absztrakt osztály és interface

- Vannak hasonlóságok és különbségek.
- Az interface tisztán osztály vázat definiál, az absztrakt osztályt tartalmazhat implementációt is.
- Nem lehet példányosítani objektumot egyikből sem.
- Mindkettő típust definiál.
- Összehasonlítás táblázatosan:

Absztrakt osztály és interface

	Absztrakt osztály	Interface
Adattag	tetszőleges	csak final
Metódus	min. egy absztrakt	csak absztrakt
Metódus implementáció	tartalmazhat	nem tartalazhat
Példányosítás	nem lehetséges	nincs értelmezve
Metódus hívás	csak static	nincs értelmezve
Öröklés	lehet ős osztály	lehet más interface (egyik) őse
Metódus megvalósítás	leszármazott felüldefiniálhatja a metódusokat	osztálynak kell definiálnia a metódusokat

Az `instaceof` operátor

- Logikai művelet. Az
  `a instanceof B`
  művelet igaz, ha az **`a`** változó dinamikus típusa leszármazottja a **`B`** típusnak, egyébként hamis.
  - Akkor igaz tehát, ha az **`a`** változó dinamikus típusát meghatározó osztály maga a **`B`** osztály, vagy annak leszármazottja, vagy implementálja a **`B`** interface-t.
- Az **`instanceof`** precedenciája azonos a **`< > <= >=`** operátorokéval.