Java és web programozás

Kovács Kristóf, Rimay Zoé

Budapesti Műszaki Egyetem

2013. október 9.

5. Előadás

Interface-ek

- Korábban már említettem az interface-eket. Akkor úgy fogalmaztam, hogy valamilyen tulajdonságot adnak az osztályoknak.
- A lényegüket talán abból lehet legjobban megérteni, hogy miért találták ki őket.
- Képzeljük el azt a helyzetet, hogy több fejlesztői cégnek meg kell állapodni valamilyen szabványon, ami szerint az általuk készített hasonló (de nem azonos) kódokat lehet használni.

Példa

- Képzeljük el, hogy léteznek gépek által vezérelt autók (tényleg léteznek). Ezeket különböző autó gyártók gyártják, sőt az őket koordináló elektronikát pedig teljesen más cégek gyártják.
- Az autó gyáraknak és az elektronika gyártóknak meg kell egyezniük egymással, hogy az autókat milyen metódusokkal lehet irányítani, hogy majd ezt használhassák az elektronika gyártók.
- Ha sikerül megegyezniük valamilyen szabványban, akkor mindenkinek könnyű dolga van, hisz ha az összes autógyár ezt a szabványt használja, akkor az összes elektronika gyár tudja használni a szabvány metódusokat, így nem kell minden autóra külön szoftvert írnia.

Interface-ek folytatás

- Az interface-ek ezt a szabványt biztosítják.
- Az interface-eknek jóformán csak konstansai és metódus deklarálásai lehetnek.
- Deklarálás csak és nem definíció. Az interface csak azt mondja meg, hogy milyen bemenetei vannak a metódusnak, milyen típusú a visszatérési érteke és mi a metódus neve.
- Amikor egy osztály *implementálja* az adott interface-t, akkor kötelező megírnia az interface-ben szereplő metódusokat.

Konkrét példa

```
public interface Movable {
    public boolean moveForward(double meters,
            double velocity);
    public boolean turnLeft(double degrees,
            double angularVelocity);
    ...
}
```

- A ... nem a szintaktika része, csak jeleztem, hogy a többi metódust is megírhatnánk.
- Hasonlóan hozunk létre interface-t mint osztályt, azzal a különbséggel, hogy itt a metodusoknak nincs kapcsos zárojeles blokkja, hanem pontosvesszővel zárjuk a deklarációjukat.

Implementálás

- Most, hogy már van interface-ünk implementálni kellene egy osztályban. Ezt az *implements* kulcsszóval tehetjük:

```
public class FutureCar implements Movable {
    public boolean moveForward(double meters,
            double velocity) {
        // TODO
        return false;
    }

    public boolean turnLeft(double degrees,
            double angVelocity) {
        // TODO
        return false;
    }
}
```

Rosszul implementált interface

- Ha nem definiáljuk egy implementált interface metódusat az osztályban, akkor futni se fut a programunk.
- Ezzel garantálja, hogy ha valamit egy adott interface-el láttunk el, akkor ő tud is úgy működni, mint ahogy azt az interface megmondja.

Mire lesz jó nekünk

- Nekünk igaz nem kell gyárakkal összehangolni a kódjaink működését, de igy is nagyon hasznosak lesznek az interface-ek.
- Képzeljük el, hogy a Facebook postok interface-ekkel működnek.
- Milyen kényelmetlen lenne, ha valahogy külön kellene tárolni a képeket és szövegeket, esetleg linkeket amiket postol az ember.
- Ehelyett tehetjük azt, hogy mindet egy *Postable* interface-el implementáljuk. Ekkor tárolhatjuk mindet például egy *Vector<Postable>*-ben. És amikor meg akarjuk jeleníteni a böngészőben csak meghívjuk a *Postable* interface *draw* metódusát.
- A *draw* máshogy működik linkeknél, képeknél, szövegeknél, de ez nem számít. Meg van írva mindegyik osztályban és csak ez a lényeges.

Nagyobb példa

- Nézzük meg a Facebookos példát.
- Az interface-ünk neve *Postable*, az egyetlen metódusa a *draw*, ami egy *String*et ad vissza és nincs bemenete.
- Ezt az interface-t implementáljuk 3 különböző osztályban, majd létrehozunk ilyen típusú objektumokat és egy vektorban tároljuk mindet.
- Majd egyenként meghívjuk a *draw* függvényeiket.

```
public interface Postable {
    String draw();
}

// Uj file

public class FacebookText implements Postable {
    private String content_;

    public FacebookText(String content) {
        content_ = content;
    }

    public String draw() {
        return "<div>" + content_ + "</div>";
    }
}
```

```
public class FacebookImage implements Postable {
    private String location_;

    public FacebookImage(String location) {
        location_ = location;
    }

    public void setLocation(String location) {
        location_ = location;
    }

    public String draw() {
        return "<img src=" + location_ + ">";
    }
}
```

```
public class FacebookLink implements Postable {
    private String location_;
    private String label_;

    public FacebookLink(String location, String label) {
        location_ = location;
        label_ = label;
    }

    public void setLocation(String location) {
        location_ = location;
    }

    public String draw() {
        return "<a href=" + location_ + ">" +
                label_ + "</a>";
    }
}
```

```
import java.util.Vector;

public class FacebookMain {
    public static void main(String[] args) {
        Vector<Postable> posts = new Vector<Postable>();

        posts.add(new FacebookText("Volt egyszer..."));
        posts.add(new FacebookImage("halacska.jpg"));
        posts.add(new FacebookLink("google.com","Google"));
        posts.add(new FacebookText("...egy kiskutya."));

        String html = new String();
        for (Postable post : posts) {
            html += post.draw();
        }

        System.out.println(html);
    }
}
```

Eredmény:

```
<div>Volt egyszer egy kiskutya...</div><img src=
halacska.jpg><a href=http://google.com>Google</a>
<div>...elment a vasarba.</div>
```

Eredmény:

```
<div>Volt egyszer egy kiskutya...</div><img src=
halacska.jpg><a href=http://google.com>Google</a>
<div>...elment a vasarba.</div>
```

- Amit nem tehetek meg, hogy olyan metódust hívok meg amit nem a megadott interface implementál. Pl:

```
for (Postable post : posts) {
    post.setLabel("kutya");
}
```

- Még akkor sem lehet ezt megtenni, ha mind3 osztálynak van setLabel metódusa. Mert mi Postable-ként hivatkozunk rájuk és amíg õket Postable-ként látjuk addig csak olyat tudunk tenni amit biztosan tud *Postable* is.

- Lehet egy osztálynak több interface-e is, és akkor is mindet implementálni kell, ami az interface-ekben van. Pl:

```
public class FacebookText implements Postable,
        Deletable {
    private String content_;

    public FacebookText(String content) {
        content_ = content;
    }

    public String draw() {
        return "<div>" + content_ + "</div>";
    }

}
```