Java tečaj

2b. dio

Apstraktni razredi. Sučelja.

Apstraktni razred

- Pretpostavimo da na raspolaganju imamo razred Slika:
  - public int getSirina() {...}
  - public int getVisina() {...}
  - public void upaliTocku(int x, int y) {...}
  - public void ugasiTocku(int x, int y) {...}

Apstraktni razred

- Kako najlakše postići da se naši geometrijski likovi mogu iscrtavati?
- Proširimo razred GeometrijskiLik s dvije metode:
  - public boolean sadrziTocku(int x, int y) {...}
  - public void popuniLik(Slika slika) {...}

Apstraktni razred

- Zadatak metode sadrziTocku jest utvrditi sadrži li trenutni lik predanu točku (x,y)
- No kako to utvrditi? GeometrijskiLik je bazni razred za svaki lik – informacija koje točke lik sadrži ovdje nije dostupna

Apstraktni razred

- Rješenje 1
  - sadržiTočku uvijek vraća false
  - Svaki razred koji nasljedi GeometrijskiLik **mora** override-ati navedenu metodu kako bi ispravno vraćao informaciju o sadržanim točkama

Apstraktni razred

- Zadatak metode popuniLik jest nacrtati sliku lika
- Nemoguće bez da znamo koji je to lik?
  - Ne! Možda neefikasno, ali ne i nemoguće!
  - Za svaku točku slike pitaj lik sadrži li tu točku, i ako da, upali je!

Apstraktni razred

```
public class GeometrijskiLik {
   ...
   public boolean sadrziTocku(int x, int y) {
       return false;
   }
   public void popuniLik(Slika slika) {
       ...
   }
}
```

Apstraktni razred

```
public class Linija extends GeometrijskiLik {
   ...
   public boolean sadrziTocku(int x, int y) {
        … nova implementacija …
   }
}
```

Apstraktni razred

- Zahvaljujući polimorfizmu, metoda popuniLik će za svaki konkretan lik pozivati njegovu redefiniranu metodu sadrziTocku(…), i uspješno obaviti crtanje

Apstraktni razred – primjer 1

```
public static void main(String[] args) {
  Slika slika = new Slika(40, 40);

  Linija l1 = new Linija(10, 10, 30, 30);
  l1.popuniLik(slika);

  Pravokutnik p = new Pravokutnik(2, 20, 10, 15);
  p.popuniLik(slika);

  Pravokutnik p2 = new Pravokutnik(30, 4, 2, 2);
  p2.popuniLik(slika);

  slika.nacrtajSliku(System.out);
}
```

Apstraktni razred

- Opisana metoda popuniLik je spora!
- Konkretni likovi mogu obaviti redefiniranje i te metode, i dodatno je ubrzati.
- Primjerice
  - Linija: bresenhamov postupak
  - Pravokutnik: dvije ograničene for petlje

Apstraktni razred

- Ponovno rješenje 1
  - sadržiTočku uvijek vraća false
  - Svaki razred koji nasljedi GeometrijskiLik **mora** override-ati navedenu metodu kako bi ispravno vraćao informaciju o sadržanim točkama
  - ----------
  - Kako postići ovaj **mora**?

Apstraktni razred

- Rješenje 2
  - Objektno orijentirana paradigma poznaje pojam apstraktne metode – metoda za koju se definira signatura, ali ne i implementacija:

    ```
    public abstract boolean sadrziTocku(int x, int y);
    ```
  - Razred koji sadrži barem jednu ovakvu metodu također je apstraktan i mora biti definiran kao:

    ```
    public abstract class ImeRazreda {...}
    ```

Apstraktni razred

- Rješenje 2
  - Apstraktni razredi ne mogu se instancirati, jer nisu potpuno definirani
  - Apstraktni razred nužno je naslijediti!
  - Razred koji ga nasljeđuje može definirati sve njegove apstraktne metode, ali i ne mora – tada je i on apstraktan i ne može se instancirati

Apstraktni razred

- Uporaba apstraktnih razreda
  - Ponuditi osnovu za definiranje drugih razreda
  - Nudi implementaciju svih dijeljenih algoritama
  - Metode u kojima se konkretni razredi razlikuju definira apstraktnima, čime ih prisiljava da ih definiraju

Sučelja

- Sučelje možemo poistovjetiti sa potpuno apstraktnim razredom – razredom koji definira niz apstraktnih metoda
- Java ipak razlikuje razred od sučelja
- Sučelje se definira ključnom riječi **interface** (a ne class kao kod razreda)

Sučelja

- Sučelje možemo definirati kao popis metoda koje svaki razred koji ga implementira ima definirane
- Terminološki, razredi se nasljeđuju:
  `class A extends B { }`
  a sučelja implementiraju:
  `class A implements I { }`

Sučelja

- Za razliku od modela nasljeđivanja gdje razred može imati samo jednog roditelja, Java razredima dozvoljava da implementiraju proizvoljan broj sučelja:

```
class A implements I1, I2, I3
{...}
```

Sučelja

- Sučelja i nasljeđivanje se međusobno ne isključuju! Primjerice, ispravno je napisati:

```
class B extends A implements I1, I2, I3 {...}
```

Sučelja

- Kako najlakše postići da se naši geometrijski likovi mogu iscrtavati?
- Umjesto da proširimo razred GeometrijskiLik s dvije metode:
  - public boolean sadrziTocku(int x, int y) {...}
  - public void popuniLik(Slika slika) {...}

Sučelja

- Napravimo sljedeće: definirajmo sučelje SadrziocTocaka:

```
interface SadrziocTocaka {
  public boolean sadrziTocku(int x, int y);
}
```

Sučelja

- Za svaki konkretan lik recimo da implementira sučelje SadrziocTocaka, i definirajmo potrebne metode; npr.:

```
class Linija extends GeometrijskiLik
   implements SadrziocTocaka {
  public boolean sadrziTocku(int x, int y)
   {…}
}
```

Sučelja

- Za svaki konkretan lik recimo da implementira sučelje SadrziocTocaka, i definirajmo potrebne metode; npr.:

```
class Pravokutnik extends GeometrijskiLik
   implements SadrziocTocaka {
  public boolean sadrziTocku(int x, int y)
   {…}
}
```

Sučelja

- Razred Kvadrat ne mora implementirati SadrziocTocaka – on to čini samom činjenicom da nasljeđuje razred Pravokutnik
- Može redefinirati potrebne metode zbog brzine izvođenja (ako ih može implementirati brže)

Sučelja

- Metodu za crtanje prebacimo u razred Slika:

```
class Slika {
    ….
  public void popuniLik(SadrziocTocaka lik) {
    for(int y=0; y<visina; y++) {
      for(int x=0; x<sirina; x++) {
        if(lik.sadrziTocku(x, y)) {
          upaliTocku(x, y);
        }
      }
    }
  }
}
```

Apstraktni razred – primjer 2

```
public static void main(String[] args) {
  Slika slika = new Slika(40, 40);

  Linija l1 = new Linija(10, 10, 30, 30);
  slika.popuniLik(l1);

  Pravokutnik p = new Pravokutnik(2, 20, 10, 15);
  slika.popuniLik(p);

  Pravokutnik p2 = new Pravokutnik(30, 4, 2, 2);
  slika.popuniLik(p2);

  slika.nacrtajSliku(System.out);
}
```

Sučelja vs apstraktni razredi

- Apstraktni razred je – razred
  - Dobro rješenje za definirati osnovu za izvođenje novih razreda
  - Može ponuditi implementaciju zajedničkih algoritama

Sučelja vs apstraktni razredi

- Sučelje je – “popis”!
  - Dobro rješenje za dodavanje “karakteristika” postojećim razredima
  - Funkcionira neovisno o strukturi nasljeđivanja: razred koji već ima definiranu strukturu nasljeđivanja može implementirati proizvoljna sučelja

Sučelja vs apstraktni razredi

- Sučelje je – “popis”!
  - Izuzetno često korišteni u Javi
  - Susrest ćemo se s njima uskoro kod Collection Frameworka i Swinga
  - Omogućava razdvajanje implementacije i “obećane” funkcionalnosti
  - Može se shvatiti i kao “pogled” kroz koji se vidi neki objekt