Univerzitet u Beogradu 29.01.2015.
Mašinski fakultet

Drugi kolokvijum iz Objektno orijentisanog programiranja i Java-e

1. Implementirati klasu

```
CurrencyConverter
```

sa privatnim atributom `conversionRates`, tipa `HashMap<String,Double>`, koji implementira uredjene parove tipa (`String`,`Double`) , gde je prva komponenta naziv valute a druga je odnos vrednosti valute prema vrednosti dinara. Obezbediti da konstruktor klase `CurrencyConverter` ima jedan argument tipa `HashMap<String,Double>` koji inicijalizuje atribut `conversionRates`. Implementirati metodu

```
void updateConversionRates(String currency, Double rate)
```

koja unosi u atribut `conversionRates` par (`currency`,`rate`). Implementirati metodu

```
double getConversionRate(String currency)
```

koja za zadatu vrednost `currency` vraća vrednost odnosa vrednosti valute `currency` prema vrednosti dinara. Implementirati metodu

```
void setConversionRates(HashMap<String,Double>)
```

koja postavlja novu vrednost atributa `conversionRates`.

U klasi `A` implementirati metodu

```
public static void main(String arg[])
```

kreirati promenljivu tipa `HashMap<String,Double>`, dodati parove (`"dollars"`, 100), (`"euros"`,120) i (`"pounds"`, 130) i kreirati objekat klase `CurrencyConverter` sa argumentom koji je objekat klase `HashMap<String, Double>`.

Rešenje. Sledeća implementacija, u fileu `CurrencyConverter.java`, odgovara uslovima zadatka

```java
package zadatak1

public class CurrencyConverter {

    private HashMap<String, Double> conversionRates;

    public CurrencyConverter(HashMap<String, Double> hm) {
        conversionRates = hm;
    }
    public void updateConversionRates(String currency, Double
    rate) {
        conversionRates.put(currency, rate);
    }
    public double getConversionRate(String currency) {
        return conversionRates.get(currency).doubleValue();
    }
    public void setConversionRates(HashMap<String, Double> hm) {
        conversionRates = hm;
    }
}
```

File `A.java`

```java
package zadatak1

public class A {
    public static void main(String argc[]) {
        HashMap<String, Double> hm = new HashMap<>();
        hm.put("dollars",new Double(100));
        hm.put("euros", new Double(120));
        hm.put("pounds", new Double(130));

        CurrencyConverter cC = new CurrencyConverter(hm);
    }
}
```

2. U klasi `CurrencyConverter` implementirati unutrašnju klasu `Currency`, privatnog pristupa, koja ima zaštićeni atribut `name` tipa `String`. U klasi `CurrencyConverter` definisati

`interface Conversion`

sa javnim pristupom, koji ima metodu `convertToDinars` koja prihvata argument tipa `double` i vraća argument tipa `double`. Implementirati unutrašnje klase, klase `CurrencyConverter`, sa imenima `Dollars`, `Euros` i `Pounds`, sa javnim pristupom, koje implementiraju `interface Conversion` i koje nasledjuju klasu `Currency`. Obezbediti da unutrašnje klase `Dollars`, `Euros` i `Pounds` u svom konstruktoru postavljaju vrednost atributa `name` klase `Currency` na vrednosti `"dollars"`, `"euros"` i `"pounds"`, redom.
U klasi `A` implementirati metodu

```
public static void main(String[] argc)
```

koja kreira objekte klasa `Dollars`, `Euros` i `Pounds` i pozvati metode `converToDinars`.

Rešenje. Sledeća implementacija, u fileu CurrencyConverter.java!, odgovara uslovima zadatka

```
package zadatak2

public class CurrencyConverter {
    private class Currency {
        protected String name;
    }
    public interface Conversion {
        public double convertToDinars(double amount);
    }
    public class Dollars extends Currency implements Conversion
    {
        public Dollars() {
            name = "dollars";
        }
        @Override
        public double convertToDinars(double amount) {
            return 1*amount;
        }
    }
    public class Euros extends Currency implements Conversion {
        public Euros() {
            name = "euros";
        }
        @Override
        public double convertToDinars(double amount) {
            return 1*amount;
        }
    }
    public class Pounds extends Currency implements Conversion {
```

```
        public Pounds() {
30          name = "pounds";
        }
32      @Override
        public double convertToDinars(double amount) {
34          return 1*amount;
        }
36  }
```

File `A.java`

```
package zadatak3;
2
public class A {
4   public static void main(String[] argc) {
        CurrencyConverter cC = new CurrencyConverter();
6       CurrencyConverter.Dollars dollars = cC.new Dollars();
        CurrencyConverter.Euros euros = cC.new Euros();
8       CurrencyConverter.Pounds pounds = cC.new Pounds();

10      dollars.convertToDinars(1);
        euros.convertToDinars(1);
12      pounds.convertToDinars(1);
    }
14  }
```

3. Ako ste rešili prva dva zadatka integrišite rešenja u jedinstvenu klasu `CurrencyConverter` sa sledećim dodacima. Metode

```
convertToDinars(double amount)
```

vrše konverziju na taj način što množe vrednost `amount` sa vrednošću koja se dobija iz atributa `conversionRates` za vrednost atributa `name`. Obezbediti da metoda `getConversionRate`, klase `CurrencyConverter`, generiše izuzetak `NoCurrencyConversionRate` ako je metoda pozvana sa vrednošću argumetna `currency` koji se ne nalazi u atributu `conversionRates`. Obezbediti da izuzetak

```
NoCurrencyConversionRate
```

ima konstruktor koji prihvata argument tipa `String` koji prihvata vrednost tipa `String` čija se vrednost ne nalazi u atributu `conversionRates`. Obezbediti da metode `convertToDinars` unutrašnjih klasa `Dollars`, `Euros` i `Pounds` generišu izuzetke

```
NoDollarsConversionRate
NoEurosConversionRate
NoPoundsConversionRate
```

redom, u slučaju da atribut `conversionRates` nema odgovarajuće vrednosti. Izuzeci

```
NoDollarsConversionRate
NoEurosConversionRate
NoPoundsConversionRate
```

su podklase klase `NoCurrencyConversionRate`, koje imaju konstruktore bez argumenata i koje prosleđuju konstruktoru klase `NoCurrencyConversionRate` vrednosti identične vrednostima atributa `name` odgovarajućih klasa.

Rešenje. Sledeća implementacija, u fileu `CurrencyConverter.java`, zadovoljava uslove zadatka

```
package zadatak3;

public class CurrencyConverter {

    private HashMap<String, Double> conversionRates;

    public CurrencyConverter(HashMap<String, Double> hm) {
        conversionRates = hm;
    }
    public void updateConversionRates(String currency, Double
    rate) {
        conversionRates.put(currency, rate);
    }
    public double getConversionRate(String currency) throws
    NoCurrencyConversionRate {
        Double d = conversionRates.get(currency);
        if( d == null )
            throw new NoCurrencyConversionRate(currency);
        return d;
    }
    public void setConversionRates(HashMap<String, Double> hm) {
        conversionRates = hm;
```

```
    }

    private class Currency {
        protected String name;
    }
    public interface Conversion {
        public double convertToDinars(double amount) throws
    NoCurrencyConversionRate;
    }
    public class Dollars extends Currency implements Conversion
    {
        public Dollars() {
            name = "dollars";
        }
        @Override
        public double convertToDinars(double amount) throws
    NoDollarsConversionRate  {
            Double d = conversionRates.get("dollars");
            if( d == null )
                throw new NoDollarsConversionRate();
            return d*amount;
        }
    }
    public class Euros extends Currency implements Conversion {
        public Euros() {
            name = "euros";
        }
        @Override
        public double convertToDinars(double amount) throws
    NoEurosConversionRate {
            Double d = conversionRates.get("euros");
            if( d == null )
                throw new NoEurosConversionRate();
            return d*amount;
        }
    }
    public class Pounds extends Currency implements Conversion {
        public Pounds() {
            name = "pounds";
        }
        @Override
        public double convertToDinars(double amount) throws
    NoPoundsConversionRate {
            Double d = conversionRates.get("pounds");
```

```
            if( d == null )
                throw new NoPoundsConversionRate();
            return d*amount;
        }
    }
}
```

File `NoCurrencyConversionRate.java`

```
package zadatak3;

public class NoCurrencyConversionRate extends Exception {

    private String name;
    public NoCurrencyConversionRate(String name) {
        this.name = name;
    }
}
```

File `NoDollarsConversionRate.java`

```
package zadatak3;

public class NoDollarsConversionRate extends
    NoCurrencyConversionRate {

    public NoDollarsConversionRate() {
        super("dollars");
    }
}
```

File `NoEurosConversionRate.java`

```
package zadatak3;

public class NoEurosConversionRate extends
    NoCurrencyConversionRate {

    public NoEurosConversionRate() {
        super("euros");
    }
}
```

File `NoPoundsConversionRate.java`

```
package zadatak3;

public class NoPoundsConversionRate extends
    NoCurrencyConversionRate {

    public NoPoundsConversionRate() {
        super("pounds");
    }
}
```

3. Ako niste rešili prvi i drugi zadatak objasnite sve konstruktore klase `Exception`.

Rešenje. Svi javni konstruktori klase `Exception` su

```
Exception()
Exception(String mesage)
Exception(String message, Throwable cause)
Exception(Throwable cause)
```

Treći konstruktor prihvata `String` i objekat klase `Throwable`. Prvi argument je poruka koju vraća objekat klase `Exception` metodom `getMessage()`, Drugi argument je objekat koji se označava kao uzrok izuzetka, i koji se dobija pozivom metode `Throwable getCause()`. Treći konstruktor postavlja argument klase `Throwable` kao uzor, dok je vrednost poruke, koja se dobija pozivom metode `\getMessage`, postavljena na vrednost koju vraća metoda `toString()` objekta `cause`. Drugi konstruktor prihvata vrednost poruke, koja se kasnije može dobiti metodom `getMessage()`. Prvi konstruktor ne postavlja ni poruku ni uzor.