Objektno orijentisano programiranje 2

Zrna Jave

Uvod

- Zrna Jave (*JavaBeans*)
  - softverske komponente pisane na Javi
- Komponenta
  - fizički celovit i nezavisno isporučiv deo sistema
- Analogija
  - Java – kafa,
  - Bean – zrno, komponenta na Javi je "zrno kafe"
- Zrno može da bude vidljivo (komponenta korisničkog interfejsa), ali ne mora (tajmer)
- Osnovna ideja
  - da se zrna mogu konfigurisati i povezivati kroz okruženja za razvoj
- Alat za rad sa zrnima
  - mora posedovati mogućnosti za introspekciju zrna
  - potrebno je da prikaže svojstva, događaje i javne metode

Arhive

- Zrna se isporučuju u JAR (Java ARchive) fajlovima
- Analogija
  - Java Beans – zrna kafe
  - JAR – tegla u kojoj se drže zrna kafe
- U arhivama se čuvaju
  - `class` fajlovi
  - resursni fajlovi (slike, podaci,...) i
  - manifest
    - tekst datoteka u kojoj se navodi spisak datoteka koje obrazuju arhivu
    - ako je neka `class` datoteka zrno – u redu ispod se navodi: `Java-Bean: True`
- JAR koristi kompresiju istu kao i ZIP
  - sadržaj se može pregledati sa WinZip, WinRAR

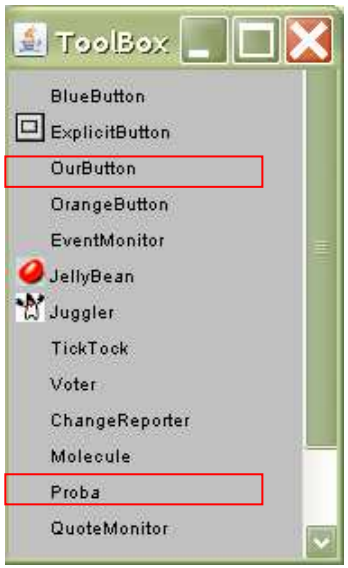
Alat za rad sa arhivama

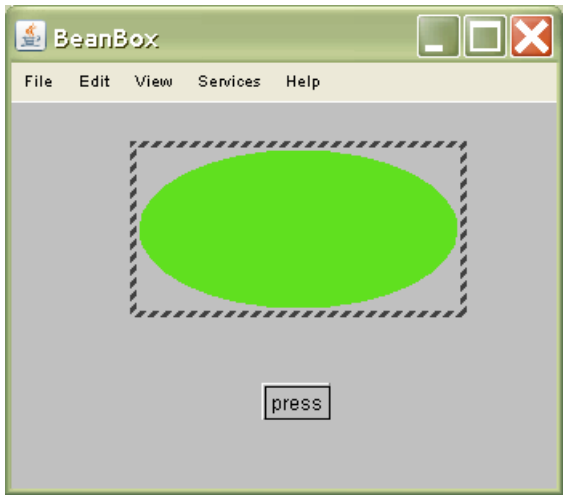
- JAR je i naziv alata za rad sa arhivama
- Sintaksa poziva:
  - `jar {ctxu}[fm0] <arhiva><manifest>[-C dir]<fajlovi>`
- Komande:
  `c` – formiranje nove arhive
  `t` – prikazuje sadržaj arhive
  `x` – raspakivanje arhive
  `u` – ažurira se postojeća arhiva (dodaju se fajlovi)
- Opcije (važnije):
  `f` – pozicija odredišne `*.jar` arhive u listi fajlova (prvi fajl za redosled `fm`)
  `m` – pozicija manifesta u listi fajlova (drugi fajl za redosled `fm`)
  `0` – ne koristi se kompresija pri arhiviranju
  `C` – menja se folder i uključuju odgovarajući fajlovi u arhivu
- Primer:
  `jar cfm a.jar m.txt f/*.*`
  - formira se arhiva `a.jar` na osnovu manifesta `m.txt` i svih fajlova iz foldera `f/`

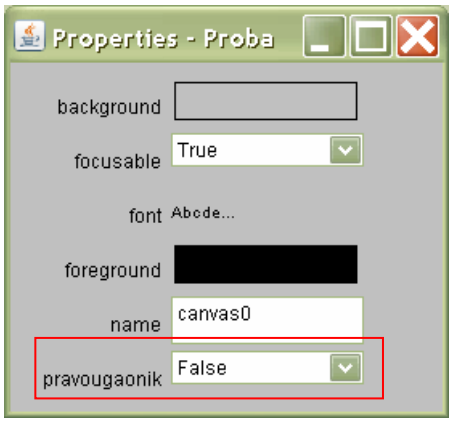
Alat za razvoj zrna

- BDK (*Bean Developer Kit*) – Sun Microsystems (JavaSoft)
  - alat za razvoj zrna Jave (ne održava se, dobar za akademske potrebe)
- Alat omogućava da se
  - napravi,
  - podesi i
  - poveže grupa zrna
- Prozori:
  - `ToolBox` – prozor sa listom zrna Jave
  - `BeanBox` – prozor sa kontejnerom u kojem se konfigurišu i povezuju zrna Jave
  - `Properties` – prozor za podešavanje svojstava (*properties*) zrna Jave
- Postupak:
  - iz `ToolBox` prozora se bira zrno koje se pozicionira u `BeanBox` prozoru
  - za selektovano zrno u `BeanBox` prozoru prikazuju se svojstva u `Properties`
  - u `BeanBox` prozoru se selektuju i izvor i odredište događaja uz pomoć vizuelnih alata
- Drugi alati:
  - NetBeans (8.0), tutorijal (http://docs.oracle.com/javase/tutorial/javabeans/)
  - Bean Builder – Java.net projekat (https://java.net/projects/bean-builder)

BDK – korisnički interfejs







Svojstva i metodi zrna

- Svojstvo (*property*) je element za opis stanja objekta
  - ima pridružen atribut (može biti i niz) koji čuva stanje
  - izvan klase se vidi preko specifičnih metoda za postavljanje/čitanje (`set/get`)
  - svojstvo može biti i takvo da se
    - samo čita (samo `get`) ili
    - samo upisuje (samo `set`)
- Metodi
  - metodi koje zrno ispoljava moraju biti javni (privatni i zaštićeni se ne ispoljavaju)
  - nema posebnog načina imenovanja (ako nisu svojstva)
  - mehanizam introspekcije pronalazi javne metode
- Vrste svojstava:
  - jednostavna svojstva (jedna vrednost proizvoljnog tipa)
  - logička svojstva (jedna logička vrednost)
  - indeksna svojstva (više vrednosti)

Jednostavna (skalarna) svojstva

- Klasa sadrži sledeće metode za neko jednostavno svojstvo:

```
public T getN();
public void setN(T argument);
```

  - gde su `T` – proizvoljan tip, a `N` proizvoljno ime svojstva
- Primer:

```
public class Pravougaonik{
  private double sirina, visina;
  public double getSirina(){return sirina;}
  public void setSirina(double s){sirina=s;}
  public double getVisina(){return visina;}
  public void setVisina(double v){visina=v;}
}
```

Logička svojstva

- Klasa sadrži sledeće metode za neko logičko svojstvo:

```
public boolean isN();
public boolean getN();
public void setN(boolean argument);
```

  - gde je `N` proizvoljno ime svojstva
- Primer:

```
public class PopunjenaFigura{
  private boolean popunjena;
  public boolean isPopunjena(){return popunjena;}
  public void setPopunjena(boolean p){popunjena=p;}
}
```

Indeksna (vektorska) svojstva

- Klasa sadrži sledeće metode za neko indeksno svojstvo:

```
public T getN(int indeks);
public void setN(int indeks, T vrednost);
public T[] getN();
public void setN(T[] vrednost);
```

  - gde su `T` – proizvoljan tip elementa svojstva, a `N` proizvoljno ime svojstva
- Primer:

```
public class Brojevi{
  private double[] niz;
  public double getNiz(int i){return niz[i];}
  public void setNiz(int i, double b){niz[i]=b;}
  public double[] getNiz(){return niz;}
  public void setNiz(double[] b){niz=b;}
}
```

  - u primeru se pretpostavlja da objekat klase `Brojevi` nije odgovoran za životni vek niza brojeva (sadrži niz samo po referenci)

Događaji

- Zrna poštuju model delegirane obrade događaja
- Zrno može biti izvor događaja ili osluškivač (i obrađivač) događaja
- Neki izvori omogućavaju više, a neki samo jednog osluškivača
- Da bi se prijavilo kao jedan od osluškivača zrno poziva metod izvora:

```
public void addTListener(TListener prijemnik);
```

- Za izvor koji podržava samo jednog osluškivača metod baca izuzetak:

```
public void addTListener(TListener prijemnik)
                                    throws TooManuListeners;
```

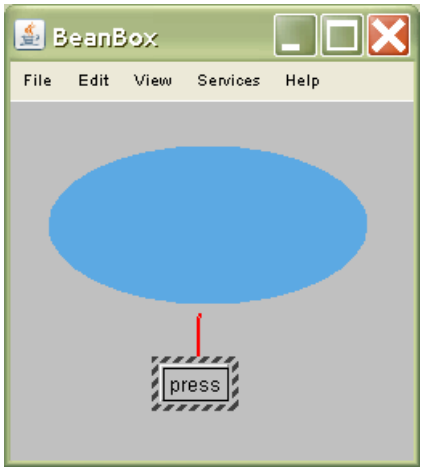
- Da bi se odjavilo zrno osluškivača poziva metod izvora:

```
public void removeTListener(TListener prijemnik);
```

- U nazivima metoda i tipova osluškivača, *T* je vrsta događaja (npr. `Action`)
- Alati omogućavaju:
  - manuelno povezivanje izvora sa osluškivačem kroz GUI
  - automatsko generisanje obrade događaja na strani osluškivača

Obrada događaja

- Alat (BDK) oslobađa programera pisanja metoda za obradu događaja
  - selektuje se zrno izvor događaja
  - iz menija (`Edit/Events`) se izabere vrsta događaja (`Action`) i metod za obradu (`actionPerformed`)
  - pojavi se razvlačeća (*rubber-band*) linija koja se vuče do zrna koje obrađuje događaj
  - iz dijaloga se izabere javni metod koji treba da bude pozvan iz metoda za obradu događaja
  - metod za obradu događaja se automatski generiše



Primer - 1

```
package sunw.demo.proba;
import java.awt.*;
import java.awt.event.*;

public class Proba extends Canvas {
  private Color boja;
  private boolean pravougaonik;

  public Proba(){
    addMouseListener(new MouseAdapter(){
      public void mousePressed(MouseEvent me){promeniBoju();}
    });
    pravougaonik=false; setSize(200,100); promeniBoju();
  }

  public boolean getPravougaonik(){return pravougaonik;}
  public void setPravougaonik(boolean p){pravougaonik=p; repaint();}
```

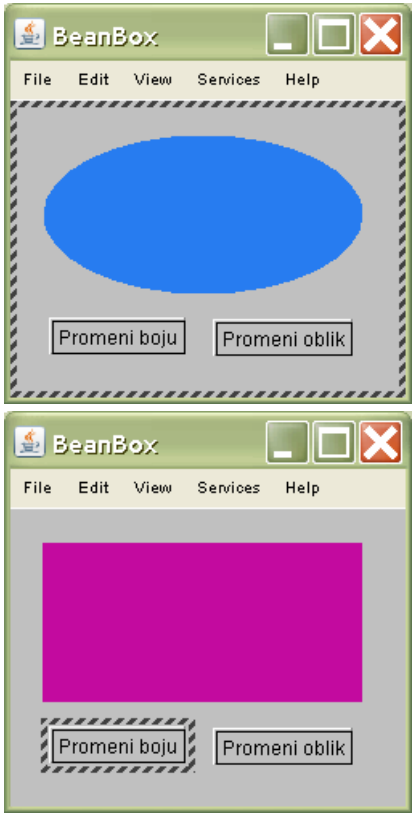
Primer - 2

```
  public void promeniBoju(){boja=slucajnaBoja(); repaint();}
  public void promeniOblik(){pravougaonik=!pravougaonik; repaint();}

  private Color slucajnaBoja(){
    int r=(int)(256*Math.random());
    int g=(int)(256*Math.random());
    int b=(int)(256*Math.random());
    return new Color(r,g,b);
  }
  public void paint(Graphics g){
    Dimension d=getSize();
    g.setColor(boja);
    if (pravougaonik){
      g.fillRect(0,0,d.width-1,d.height-1);
    } else {
      g.fillOval(0,0,d.width-1,d.height-1);
    }
  }
}
```

Primer - povezivanje

- Kreira se fajl `proba.mft`:
  ```
  Name: sunw/demo/proba/Proba.class
  Java-Bean: True
  ```
- Prevede se Proba.java
  ```
  javac proba.java
  jar cfm proba.jar proba.mft
                  sunw/demo/proba/*.class
  ```
- Iz *ToolBox*-a se izaberu zrna
  ```
  OurButton (2 puta)
  Proba
  ```
- Promene se natpisi na dugmadima
  - kroz prozor *Properties*
- Povežu se `actionPerformed` metode dugmadi sa odgovarajućim javnim metodama klase `Proba`



Perzistencija

- Perzistencija – sposobnost snimanja stanja zrna u trajnoj memoriji
- Moguće je kasnije učitavanje uz restauraciju stanja
- Snimak obuhvata vrednosti svojstava i polja objekta
- Perzistencija se obezbeđuje mehanizmom serijalizacije objekta
- Zrno se najjednostavnije serijalizuje ako implementira interfejs
  - `java.io.Serializable`
- Ograničenje: klasa mora da ima konstruktor bez argumenata
- Polja označena ključnom reči `transient` se ne snimaju pri serijalizaciji
- U prethodnom primeru:

```
import java.io.Serializable;
public class Proba extends Canvas implements Serializable{
  transient private Color boja;          //nije potrebna perzistencija
  private boolean pravougaonik;          //potrebna perzistencija
```

Introspekcija

- JavaBeans tehnologija se zasniva na mehanizmima introspekcije
- Dva načina introspekcije:
  - posebno imenovanje svojstava – svojstva se otkrivaju na osnovu imena `get/set`
  - implementacija interfejsa `BeanInfo` kojom se eksplicitno određuju vidljivi elementi zrna
- Interfejs `BeanInfo` definiše metode za dohvatanje svojstava, događaja i metoda:

```
PropertyDescriptor[] getPropertyDescriptors()
EventSetDescriptor[] getEventSetDescriptors()
MethodDescriptor[] getMethodDescriptors()
```

- Klase *xy*`Descriptor` su izvedene iz klase `FeatureDescriptor`
- Interfejs `BeanInfo` i klase *xy*`Descriptor` su iz paketa `java.beans`
- Definisanjem gornjih metoda programer tačno definiše elemente koje zrno ispoljava
- Klasa koja implementira `BeanInfo` treba da se imenuje *`ImeZrna`*`BeanInfo`
- Podrazumevano ispoljavanje elemenata definiše klasa `SimpleBeanInfo`
- Izvođenjem iz ove klase i redefinisanjem navedenih metoda eksplicitno se određuje šta zrno ispoljava (ako podrazumevane metode ne odgovaraju)

Primer `BeanInfo` klase

- Implementira se klasa `ProbaBeanInfo` za (prethodno napisano) zrno `Proba`

```
import java.beans.*;
public class ProbaBeanInfo extends SimpleBeanInfo{
  public PropertyDescriptor[] getPropertyDescriptors(){
    try {
      PropertyDescriptor pravougaono =
        new PropertyDescriptor("Pravougaonik", Proba.class);
      PropertyDescriptor svojstva[] = {pravougaono};
      return svojstva;
    } catch (Exception e){}
    return null;
  }
}
```

Primer klase za introspekciju

```
import java.beans.*;
public class PrimerIntrospekcije{
  public static void main(String arg[]){
    try{
      Class c=Class.forName("Proba");
      BeanInfo bI = Introspector.getBeanInfo(c);
      System.out.println("Svojstva:");
      PropertyDescriptor pD[]=bI.getPropertyDescriptors();
      for(int i=0; i<pD.length; i++)
        System.out.println("\t"+pD[i].getName());
    } catch(Exception e){System.out.println("Izuzetak"+e);}
  }
}
```