13. Swing API obrada događaja

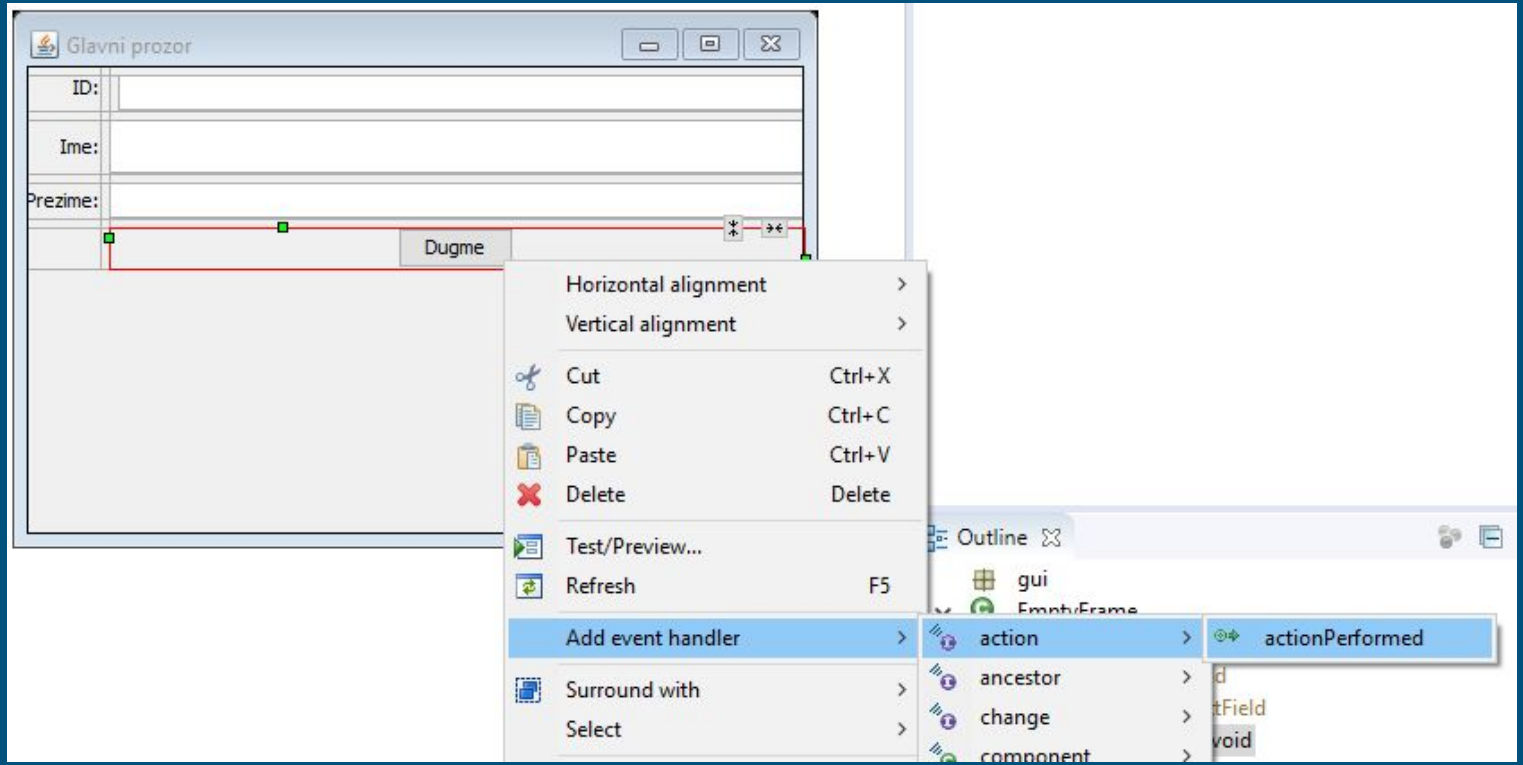
Sadržaj

- Osluškivači događaja
- Anonimne unutrašnje klase
- Obrada događaja
- Isporučivanje aplikacije

Model obrade događaja

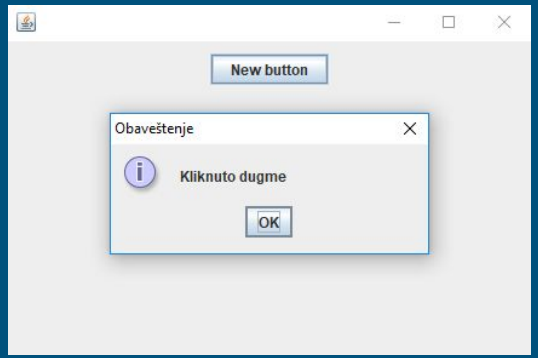
- Izvor događaja - neka komponenta nad kojom se desi događaj
  - **klik** na **dugme**
  - **izmena teksta** u **tekst polju**
  - **pomeranje pokazivača miša** nad **panelom**
- Osluškivač (engl. *listener*)
  - objekat čija klasa sadrži metodu koja se izvršava na pojavu događaja

Dodavanje *listener*-a za klik na dugme



Interface ActionListener

```
…
btnNewButton.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent arg0) {
    btnNewButtonActionPerformed();
  }
});
…
protected void btnNewButtonActionPerformed() {
        JOptionPane.showMessageDialog(this,
                                      "Kliknuto dugme",
                                      "Obave\u0161tenje",
                                      JOptionPane.INFORMATION_MESSAGE);
}
```



Klasa `JOptionPane`

- Obezbeđuje statičke metode za prikaz predefinisanih dijaloga
- `public static void showMessageDialog` prikazuje dijalog sa porukom i odgovarajućom ikonicom, parametri
  - `Component parent`
  - `String message`
  - `String title`
  - `int messageType`
- `public static int showOptionDialog()` - vraća izabranu opciju iz prosleđenog `Object[]`
- `public static String showInputDialog()` - vraća unos korisnika
- `public static int showConfirmDialog()` - vraća izabranu opciju

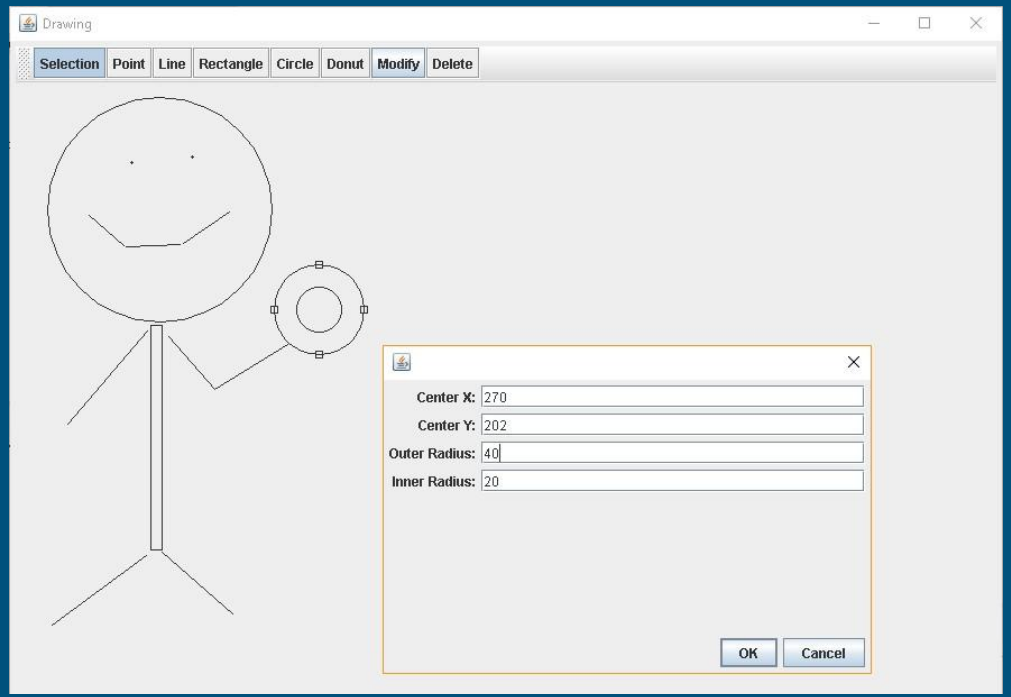
Obrada događaja miša

```
panel.addMouseListener(new MouseAdapter() {
  @Override
  public void mouseClicked(MouseEvent arg0) {
    ...
  }
});
```

- Interfejs `MouseListener` definiše pet metoda
- Klasa `MouseAdapter` implementira svih pet kao prazne, što omogućava da redefinišemo samo onu koja nam je od interesa

Studija slučaja i događaj klika mišem

- Panel “sluša” klik mišem
- Objekat `MouseEvent` sadrži poziciju klika (ima metode `getX` i `getY`)
- Aplikacija na osnovu utisnutog `JToggleButton` dugmeta (`isSelected` vraća `true`) zna koju operaciju da izvrši



Isporučivanje aplikacije

- Pakovanje svih potrebnih klasa u .jar fajl
- Za računare sa Windows-om pravljenje .bat fajla
- Podešavanje prečice za pokretanje programa

Pravci daljeg rada

- Redizajn aplikacije saglasno Model View Controller obrascu
  - *model* sadrži podatke (oblike)
  - *view* prikazuje podatke i “sluša” interakciju korisnika
  - *controller* reaguje na interakciju korisnika, menja *model* i inicira osvežavanje *view*-a
- Uobičajene funkcionalnosti aplikacije koje zasad nisu podržane
  - boje
  - višestruka selekcija
  - z-order
  - perzistencija
    - serijalizacija
    - baza podataka
  - undo - redo funkcionalnost

Undo - redo funkcionalnost

- Dizajnerski obrazac *Command*

```
public interface Command {
  void execute();
  void unexecute();
}
```

- Svaka komanda je klasa koja implementira interfejs `Command`
- Svaki objekat klase komande je konkretna komanda
  - `execute` izvršava komandu - obično modifikuje *model*
  - `unexecute` poništava komandu - vraća model u stanje pre izvršenja `execute`

Primer klase komande

```
public class AddPoint implements Command {
  private Model model;
  private Point point;
  public AddPoint(Model model, Point point) {
    this.model = model;
    this.point = point;
  }
  public void execute() {
    model.add(point);
  }
  public void unexecute() {
    model.remove(point);
  }
}
```

Rezime

- Događaji koje korisnik generiše potrebno je obraditi odgovarajućim metodama, saglasno Swing API arhitekturi
- Klasa `JOptionPane` obezbeđuje prikaz predefinisanih dijaloga
- Isporučivanje aplikacije podrazumeva smeštanje potrebnih klasa u .jar biblioteku
- Dizajn aplikacije mogao bi se unaprediti primenom principa objektno-orijentisanog dizajna