# OBJEKTNO PROGRAMIRANJE 2

Oznaka predmeta: OP2

Predavanje broj: 11

Nastavna jedinica: JAVA

Nastavne teme:

JavaFX (Application, Stage, Scene, Node). Layout klase. Paket Geometry (Pos, Orientation, Insets). Klase: Color, Cursor, Font. Upravljanje događajima (interfejs EventHandler: implementacija korišćenjem glavne klase, unutrašnje klase, anonimne klase, lambda izraza). Smenjivanje scena. Kreiranje klasa: MessageBox, ConfirmationBox). TextField, RadioButton, CheckBox.

Predavač: prof. dr Perica S. Štrbac, dipl. ing.

Literatura:

Eckel B., *Thinking in Java*, 2nd edition, Prentice-Hall, New Jersey 2000.

Cay S. Horstmann and Gary Cornell*: "Core Java, Advanced Features", Vol. 2, Prantice Hall, 2013.*

*The Java Tutorial*, Sun Microsystems 2001. *http://java.sun.com*

Branko Milosavljević, Vidaković M, *Java i Internet programiranje*, GInT, Novi Sad 2002.

# JavaFX

- JavaFX predstavlja čist Java API (Swing je bio u kombinaciji sa awt)
- Ideja je bila da se napravi alat za GUI razvoj Internet aplikacija (RIA – rich Internet applications)
- JavaFX je Java biblioteka predviđena za razne tipove uređaja (mobilni telefoni, pametni telefoni, računari, tablet računari itd) pri čemu su obuhvaćeni: audio, video, animacija i grafika.
- Paradigma JavaFXa je analogna pozorištu:
  - postoji pozornica (javafx.stage.Stage)
  - na pozornicu se može postaviti scena (javafx.scene.Scene)
  - scena ima svoj raspoređivač komponenti
  - na scenu se postavljaju komponente
- JavaFX aplikacija proširuje klasu javafx.application.Application
- Startovanje JavaFX aplikacije kreće kada se iz metode main pozove metoda *launch(String[])* sa argumentima iz komandne linije.
- Nakon ovoga će se automatski pozvati metoda *start(Stage)* čiji je parametar referenca na objekat klase javafx.stage.Stage (dobijena je pozornica).

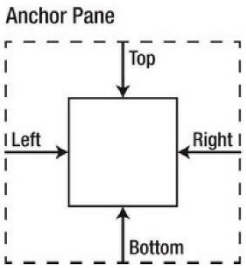
# JavaFX

- Graf scene je drvo sa korenim čvorom (root) te čvorovima-listovima i čvorovima-granama.
- Klasa javafx.scene.Node je bazna klasa za predstavljanje čvorova grafa scene.
  - Najčešće se koriste čvorovi kontrole ulaza/izlaza ili Shape objekti.
- Klasa javafx.scene.Parent je bazna klasa za čvorove grafa scene koji mogu imati decu koja upravlja svim hijerarhijskim operacijama grafa scene (dodavanje/uklanjanje dece čvorova...)
- Čvorovi-grane su tipa javafx.scene.Parent čije su podklase:
  - Group,
  - Region,
  - Control.
- Čvorovi-listovi su specifično tipa:
  - Rectangle, Text, ImageView, MediaView, ili bilo koje klase koja nema decu.
- Nakon dodavanja čvorova, scena se postavlja na pozornicu.
  - Korišćenjem metoda show (kao i ranije) prikazuje se JavaFX prozor.

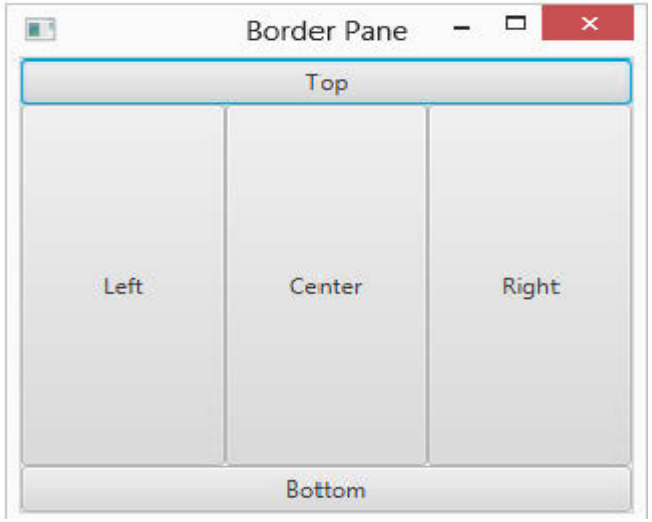
# Layout klase

- JavaFX sadrži niz layout klasa (u paketu javafx.scene.layout):
  - AnchorPane
  - BorderPane
  - FlowPane
  - GridPane
  - HBox
  - StackPane
  - TilePane
  - VBox











- Layout pane određuje poziciju i veličinu svih čvorova na sceni:
  - Menjanjem veličine prozora automatski menja veličinu svojih čvorova.

# Layout klase

- HBox čvorove-decu raspoređuej u red.
  - Veličina čvorova dece se povećava do njihove željene širine (preferred widths), ali postoji mogućnost da se zada da se pojedini čvorovi-deca povećavaju do njihove maksimalne zadate širine.
- VBox čvorove-decu raspoređuje u kolonu.
  - Padding property upravlja razmakom među čvorovima.
  - Moguće je podešavanje margina oko čvorova.
- FlowPane smešta čvorove-decu u redove tako što nakon popunjenog jednog reda prelazi u drugi.
  - Podrazumevana pozicija reda sa komponentama je centar kontejnera, a podrazumevana orijentacija je sa leva na desno.
  - Može se zadati podrazumevani razmak među komponentama.
  - Veličina čvorova-dece se ne menja.
  - Može se zadati i orijentacija.

# Layout klase

- Konstruktori klase FlowPane:

```
FlowPane()
```

kreira horizontalni FlowPane sa horizontalnim i vertikalnim razmakom 0 (hgap/vgap = 0).

```
FlowPane(double hgap, double vgap)
```

kreira horizontalni FlowPane sa zadatim hgap i vgap.

```
FlowPane(Orientation orientation)
```

kreira FlowPane zadate orijentacije sa horizontalnim i vertikalnim razmakom 0 (hgap/vgap = 0)

```
FlowPane(Orientation orientation, double hgap, double vgap)
```

kreira FlowPane zadate orijentacije sa zadatim hgap i vgap.

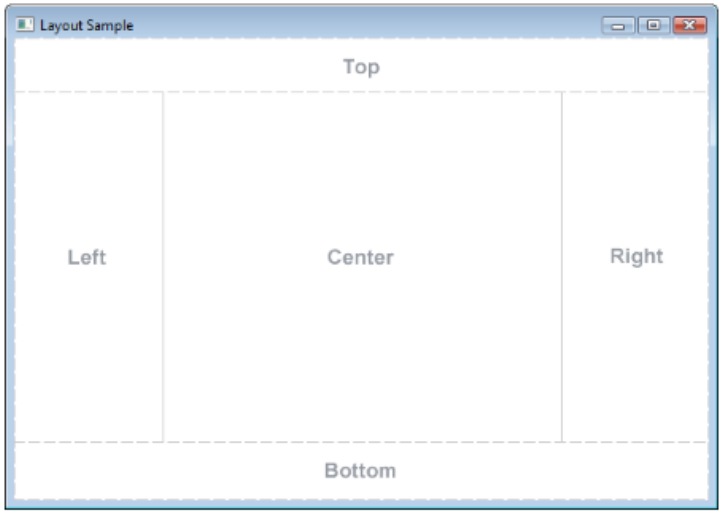
```
Image images[] = { ... };
FlowPane flow = new FlowPane(); //horizontalni FlowPane
flow.setVgap(8);
flow.setHgap(4);
flow.setPrefWrapLength(300); // preferred width = 300
for (int i = 0; i < images.length; i++) {
  flow.getChildren().add(new ImageView(images[i]);
}
```

# Layout klase

```
FlowPane flow = new FlowPane(Orientation.VERTICAL);
flow.setColumnHalignment(HPos.LEFT); // levo poravnanje elemenata
flow.setPrefWrapLength(200);         // visina = 200
for (int i = 0; i < titles.size(); i++) {
  flow.getChildren().add(new Label(titles[i]);
}
```

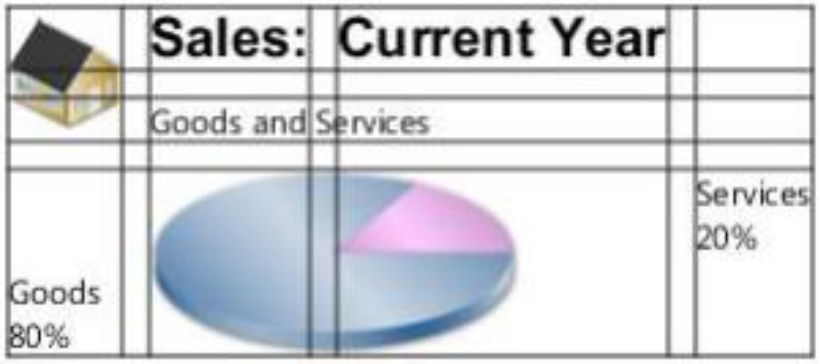
- BorderPane raspoređuje komponente (top, left, center, right, bottom) tako da kontejner bude popunjen.
  - Bilo koja pozicija može da bude null.
  - Pozadina i ivice mogu se stilizovati CSSom

```
BorderPane borderpane = new BorderPane();
ToolBar toolbar = new ToolBar();
HBox statusbar  = new HBox();
Region reg = new Region();
borderPane.setTop(toolbar);
borderPane.setCenter(reg);
borderPane.setBottom(statusbar);
```



# Layout klase

- GridPane raspoređuje čvorove u fleksibilnu pravougaonu mrežu.
  - Čvorovi mogu da budu smešteni u bilo koje ćelije u mreži.

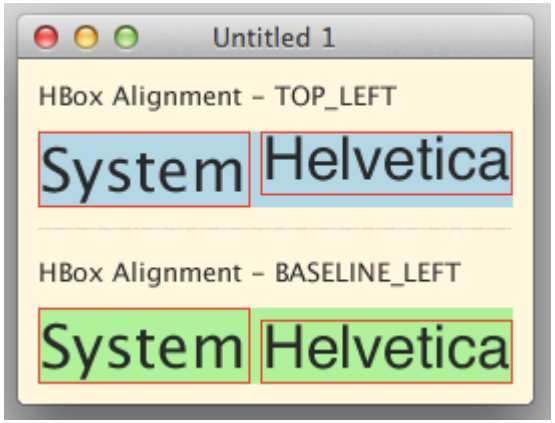



```
GridPane gridpane = new GridPane();
Button button = new Button();
GridPane.setColumnIndex(button, 2);
GridPane.setRowIndex   (button, 1);
Label label = new Label();
GridPane.setConstraints(label, 3, 1); //kolona pa red
gridpane.getChildren().addAll(button, label);
gridpane.add(new Button(), 2, 1); //postavljanje elementa u kolonu i red
gridpane.add(new Label(), 3, 1);
```

# Paket javafx.geometry

- Paket javafx.geometry obezbeđuje skup 2D klasa za definisanje i obavljanje operacija nad objektima u dvodimenzionoj geometriji (Insets, Point2d, Point3d, Rectangle2d, ...).
- Paket javafx.geometry, enumeracije:

```
javafx.geometry.VPos
javafx.geometry.VerticalDirection
javafx.geometry.Side
javafx.geometry.Pos
javafx.geometry.Orientation
javafx.geometry.HPos
javafx.geometry.HorizontalDirection
```




- javafx.geometry.Pos
  - skup vrednosti za opisivanje vertikalnog i horizontalnog pozicioniranja i poravnanja.

```
      TOP_LEFT,      TOP_CENTER,      TOP_RIGHT
   CENTER_LEFT,          CENTER,   CENTER_RIGHT,
 BASELINE_LEFT, BASELINE_CENTER, BASELINE_RIGHT
   BOTTOM_LEFT,   BOTTOM_CENTER,   BOTTOM_RIGHT
```

# javafx.geometry.Pos

- Metode javafx.geometry.Pos
  - `HPos getHpos()` vraća horizontalnu poziciju/poravnanje
  - `VPos getVpos()` vraća vertikalnu poziciju/poravnanje
  - `static Pos valueOf(java.lang.String name)` vraća enum konstantu zadatog imenom name
  - `static Pos[] values()` vraća niz konstanti enum tipa, redom kojim su deklarisani.
- javafx.geometry.Hpos
  - Skup vrednosti za opisivanje horizontalnog pozicioniranja i poravnavanja.
    - `LEFT, CENTER, RIGHT`
  - `static HPos valueOf(java.lang.String name)`
  - `static HPos[] values()`
- javafx.geometry.Vpos
  - Skup vrednosti za opisivanje vertikalnog pozicioniranja i poravnanja.
    - `TOP, CENTER, BASELINE, BOTTOM`
  - `static VPos valueOf(java.lang.String name)`
  - `static VPos[] values()`

# javafx.geometry.Orientation, javafx.geometry.Insets

- javafx.geometry.Orientation
  - Određivanje horizontalne i vertikalne orijentacije.
  - `HORIZONTAL` horizontalna (levo, desno) orijentacija.
  - `VERTICAL` vertikalna (vrh, dno) orijentacija.
  - `static Orientation valueOf(java.lang.String name)`
  - `static Orientation[] values()`
- javafx.geometry.Insets
  - Rastojanja za 4 strane pravougaone površine.
  - Konstruktori:
    - `Insets(double topRightBottomLeft)`
      - Formira se Inset objekat koji ima jednaku vrednost rastojanja za sva četiri razmaka.
    - `Insets(double top, double right, double bottom, double left)`
      - Zadaju se 4 vrednosti za rastojanja.

# javafx.scene.paint.Color,

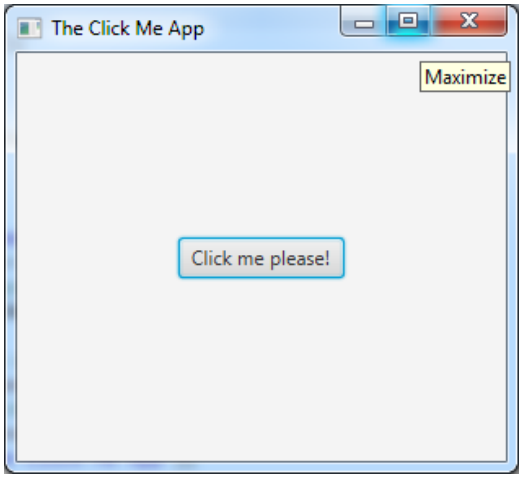
- javafx.scene.paint.Color
  - Standardni konstruktor:
    - `Color(double red, double green, double blue, double opacity)`
      - red, green, blue, opacity (sve u intervalu [0,1])
  - U RGB modelu boja se definiše kao kombinacija tri boje (RGB, odnosno, crvene, zelene i plave) u interval 0-255.
    - `Color boja = Color.rgb(0,0,0);` crna
    - `Color boja = Color.rgb(255,255,255);` bela
  - U HSB modelu boja je definisana kao kombinacija nijanse, zasićenosti, osvetljenja i neprozirnosti:
    - hsb(double hue, double saturation, double brightness)
    - hsb(double hue, double saturation, double brightness, double opacity)
      - `Color boja = Color.hsb(270,1.0,1.0);`
      - `Color boja = Color.hsb(270,1.0,1.0,1.0);`
  - Boja ima podrazumevano alpha vrednost 1.0 (ili 255 prema opsegu) što znači neprozirnost (0 je potpuna providnost).
    - `Color boja = new Color(0,0,1,0.7); // alpha = 0.7`

# javafx.scene.paint.Color, javafx.scene.Cursor

- RGB boja može se postaviti metodom web:
  - `web(java.lang.String colorString)`
  - `web(java.lang.String colorString, double opacity)`
    - `Color boja = Color.web("0x0000FF",1.0);//neprozirna plava`
- Klasa Color definiše statičke konstante za boje: WHITE (255,255,255), PINK (255,175,175), ORANGE (255,200,0), GRAY (128,128,128), CYAN (0,255,255), GREEN (0,255,0), BLACK (0,0,0), RED (255,0,0), LIGHTGRAY (192,192,192), MAGENTA (255,0,255), YELLOW (255,255,0), DARKGRAY (64,64,64), BLUE (0,0,255).
- Kreirana boja se može posvetliti ili potamniti respektivno metodama:
  - `boja.brighter();`
  - `boja.darker();`
- Vraćanje komponenti boje postiže se metodama:
  - `getRed(), getGreen(), getBlue().`
- Poređenje boja vrši se metodom:
  - `equals()`
- Klasa javafx.scene.Cursor definiše statičke konstante koje određuju tip kursora:
  - `CLOSED_HAND, CROSSHAIR, DEFAULT, HAND, TEXT, WAIT`

# JavaFX, upravljanje događajima

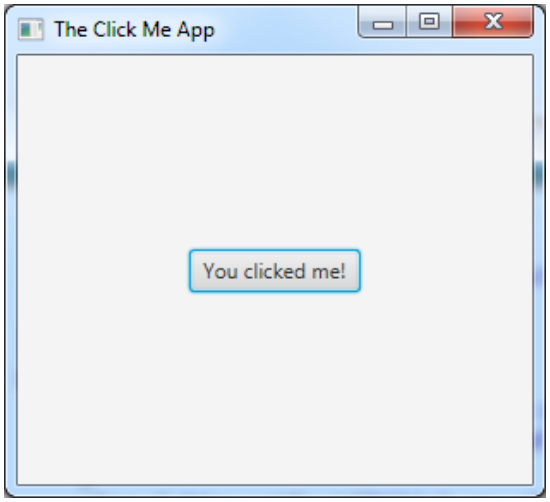
- Daje se primer koji prikazuje programsko dugme i gde postoji reakcija na događaj klik.




```
import javafx.application.*;
import javafx.stage.*;
import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
public class FXClickMe extends Application {
    public static void main(String[] args) {
        launch(args); //sinhrona
    }
    Button btn;
    @Override public void start(Stage primaryStage) {
        // Create the button
        btn = new Button();
        btn.setText("Click me please!");
        btn.setOnAction(Beograd -> buttonClick());
        // Add the button to a layout pane
        BorderPane pane = new BorderPane();
        pane.setCenter(btn);
        // Add the layout pane to a scene
        Scene scene = new Scene(pane, 300, 250);
```

# JavaFX, upravljanje događajima

```
        // Finalize and show the stage
        primaryStage.setScene(scene);
        primaryStage.setTitle("The Click Me App");
        primaryStage.show();
    }
    public void buttonClick() {
        if (btn.getText() == "Click me please!") {
                btn.setText("You clicked me!");
        }
        else { btn.setText("Click me please!");
        }
    }
}
```




- U main-u je startovana JavaFX aplikacija (sinhronom metodom *launch(String[])*) koja će automatski pozvati metodu *start(Stage)*.
- Metoda start prihvata kao argument 'pozornicu' (*Stage*) koja se koristi te se kreira scena koja će biti postavljena na 'pozornicu'.
- Scena ima odgovarajući raspored (*BorderPane*) komponenti (*Button*) koje se dodaju na scenu.
- Metoda programskog dugmeta *setOnAction* postavlja koja metoda obrađuje njegov događaj *click* sa parametrom koji je u primeru lambda izraz (*Beograd -> buttonClick()*), *Beograd* je *ActionEvent* referenca.

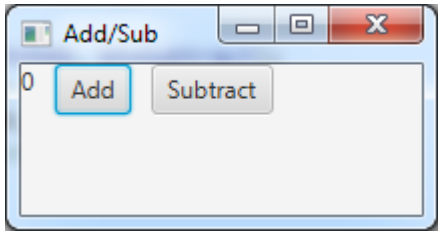
# EventHandler<ActionEvent>

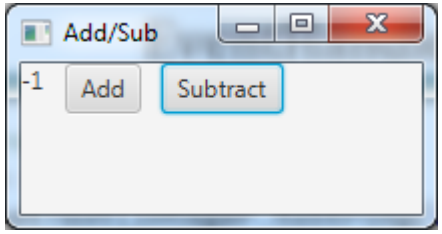
- U sledećem primeru glavna klasa implementira interfejs EventHandler<ActionEvent>:

```
import javafx.application.*; import javafx.stage.*;
import javafx.scene.*; import javafx.scene.layout.*;
import javafx.scene.control.*; import javafx.event.*;
public class AddSubtract extends Application
                         implements EventHandler<ActionEvent> {
  Button btnAdd;
  Button btnSubtract;
  Label lbl;
  int iCounter = 0;
  public static void main(String[] args) {    launch(args);  }
  @Override
  public void start(Stage primaryStage) {
    // Create the Add button
    btnAdd = new Button();
    btnAdd.setText("Add");
    btnAdd.setOnAction(this);
    // Create the Subtract button
    btnSubtract = new Button();
    btnSubtract.setText("Subtract");
    btnSubtract.setOnAction(this);
```

# EventHandler<ActionEvent>

```
    // Create the Label
    lbl = new Label();
    lbl.setText(Integer.toString(iCounter));
    // Add the buttons and label to an HBox pane
    HBox pane = new HBox(10); //spacing 10px
    pane.getChildren().addAll(lbl, btnAdd, btnSubtract);
    // Add the layout pane to a scene
    Scene scene = new Scene(pane, 200, 75);
    // Add the scene to the stage, set the title and show the stage
    primaryStage.setScene(scene);
    primaryStage.setTitle("Add/Sub");
    primaryStage.show();
  }
  @Override
  public void handle(ActionEvent e) {
    if (e.getSource() == btnAdd) { iCounter++; }
    else {
      if (e.getSource() == btnSubtract) { iCounter--; }
    }
    lbl.setText(Integer.toString(iCounter));
  }
}
```



# Unutrašnja klasa koja implementira EventHandler<ActionEvent>

- U ovom primeru dato je rešenje korišćenjem unutrašnje klase koja implementira interfejs EventHandler<ActionEvent>.

```
import javafx.application.*; import javafx.stage.*;
import javafx.scene.*; import javafx.scene.layout.*;
import javafx.scene.control.*; import javafx.event.*;
public class AddSubtract2 extends Application {
  Button btnAdd; Button btnSubtract; Label lbl; int iCounter = 0;
  public static void main(String[] args) { launch(args); }
  @Override
  public void start(Stage primaryStage) {
    // Create a ClickHandler instance
    ClickHandler ch = new ClickHandler();
    btnAdd = new Button();       btnAdd.setText("Add");
    btnAdd.setOnAction(ch);
    btnSubtract = new Button(); btnSubtract.setText("Subtract");
    btnSubtract.setOnAction(ch);
    lbl = new Label(); lbl.setText(Integer.toString(iCounter));
    HBox pane = new HBox(10);
    pane.getChildren().addAll(lbl, btnAdd, btnSubtract);
    Scene scene = new Scene(pane, 200, 75);
    primaryStage.setScene(scene);     primaryStage.setTitle("Add/Sub");
    primaryStage.show();
  }
```

# Unutrašnja klasa koja implementira EventHandler<ActionEvent>

```
  private class ClickHandler implements EventHandler<ActionEvent> {
    @Override
    public void handle(ActionEvent e) {
      if (e.getSource() == btnAdd) { iCounter++; }
      else {
        if (e.getSource() == btnSubtract) { iCounter--; }
      }
      lbl.setText(Integer.toString(iCounter));
    }
  }
}
```

- Za oba programska dumeta postavljeno je metodom *setOnAction* da događaj *klik* bude obrađen pozivom metode *handle* objekta klase *ClickHandler* koja je unutrašnja klasa glavne klase.
- U sledećem primeru dato je rešenje koje koristi anonimnu klasu za obradu događaja klik.

# Rešenje korišćenjem anonimne klase

```
import javafx.application.*; import javafx.stage.*;
import javafx.scene.*; import javafx.scene.layout.*;
import javafx.scene.control.*; import javafx.event.*;
public class AddSubtract3 extends Application {
  Button btnAdd; Button btnSubtract; Label lbl; int iCounter = 0;
  public static void main(String[] args) { launch(args); }
  @Override
  public void start(Stage primaryStage) {
    btnAdd = new Button(); btnAdd.setText("Add");
    //anonimna klasa
    btnAdd.setOnAction(new EventHandler<ActionEvent>() {
      public void handle(ActionEvent e) {
                    iCounter++;
                    lbl.setText(Integer.toString(iCounter));
      }
    });
    btnSubtract = new Button(); btnSubtract.setText("Subtract");
    btnSubtract.setOnAction(new EventHandler<ActionEvent>() {
      public void handle(ActionEvent e) {
                    iCounter--;
                    lbl.setText(Integer.toString(iCounter));
      }
    });
```

# Rešenje korišćenjem anonimne klase

```
        lbl = new Label();
        lbl.setText(Integer.toString(iCounter));
        HBox pane = new HBox(10);
        pane.getChildren().addAll(lbl, btnAdd, btnSubtract);
        Scene scene = new Scene(pane, 200, 75);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Add/Sub");
        primaryStage.show();
    }
}
```

- U sledećem primeru dato je rešenje koje koristi lambda izraz.
- Interfejs *EventHandler* ima samo jednu apstraktnu metodu (metoda *handle*). tako da je *EventHandler* funkcionalni interfejs i može se koristiti korišćenjem lambda izraza (*Lambda expression*).
- U ovom slučaju korišćenjem lambda izraza kreira se anonimna klasa koja implementira funkcionalni interfejs tako što se definiše samo:
  - parametar
  - telo metode.

  Java kompajler sada zaključuje ostatak koda na temelju konteksta u kom se koristi lambda izraz.

# Rešenje korišćenjem lambda izraza

- Parametar i telo metode se razdvojeni znakom -> kao što sledi:

```
e ->
    {
       iCounter++;
       lbl.setText(Integer.toString(iCounter);
    }
```

  - Ovde lambda izraz implementira funkcionalni interfejs čija jedina metoda prihvata jedini parametar označen sa e.
  - U telu metode se inkrementira *iCounter* i ažurira tekst labele *lbl* da prikaže novu vrednost *iCounter-a*.
- Registrovanje lambda izraza kao upravljača događaja za programsko dugme je kao što sledi:

```
btnAdd.setOnAction( e ->
                          {
                            iCounter++;
                            lbl.setText(Integer.toString(iCounter));
} );
```

- Naziv metode se ne mora znati pošto funkcionalni interfejs *EventHandler* ima jedan apstraktni metod (*handle*) a ne mora se znati ni naziv interfejsa pošto je interfejs određen kontekstom (*setOnAction* metoda ima jedan parametar tipa *EventHandler<ActionEvent>*) što sve odradi kompajler.

# Rešenje korišćenjem lambda izraza

```
import javafx.application.*; import javafx.stage.*;
import javafx.scene.*; import javafx.scene.layout.*;
import javafx.scene.control.*;
public class AddSubtract4 extends Application {
    Button btnAdd; Button btnSubtract; Label lbl; int iCounter = 0;
    public static void main(String[] args) { launch(args);  }
    @Override
    public void start(Stage primaryStage) {
        btnAdd = new Button();          btnAdd.setText("Add");
        btnAdd.setOnAction( e ->
            {
                iCounter++;
                lbl.setText(Integer.toString(iCounter));
            } );
        btnSubtract = new Button();    btnSubtract.setText("Subtract");
        btnSubtract.setOnAction( e ->
            {
                iCounter--;
                lbl.setText(Integer.toString(iCounter));
            } );
        lbl = new Label();       lbl.setText(Integer.toString(iCounter));
        HBox pane = new HBox(10);
        pane.getChildren().addAll(lbl, btnAdd, btnSubtract);
```

# Rešenje korišćenjem lambda izraza

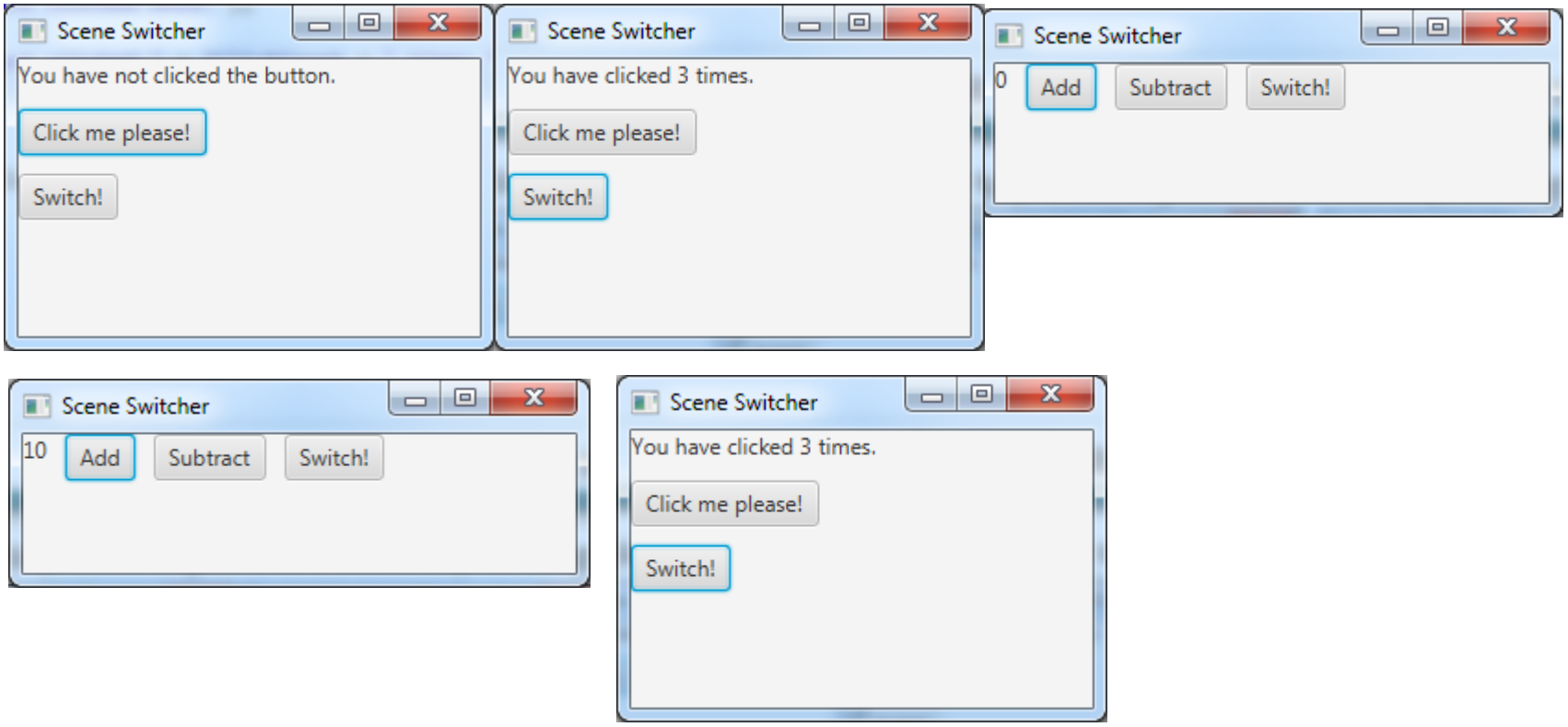
```
        Scene scene = new Scene(pane, 200, 75);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Add/Sub");
        primaryStage.show();
    }
}
```

- Moguće je i rešenje u kome bi bio implementiran poziv korisničkih metoda :

```
...
btnAdd.setOnAction( e -> btnAdd_Click() );
btnSubtract.setOnAction( e -> btnSubtract_Click() );
...
private void btnAdd_Click()
{
     iCounter++;
     lbl.setText(Integer.toString(iCounter));
}
private void btnSubtract_Click()
{
     iCounter--;
     lbl.setText(Integer.toString(iCounter));
}
```

# Smenjivanje scena

- Stage ne mora da se veže za samo jednu scenu. Sledi primer u kome se dve scene smenjuju na istom Stage-u.




```
import javafx.application.*; import javafx.stage.*;
import javafx.scene.*; import javafx.scene.layout.*;
import javafx.scene.control.*;
public class SceneSwitcher extends Application {
  public static void main(String[] args) { launch(args); }
```

# Smenjivanje scena

```
// class fields for Click-Counter scene
int iClickCount = 0; Label lblClicks;
Button btnClickMe; Button btnSwitchToScene2;
Scene scene1;
// class fields for Add-Subtract scene
int iCounter = 0; Label lblCounter;
Button btnAdd; Button btnSubtract; Button btnSwitchToScene1;
Scene scene2;
// class field for stage
Stage stage;
@Override
public void start(Stage primaryStage) {
  stage = primaryStage;
  // Build the Click-Counter scene
  lblClicks = new Label();
  lblClicks.setText("You have not clicked the button.");
  btnClickMe = new Button();
  btnClickMe.setText("Click me please!");
  btnClickMe.setOnAction(e -> btnClickMe_Click());
  btnSwitchToScene2 = new Button();
  btnSwitchToScene2.setText("Switch!");
  btnSwitchToScene2.setOnAction(e -> btnSwitchToScene2_Click());
```

# Smenjivanje scena

```
    VBox pane1 = new VBox(10);
    pane1.getChildren().addAll(lblClicks, btnClickMe, btnSwitchToScene2);
    scene1 = new Scene(pane1, 250, 150);

    // Build the Add-Subtract scene
    lblCounter = new Label();
    lblCounter.setText(Integer.toString(iCounter));
    btnAdd = new Button();      btnAdd.setText("Add");
    btnAdd.setOnAction(e -> btnAdd_Click());
    btnSubtract = new Button(); btnSubtract.setText("Subtract");
    btnSubtract.setOnAction(e -> btnSubtract_Click());
    btnSwitchToScene2 = new Button();
    btnSwitchToScene2.setText("Switch!");
    btnSwitchToScene2.setOnAction(e -> btnSwitchToScene1_Click());
    HBox pane2 = new HBox(10);
    pane2.getChildren().addAll(lblCounter, btnAdd, btnSubtract,
                                                   btnSwitchToScene2);
    scene2 = new Scene(pane2, 300, 75);

    // Set the stage with scene 1 and show the stage
    primaryStage.setScene(scene1);
    primaryStage.setTitle("Scene Switcher");
    primaryStage.show();
}
```

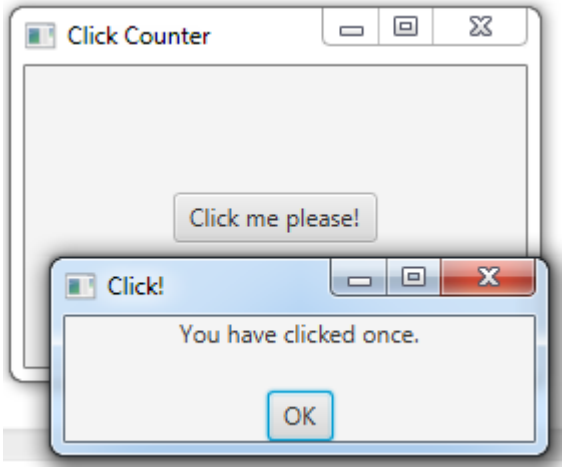
# Smenjivanje scena

```
  // Event handlers for scene 1
  public void btnClickMe_Click() {
    iClickCount++;
    if (iClickCount == 1) { lblClicks.setText("You have clicked once.");
    } else {
       lblClicks.setText("You have clicked " + iClickCount + " times.");
    }
  }
  private void btnSwitchToScene2_Click() {
    stage.setScene(scene2);
  }

  // Event handlers for scene 2
  private void btnAdd_Click() {
    iCounter++;
    lblCounter.setText(Integer.toString(iCounter));
  }
  private void btnSubtract_Click() {
    iCounter--;
    lblCounter.setText(Integer.toString(iCounter));
  }
  private void btnSwitchToScene1_Click() { stage.setScene(scene1); }
}
```

# Kreiranje MessageBox klase

- Data je implementacija klase MesageBox za ispisivanje poruka.

```
import javafx.application.*;   import javafx.stage.*;
import javafx.scene.*;         import javafx.scene.layout.*;
import javafx.scene.control.*; import javafx.geometry.*;
public class MessageBox {
  public static void show(String message, String title) {
    Stage stage = new Stage();
    stage.initModality(Modality.APPLICATION_MODAL);
    stage.setTitle(title);
    stage.setMinWidth(250);
     Label lbl = new Label();
     lbl.setText(message);
     Button btnOK = new Button();
     btnOK.setText("OK");
     btnOK.setOnAction(e -> stage.close());
    VBox pane = new VBox(20);
    pane.getChildren().addAll(lbl, btnOK);
    pane.setAlignment(Pos.CENTER);
    Scene scene = new Scene(pane);
    stage.setScene(scene);
    stage.showAndWait();
  }
}
```



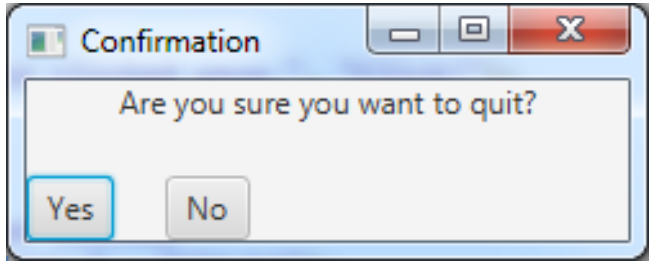
# Korišćenje MessageBox klase

- Pritiskom na dugme MessageBox ispisuje informaciju o broju klik-ova.

```
import javafx.application.*; import javafx.stage.*;
import javafx.scene.*; import javafx.scene.layout.*;
import javafx.scene.control.*;
public class ClickCounter extends Application {
  public static void main(String[] args) { launch(args);   }
  int iClickCount = 0; Button btn;
  @Override
  public void start(Stage primaryStage)  {
    btn = new Button(); btn.setText("Click me please!");
    btn.setOnAction(e -> buttonClick());
    BorderPane pane = new BorderPane();
    pane.setCenter(btn);
    Scene scene = new Scene(pane, 250, 150);
    primaryStage.setScene(scene);primaryStage.setTitle("Click Counter");
    primaryStage.show();
  }
  public void buttonClick() {
    iClickCount++;
    if(iClickCount==1)MessageBox.show("You have clicked once.","Click!");
    else MessageBox.show("You have clicked "+iClickCount+" times.",
                                                        "Click!");
    }}
```

# Kreiranje ConfirmationBox klase

```
import javafx.application.*;  import javafx.stage.*;
import javafx.scene.*;        import javafx.scene.layout.*;
import javafx.scene.control.*;import javafx.geometry.*;

public class ConfirmationBox {
  static Stage stage;
  static boolean btnYesClicked;

  public static boolean show(String message, String title,
                             String textYes, String textNo) {
    btnYesClicked = false;
    stage = new Stage();
    stage.initModality(Modality.APPLICATION_MODAL);
    stage.setTitle(title);
    stage.setMinWidth(250);
    Label lbl = new Label();
    lbl.setText(message);
    Button btnYes = new Button();
    btnYes.setText(textYes);
    btnYes.setOnAction(e -> btnYes_Clicked());
    Button btnNo = new Button();
    btnNo.setText(textNo);
    btnNo.setOnAction(e -> btnNo_Clicked());
```



# Kreiranje ConfirmationBox klase, pripadni poziv

```
    HBox paneBtn = new HBox(20);
    paneBtn.getChildren().addAll(btnYes, btnNo);
    VBox pane = new VBox(20);
    pane.getChildren().addAll(lbl, paneBtn);
    pane.setAlignment(Pos.CENTER);
    Scene scene = new Scene(pane);
    stage.setScene(scene);     stage.showAndWait();
    return btnYesClicked;
  }
  private static void btnYes_Clicked() { stage.close();
                                         btnYesClicked = true;
  }
  private static void btnNo_Clicked() {  stage.close();
                                         btnYesClicked = false;
  } }
```

- Poziv ConfirmationBox-a bi bio npr:

```
public void btnClose_Click()   {
  boolean confirm = false;
  confirm = ConfirmationBox.show(
     "Are you sure you want to quit?", "Confirmation", "Yes", "No");
  if (confirm) stage.close();
 }
```

# Unos teksta: TextField

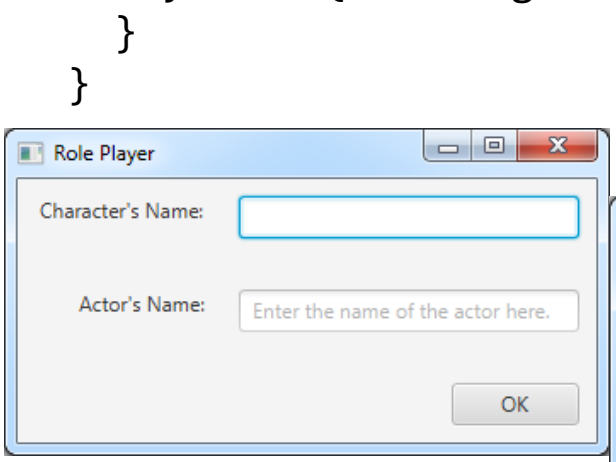
```
import javafx.application.*;
import javafx.stage.*; import javafx.scene.*;
import javafx.scene.layout.*;
import javafx.scene.control.*;
import javafx.geometry.*;

public class RolePlayer extends Application{

  public static void main(String[] args) {  launch(args);  }

  TextField txtCharacter;   TextField txtActor;
  @Override
  public void start(Stage primaryStage) {
    // Create the Character
    Label lblCharacter = new Label("Character's Name:");
    lblCharacter.setMinWidth(100);
    lblCharacter.setAlignment(Pos.BOTTOM_RIGHT);
    // Create the Character text field
    txtCharacter = new TextField();
    txtCharacter.setMinWidth(200);
    txtCharacter.setMaxWidth(200);
    txtCharacter.setPromptText("Enter the name of the character here.");
    // Create the Actor label
    Label lblActor = new Label("Actor's Name:");
    lblActor.setMinWidth(100); lblActor.setAlignment(Pos.BOTTOM_RIGHT);
```

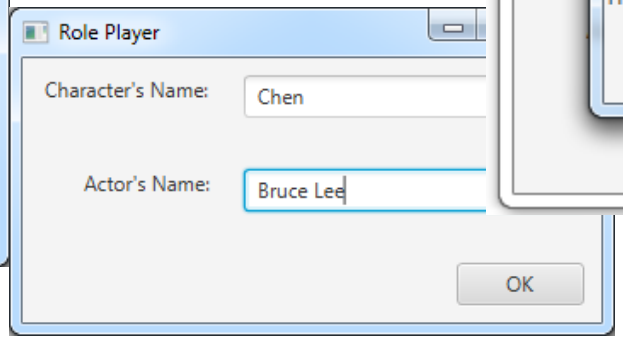
# Unos teksta: TextField

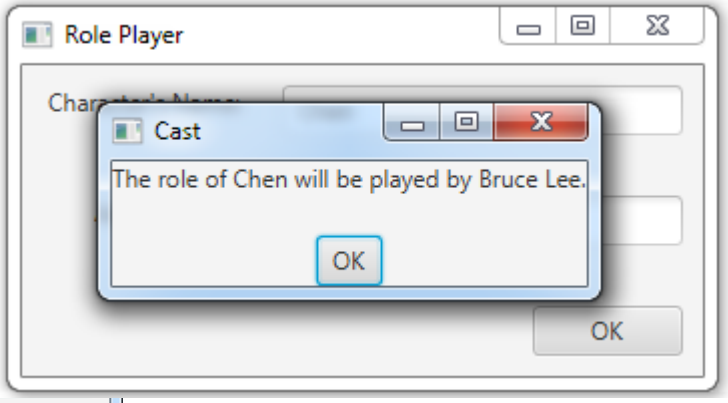
```
// Create the Actor text field
txtActor = new TextField();
txtActor.setMinWidth(200);
txtActor.setMaxWidth(200);
txtActor.setPromptText("Enter the name of the actor here.");
// Create the OK button
Button btnOK = new Button("OK");
btnOK.setMinWidth(75);
btnOK.setOnAction(e -> btnOK_Click());
// Create the Character pane
HBox paneCharacter = new HBox(20, lblCharacter, txtCharacter);
paneCharacter.setPadding(new Insets(10));
// Create the Actor pane
HBox paneActor = new HBox(20, lblActor, txtActor);
paneActor.setPadding(new Insets(10));
// Create the Button pane
HBox paneButton = new HBox(20, btnOK);
paneButton.setPadding(new Insets(10));
paneButton.setAlignment(Pos.BOTTOM_RIGHT);
// Add the Character, Actor, and Button panes to a VBox
VBox pane = new VBox(10, paneCharacter, paneActor, paneButton);
// Set the stage
Scene scene = new Scene(pane);
```

# Unos teksta: TextField

```
        primaryStage.setScene(scene); primaryStage.setTitle("Role Player");
        primaryStage.show();
    }
    public void btnOK_Click() {
        String errorMessage = "";
        if (txtCharacter.getText().length() == 0)
                errorMessage += "\nCharacter is a required field.";
        if (txtActor.getText().length() == 0)
                errorMessage += "\nActor is a required field.";
        if (errorMessage.length() == 0) {
          String message = "The role of " + txtCharacter.getText() +
                           " will be played by " + txtActor.getText() + ".";
          MessageBox.show(message, "Cast");
        } else {  MessageBox.show(errorMessage, "Missing Data"); }
    }
}
```



# TextField, RadioButton, CheckBox

```
import javafx.application.*;   import javafx.stage.*;
import javafx.scene.*;         import javafx.scene.layout.*;
import javafx.scene.control.*; import javafx.geometry.*;
import javafx.scene.text.*;
public class PizzaOrder extends Application {
  public static void main(String[] args) { launch(args); }
  Stage stage;
  // Customer name, phone, and address fields
  TextField txtName;
  TextField txtPhone;
  TextField txtAddress;
  // Size radio buttons
  RadioButton rdoSmall;
  RadioButton rdoMedium;
  RadioButton rdoLarge;
  // Crust style radio buttons
  RadioButton rdoThin;
  RadioButton rdoThick;
  // Topping radio buttons
  CheckBox chkPepperoni;
  CheckBox chkSausage;
  CheckBox chkLinguica;
  CheckBox chkOlives;
```

# TextField, RadioButton, CheckBox

```
CheckBox chkMushrooms;
CheckBox chkTomatoes;
CheckBox chkAnchovies;
@Override
public void start(Stage primaryStage) {
  stage = primaryStage;
  // ----- Create the top pane -----
  Text textHeading = new Text("Order Your Pizza Now!");
  textHeading.setFont(new Font(20));
  HBox paneTop = new HBox(textHeading);
  paneTop.setPadding(new Insets(20, 10, 20, 10));
  // ---------- Create the customer pane ----------
  // Create the name label and text field
  Label lblName = new Label("Name:");
  lblName.setPrefWidth(100);
  txtName = new TextField();
  txtName.setPrefColumnCount(20);
  txtName.setPromptText("Enter the customer's name here");
  txtName.setMaxWidth(Double.MAX_VALUE);
  HBox paneName = new HBox(lblName, txtName);
  // Create the phone number label and text field
  Label lblPhone = new Label("Phone Number:");
  lblPhone.setPrefWidth(100);
```

# TextField, RadioButton, CheckBox

```
txtPhone = new TextField();
txtPhone.setPrefColumnCount(20);
txtPhone.setPromptText("Enter the customer's phone number here");
HBox panePhone = new HBox(lblPhone, txtPhone);
// Create the address label and text field
Label lblAddress = new Label("Address:");
lblAddress.setPrefWidth(100);
txtAddress = new TextField();
txtAddress.setPrefColumnCount(20);
txtAddress.setPromptText("Enter the customer's address here");
HBox paneAddress = new HBox(lblAddress, txtAddress);
// Create the customer pane
VBox paneCustomer = new VBox(10, paneName, panePhone, paneAddress);
// ---------- Create the order pane ----------
// Create the size pane
Label lblSize = new Label("Size");
rdoSmall = new RadioButton("Small");
rdoMedium = new RadioButton("Medium");
rdoLarge = new RadioButton("Large");
rdoMedium.setSelected(true);
ToggleGroup groupSize = new ToggleGroup();
rdoSmall.setToggleGroup(groupSize);
rdoMedium.setToggleGroup(groupSize);
```

# TextField, RadioButton, CheckBox

```
rdoLarge.setToggleGroup(groupSize);
VBox paneSize = new VBox(lblSize, rdoSmall, rdoMedium, rdoLarge);
paneSize.setSpacing(10);
// Create the crust pane
Label lblCrust = new Label("Crust");
rdoThin = new RadioButton("Thin");
rdoThick = new RadioButton("Thick");
rdoThin.setSelected(true);
ToggleGroup groupCrust = new ToggleGroup();
rdoThin.setToggleGroup(groupCrust);
rdoThick.setToggleGroup(groupCrust);
VBox paneCrust = new VBox(lblCrust, rdoThin, rdoThick);
paneCrust.setSpacing(10);
// Create the toppings pane
Label lblToppings = new Label("Toppings");
chkPepperoni = new CheckBox("Pepperoni");
chkSausage = new CheckBox("Sausage");
chkLinguica = new CheckBox("Linguica");
chkOlives = new CheckBox("Olives");
chkMushrooms = new CheckBox("Mushrooms");
chkTomatoes = new CheckBox("Tomatoes");
chkAnchovies = new CheckBox("Anchovies");
```

# TextField, RadioButton, CheckBox

```
FlowPane paneToppings = new FlowPane(Orientation.VERTICAL,
      chkPepperoni, chkSausage, chkLinguica, chkOlives,
      chkMushrooms, chkTomatoes, chkAnchovies);
paneToppings.setPadding(new Insets(10, 0, 10, 0));
paneToppings.setHgap(20);
paneToppings.setVgap(10);
paneToppings.setPrefWrapLength(100);
VBox paneTopping = new VBox(lblToppings, paneToppings);
// Add the size, crust, and toppings pane to the order pane
HBox paneOrder = new HBox(50, paneSize, paneCrust, paneTopping);
// Create the center pane
VBox paneCenter = new VBox(20, paneCustomer, paneOrder);
paneCenter.setPadding(new Insets(0, 10, 0, 10));
// ---------- Create the bottom pane ----------
Button btnOK = new Button("OK");
btnOK.setPrefWidth(80);
btnOK.setOnAction(e -> btnOK_Click());
Button btnCancel = new Button("Cancel");
btnCancel.setPrefWidth(80);
btnCancel.setOnAction(e -> btnCancel_Click());
Region spacer = new Region();
HBox paneBottom = new HBox(10, spacer, btnOK, btnCancel);
```

# TextField, RadioButton, CheckBox

```
        paneBottom.setHgrow(spacer, Priority.ALWAYS);
        paneBottom.setPadding(new Insets(20, 10, 20, 10));
        // ---------- Finish the scene ----------
        BorderPane paneMain = new BorderPane();
        paneMain.setTop(paneTop);
        paneMain.setCenter(paneCenter);
        paneMain.setBottom(paneBottom);
        // Create the scene and the stage
        Scene scene = new Scene(paneMain);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Pizza Order");
        primaryStage.show();
    }
    public void btnOK_Click() {
        // Create a message string with the customer information
        String msg = "Customer:\n\n";
        msg += "\t" + txtName.getText() + "\n";
        msg += "\t" + txtAddress.getText() + "\n";
        msg += "\t" + txtPhone.getText() + "\n\n";
        msg += "You have ordered a ";
        // Add the pizza size
        if (rdoSmall.isSelected()) msg += "small "; //getText() dace Small
        if (rdoMedium.isSelected())msg += "medium ";
```

# TextField, RadioButton, CheckBox

```
    if (rdoLarge.isSelected()) msg += "large ";
    if (rdoThin.isSelected()) msg += "thin crust pizza with ";  //kora
    if (rdoThick.isSelected())msg += "thick crust pizza with "; //kora
    String toppings = ""; //prelivi
    toppings = buildToppings(chkPepperoni, toppings);
    toppings = buildToppings(chkSausage, toppings);
    toppings = buildToppings(chkLinguica, toppings);
    toppings = buildToppings(chkOlives, toppings);
    toppings = buildToppings(chkTomatoes, toppings);
    toppings = buildToppings(chkMushrooms, toppings);
    toppings = buildToppings(chkAnchovies, toppings);
    if (toppings.equals("")) msg += "no toppings.";
    else                     msg += "the following toppings:\n" + toppings;
    MessageBox.show(msg, "Order Details"); //od ranije
  }
  public String buildToppings(CheckBox chk, String msg) {
    if (chk.isSelected()) {    if (!msg.equals(""))  msg += ", ";
                               msg += chk.getText();
    }
    return msg;
  }
  public void btnCancel_Click() {stage.close();}
}
```

# TextField, RadioButton, CheckBox