GUI aplikacije i JavaFX

- Objektno-orjentisano programiranje 1

GUI aplikacije

- **GUI (graphical user interface) aplikacija – aplikacija sa interaktivnim grafičkim korisničkim interfejsom**
  - Aplikacija sadrži grafičke komponente (polja za unos teksta, dugmadi, menije, itd.)
  - Korisnik u interakciji sa aplikacijom koristeći tastaturu, miš, ekran osetljiv na dodir…
  - **Svaka grafička komponenta je jedan objekat**
- **Programiranje vođeno događajima (*event-driven programming*)**
  - Program izvršava **pozadinsku nit koja čeka na neki događaj** koji korisnik generiše preko grafičkih komponenti interfejsa
  - Za svaki događaj je vezan ***event handler*** – kod koji se izvršava kada je događaj generisan (kod za obradu događaja)
  - **Kod konzolnih aplikacija interakciju program-korisnik kontroliše program, kod GUI aplikacija tu interakciju kontroliše korisnik**

GUI aplikacije

- Programiranje GUI aplikacija se sastoji od
  - Programiranja grafičkog korisničkog interfejsa
    - Kreiranje objekata koji predstavljaju grafičke komponente interfejsa
    - Raspoređivanje grafičkih komponenti unutar okna
      - **Automatski raspoređivači komponenti u zavisnosti od tipa okna u koje smeštamo komponente**
    - **Ručno programiranje interfejsa VS grafički *drag-and-drop* editori sa automatskim generisanjem koda / konfiguracionim fajlovima koji opisuju interfejs**
  - Programiranja *event handler*-a
  - Vezivanje *event handler*-a za grafičke komponente
    - Jedna komponenta interfejsa može generisati više tipova događaja – za svaki tip događaja imamo zaseban *event handler*
    - Jedan *event handler* se može izvršavati na događaje koje generišu različite grafičke komponente

JavaFX

- Skup klasa za programiranje Java GUI aplikacija
- GUI JavaFX aplikacije ima strukturu stabla u čijem korenu se nalazi **pozornica** (eng. ***stage***)
  - **Kod desktop GUI aplikacija pozornica je prozor sa naslovnom linijom**
- **Scena** (eng. ***scene***) – deo prozora u koji možemo smeštati grafičke komponente.
  - **Pozornica ima tačno jednu scenu**
- **Čvorovi** (eng. ***nodes***) – **grafičke komponente** ili **okna**
  - Okna su kontejneri koji mogu da sadrže grafičke komponente ili druga okna
  - **Na scenu možemo postaviti tačno jedan čvor**
  - **Čvorovi se automatski raspoređuju u oknu u zavisnosti od tipa okna**

JavaFX

- JavaFX aplikaciju pravimo tako što nasledimo apstraktnu klasu **Application** iz **javafx.application**

- Implementiramo apstraktni metod

  void start(Stage stage) throws Exception

  kojim se kreira scena aplikacije i postavi na pozornicu

- JavaFX aplikacija se pokreće tako što pozovemo statički metod **launch** iz klase **Application**
  - Ovaj metod napravi instancu naše klase koja nasleđuje klasu **Application**
  - ... i nad tom instancom nakon neophodnih inicijalizacija pozove metod **void start(Stage stage) throws Exception**

JavaFX kostur (*hello world*) aplikacija

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.stage.Stage;

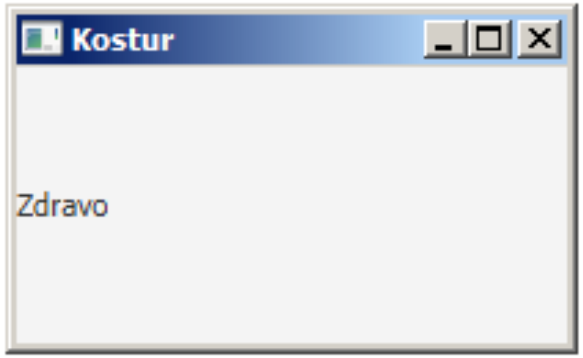
public class Kostur extends Application {

	@Override
	public void start(Stage stage) throws Exception {
		// kreiramo scenu duzine 200 i sirine 100 piksela
		// prvi argument: cvor koji postavljamo na scenu
		Scene scene = new Scene(new Label("Zdravo"), 200, 100);

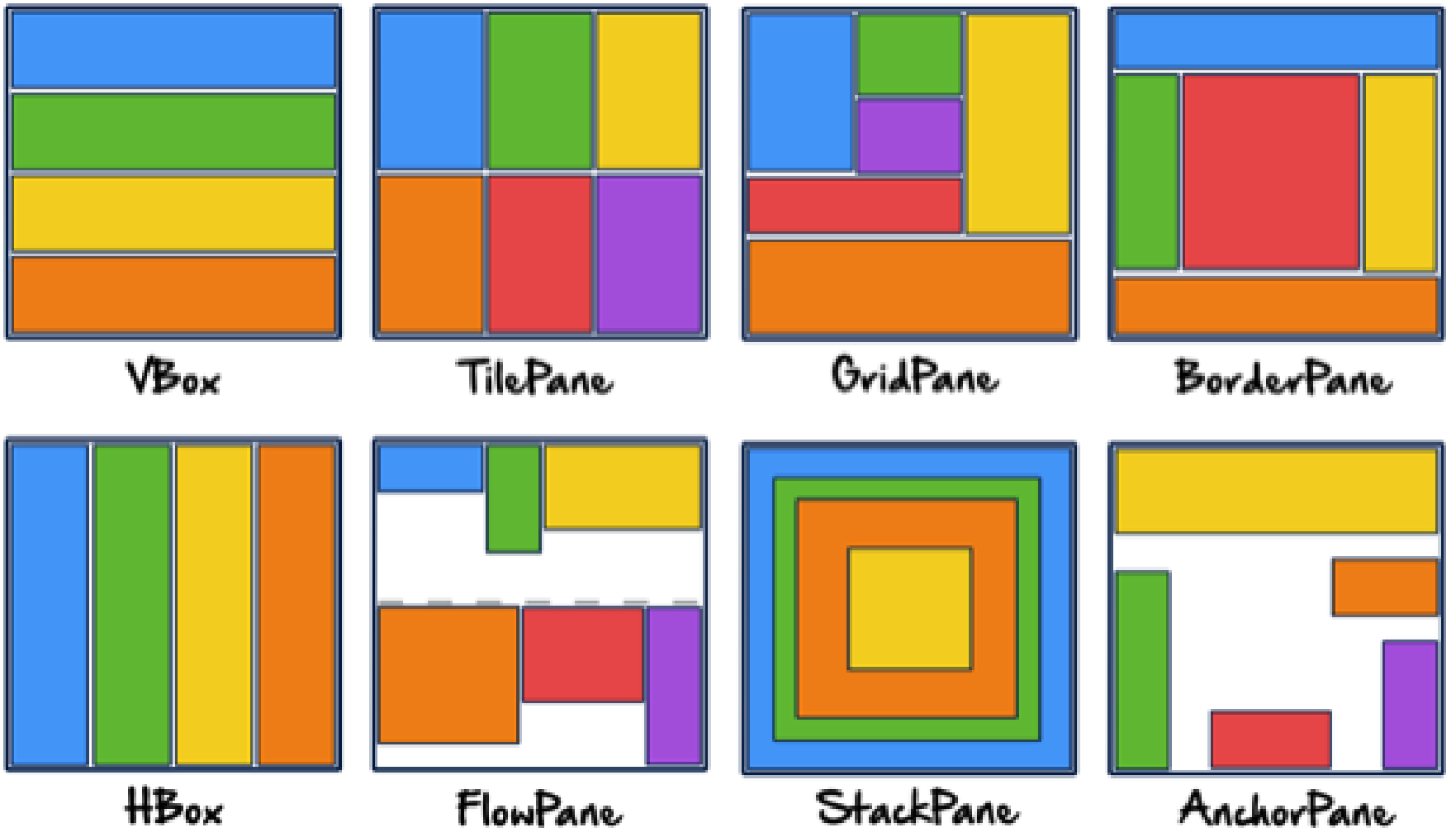
		// inicijalizujemo pozornicu
		stage.setScene(scene);
		stage.setTitle("Kostur");

		// prikazemo pozornicu
		stage.show();
	}

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



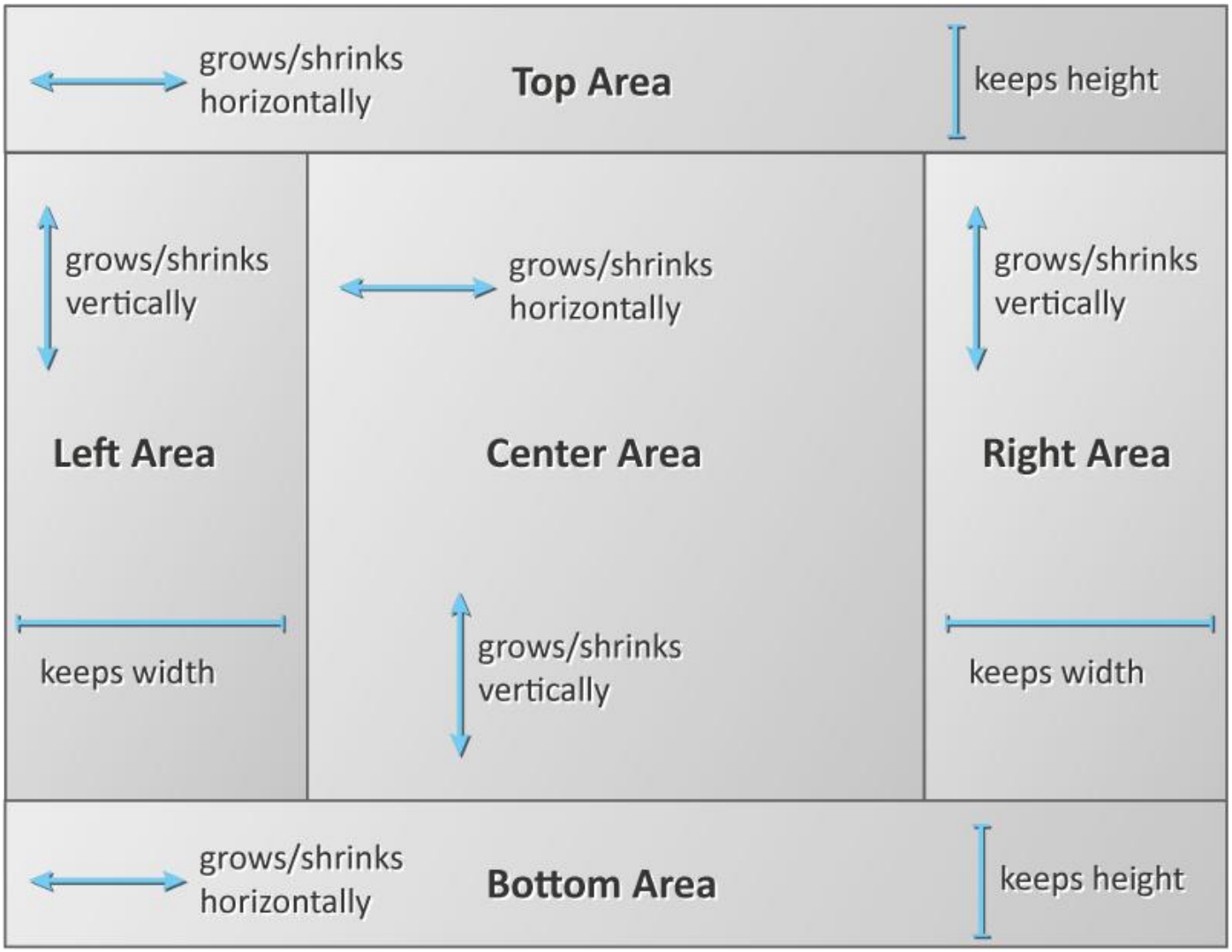
JavaFX okna


VBox
TilePane
GridPane
BorderPane
HBox
FlowPane
StackPane
AnchorPane

JavaFX

- **Okna se prilagođavaju dimenzijama scene**
  - Raspored i/ili veličina čvorova može da se promeni promenom veličine pozornice
- **Za svaki čvor je moguće zadati minimalnu, poželjnu i maksimalnu veličinu (širinu i visinu)**
  - Sve tri veličine se automatski inicijalizuju u zavisnosti od tipa grafičke komponente
  - Čvor ima željenu veličinu ukoliko u oknu ima mesta za takvu veličinu
  - Veličina čvora se može smanjivati do minimalne, a povećavati do maksimalne veličine
  - U slučaju smanjivanja pozornice grafičke komponente mogu da se preklope
- Ako želimo za svaki čvor možemo eksplicitno zadati poziciju i veličinu (**ne preporučuje se!**)

BorderPane okno



BorderPane okno

- Paket **javafx.scene.layout**
- Konstruktori
  - BorderPane()
  - BorderPane(Node center)
  - BorderPane(Node c, Node t, Node r, Node b, Node l)
- Metode
  - void setCenter(Node n)
  - void setLeft(Node n)
  - void setRight(Node n)
  - void setTop(Node n)
  - void setBottom(Node n)
- Metode za postavljanje poravnanja za komponente unutar dela okna
- Metoda za postavljanje margine oko okna
- Metode za postavljanje margine oko komponenti unutar okna

BorderPane okno

- Paket **javafx.scene.layout**
- Konstruktori
  - BorderPane()
  - BorderPane(Node center)
  - BorderPane(Node c, Node t, Node r, Node b, Node l)
- Metode
  - void setCenter(Node n)
  - void setLeft(Node n)
  - void setRight(Node n)
  - void setTop(Node n)
  - void setBottom(Node n)
- Metode za postavljanje poravnanja za komponente unutar okna
- Metoda za postavljanje margine oko okna
- Metode za postavljanje margine oko komponenti unutar okna

```
public class DemoBorderPane extends Application {
	public void start(Stage stage) throws Exception {
		// neke graficke komponente
		Label label = new Label("Labela");
		Button button = new Button("Dugme");
		TextField textField = new TextField("Tekst polje");
		CheckBox checkBox = new CheckBox("Box za cekiranje");
		TextArea textArea = new TextArea();

		BorderPane container = new BorderPane();
		container.setCenter(textArea);
		container.setBottom(button);
		container.setTop(textField);
		container.setLeft(checkBox);
		container.setRight(label);

		Scene s = new Scene(container, 600, 400);
		stage.setScene(s);
		stage.setTitle("Border pane demo");
		stage.show();
	}

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

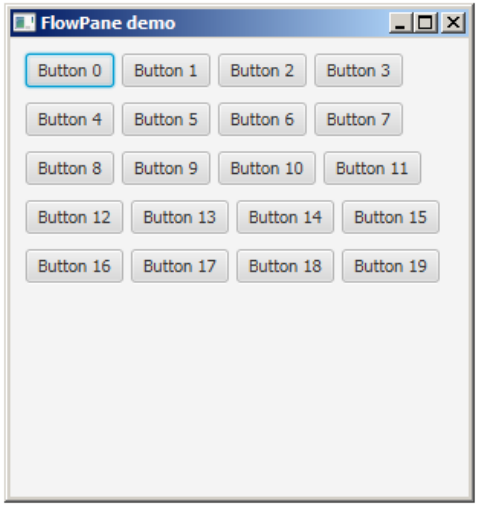
FlowPane okno

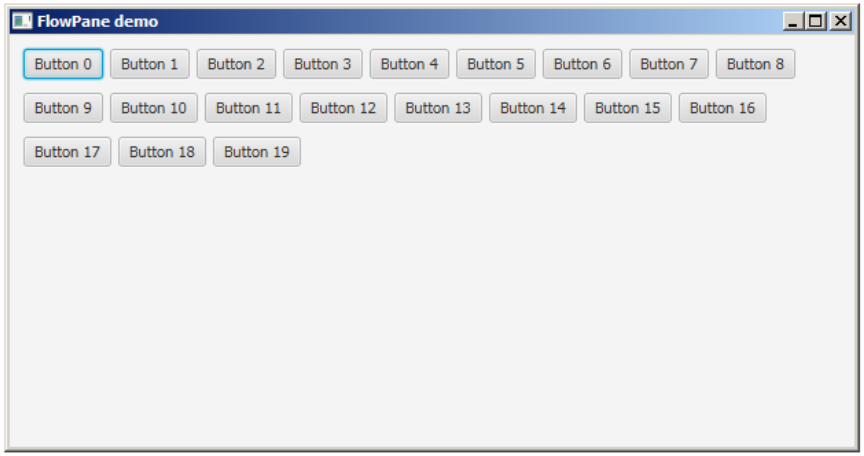
- Ređa komponente jednu za drugom horizontalno/vertikalno sa prelaskom u novu “vrstu”/”kolonu”
- Paket **javafx.scene.layout**
- Konstruktori
  - FlowPane()
  - FlowPane(double hgap, double vgap)
  - FlowPane(Orientation orient)
    - Orientation je enum koji definiše HORIZONTAL i VERTICAL
  - FlowPane(Orientation orient, double hgap, double vgap)
- Metoda koja vraća listu čvorova koji se nalaze u oknu **ObservableList<Node> getChildren()**
- Pozivajući metod **add(Node n)** nad tom listom dodajemo čvor u okno
- ObervableList – lista uz koji možemo vezati **funkcijski objekat** koji se izvrši automatski kada se lista promeni

```
public class DemoFlowPane extends Application {
  public void start(Stage stage) throws Exception {
    FlowPane container = new FlowPane(Orientation.HORIZONTAL);
    container.setHgap(5);
    container.setVgap(10);
    container.setPadding(new Insets(10));
    for (int i = 0; i < 20; i++)
      container.getChildren().add(new Button("Button " + i));

    Scene s = new Scene(container, 300, 300);
    stage.setScene(s);
    stage.setTitle("FlowPane demo");
    stage.show();
  }

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





Hbox i VBox okna

- Ređaju komponente jednu za drugom horizontalno/vertikalno **(bez prelaska u novu “vrstu”/”kolonu”)**
- **Ako je pozornica mala imamo nevdljive komponente koje se pojavljuju proširenjem pozornice**

```
VBox vbContainer = new VBox();
// rastojanje izmedju komponenti
vbContainer.setSpacing(5);
// margina oko okna
vbContainer.setPadding(new Insets(10));
// dodavanje komponenti u okno
for (int i = 0; i < 10; i++)
      vbContainer.getChildren().add(new Button("Dugme -- " + i));

HBox hbContainer = new HBox();
hbContainer.setSpacing(10);
hbContainer.setPadding(new Insets(20));
for (int i = 0; i < 5; i++)
      hbContainer.getChildren().add(new Label("Labela " + i));
```

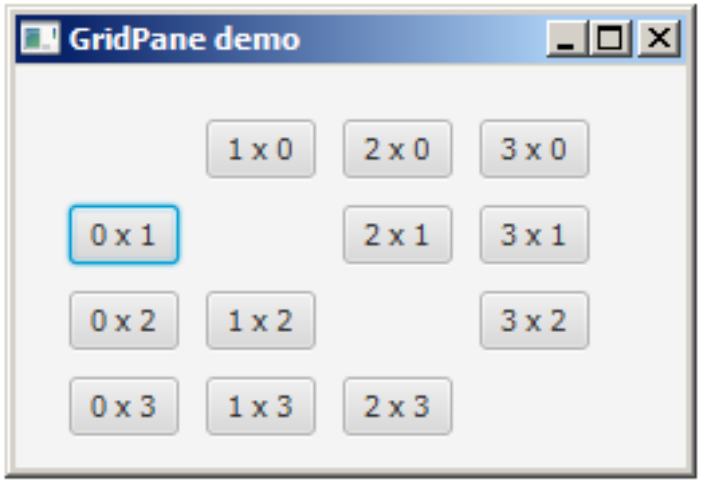
GridPane okno

- **GridPane okno predstavlja tabelu (matricu)**
- Čvor možemo postaviti u proizvoljnu ćeliju u tabeli
- Tabela raste automatski kako dodajemo čvorove u nju
  - **Ne navodimo veličinu tabele prilikom kreiranja GridPane-a**
- Jedna komponenta može da se prostire kroz više ćelija u tabeli
- **Metode za dodavanje čvora u GridPaneOkno**
  - `void add(Node child, int columnIndex, int rowIndex)`
  - ```
    void add(Node child, int column, int row,
             int columnSpan, int rowSpan)
    ```
  - **Obratiti pažnju: prvo navodimo indeks kolone, a onda indeks vrste**
    ```

```
public class DemoGridPane extends Application {
  public void start(Stage stage) throws Exception {
    GridPane container = new GridPane();
    container.setVgap(10);
    container.setHgap(10);
    container.setPadding(new Insets(20));
    for (int i = 0; i < 4; i++)
      for (int j = 0; j < 4; j++)
        if (i != j)
          container.add(new Button(i + " x " + j), i, j);

    Scene s = new Scene(container, 250, 150);
    stage.setScene(s);
    stage.setTitle("GridPane demo");
    stage.show();
  }

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



Tipovi događaja

- Svaki tip događaja je realizovan odgovarajućom klasom koja direktno ili indirektno nasleđuje klasu **javafx.event.Event**
- Neki od tipova događaja su (paket **javafx.event**)
  - KeyEvent
  - MouseEvent
  - MouseDragEvent
  - WindowEvent
  - TouchEvent, SwipeEvent, ZoomEvent
  - **ActionEvent – događaji visokog nivoa apstrakcije uključiv i logički (a ne fizički) klik na dugme**

Obrada događaja

- *Event handler* je objekat klase koji implementira interfejs **EventHandler<T extends Event>**
- Interfejs **EventHandler** propisuje implementaciju samo jednog metoda:
  void handle(T event)
- ***Event handler* je potrebno vezati za grafičku komponentu**

```
Button btn = new Button("Click me");

class ClickHandler implements EventHandler<ActionEvent> {
      @Override
      public void handle(ActionEvent event) {
            ...
      }
}

btn.setOnAction(new ClickHandler());
```

Anonimne metode

- Anonimna metoda je metoda koja nema ime
- Anonimne metode zadajemo lambda izrazima

```
(lista parametara) -> telo_metoda
```

  - Telo metoda je izraz ili blok naredba
  - Tipovi parametara se mogu izostaviti ukoliko kompajler može da ih sam zaključi (*type inference*)
  - Zagrade u listi parametara se mogu izostaviti ukoliko imamo jedan parametar naveden bez tipa

- Primeri.
  - `(String s, char c) -> s.charAt(0) == c`
  - `(s, c) -> s.charAt(0) == c`
  - `s -> s.length() + 5`

Anonimne metode i funkcijski interfejsi

- Funkcijski interfejs je interfejs koji propisuje implementaciju tačno jednog metoda

- **Ako je x promenljiva tipa X pri čemu je X funkcijski interfejs tada promenljivoj x možemo dodeliti anonimni metod**
  - x - referenca na funkcijski objekat

- Interfejs **EventHandler<T extends Event>** je funkcijski interfejs

```
EventHandler<ActionEvent> h = (ActionEvent e) -> {…}

EventHandler<ActionEvent> h = e -> {…}

btn.setOnAction(h)

btn.setOnAction(e -> {…})
```

Referencijalni lambda izrazi

- Referencijalni lambda izraz predstavlja anonimni metod koji samo pozove neki konkretan metod za sopstvene argumente

```
(x1, x2, ..., xk) -> a.m(x1, x2, …, xk)
```

- Referencijalni lambda izraze skraćeno možemo navesti kao a::m, pri čemu je *a* objekat (*m* - nestatički metod) ili klasa (*m* - statički metod)

```
public class JavaFxApp extends Application {
      private Button btn = new Button("Click me");

      public void start(Stage stage) throws Exception {
            ...
            btn.setOnAction(this::handleClick);
            ...
      }

      private void handleClick(ActionEvent e) {
            ...
      }
}
```