UNIVERZITET METROPOLITAN BEOGRAD

CS202 - OBJEKTNO-ORIJENTISANO PROGRAMIRANJE 2

Osnove JavaFX

Lekcija 01

PRIRUČNIK ZA STUDENTE

CS202 - OBJEKTNO-ORIJENTISANO PROGRAMIRANJE 2

Lekcija 01

OSNOVE JAVAFX

UVOD

Sadržaj i ciljevi ove lekcije

Sadržaj lekcije:

1. Okna, UI kotrola i oblici
2. Povezivanje svojstava
3. Zajednička svojstva i metodi za čvorove
4. Klasa Color
5. Klasa Font
6. Klase Image i ImageView
7. Okna rasporeda
8. Klasa Shapes
9. Studija slučaja: Klasa ClockPane
10. Vežba – Pokazni primeri
11. Vežba – Zadaci za samostalni rad
12. Zaključak

Ciljevi lekcije:.

1. Kreiranje korisničkog interfejsa korišćenjam okna (panes), kontrole korisničke interakcije, i oblika (shapes)
2. Automatsko ažuriranje vrednosti svojstava preko povezivanja svojstava.
3. Upotreba zajedničkih stilova svojstava i rotacija za čvorove.
4. Kreiranje boja upotrebnom klase Color
5. Kreiranje fontova upotrebom klase Font
6. Kreiranje slika ipotrebom klase Image i kreiranje poglede slika upotrebom klase ImageView
7. Čvorovi raspoređivanja upotrebom klasa Pane, StackPane, FlowPane, GridPane, BorderPane, Hbox i Vbox.
8. Prikazivanje teksta upotrebom klase Text i kreiranje oblika upotrebom Line, Circle, Rectangle, Elipse, Arc, Polygon i Polyline
9. Razvoju višestruko upotrebljive GUI komponente ClockPane za prikaz analognog sata

Poglavlje 1

Okna, UI kontrola i oblici

NAŠE POZORIŠTE

Okna (panes), UI kontrole, i oblici (shapes) su podtipovi od Node.

Čvor **(*Node*)** je vizualna komponenta oblik (***Shape***), pogled slike (***ImageView***), UI kontrola, ili okno (***Pane***).

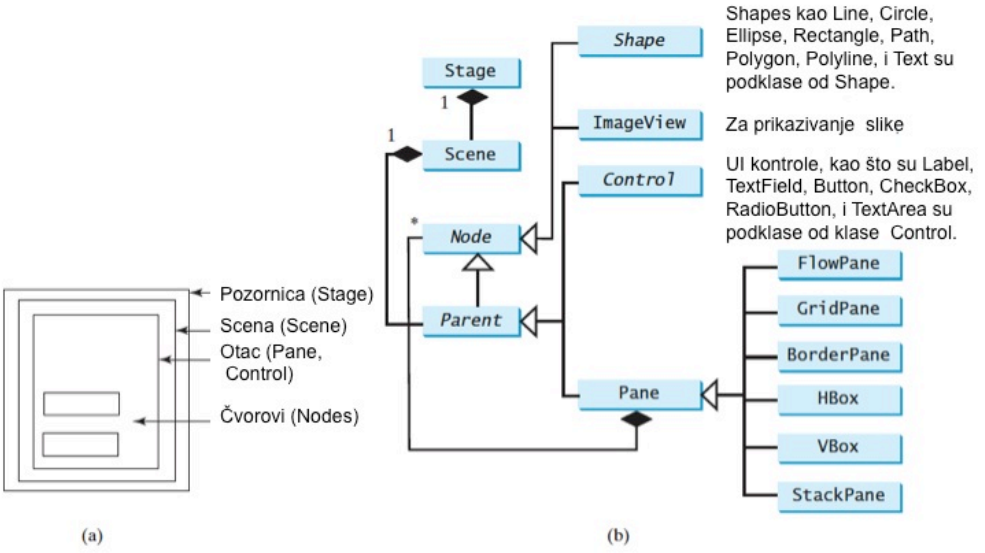
Okna **(**panes**)** su kontejner klase koje automatski raspoređuju čvorove na određene lokacije i sa određenim veličinama.

Oblik **(**shape**)** može da predstavlja tekst, liniju, krug, elipsu, luk, pravougaonik, poligon, polilinuju i dr.

UI kontrola **(**UO controls**)** se odnosi na nalepnicu (label), dugme (button), polje za potvrdu (check box), dugme opcoije (radio button), tekstualno polje (text field), polje za tekst u više redova (text area) i dr.

Scena **(*Scene*)** je prikaz na pozornici (***Stage***), kao što je prikazano na slici 1.a. UML dijagram klasa za Stage, Scene, Note, Control i Pane je dat na slici 1.b.

Jedna pozornica (Stage) može imati više scena (Scene). Scena sadrži ***Control*** ili ***Pane*** objekte, ali ne i ***Shape*** ili ***ImageView*** objekte. Okno, tj. ***Pane*** objekat, može da sadrži bilo koji podtip ***Node*** objekta. ***Scene*** objekat se kreira konstruktorima ***Scene(Parent, width, height***) ili ***Scene(Parent***). Podklase ***Node*** klase imaju konstruktore bez argumenata za kreiranje unapred definisanih elemenata.



Slika 1.1 .1: a) Java koordinatni sistem b) Krug u centru okna i scene c) Krug nije u centru promrnom veličine okna [1]

PRIMER: DUGME U CENTRU OKNA

Program postavlja dugme u centar okna.

Sledeći listing prikazuje program koji postavlja dugme u okno.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
import javafx.scene.layout.StackPane;

public class ButtonInPane extends Application {

    @Override // Redefinicija metoda start() klase Application
    public void start(Stage primaryStage) {
        // Kreiranje scene i stavljanje dugmenta u scenu e
        StackPane pane = new StackPane();
        pane.getChildren().add(new Button("OK"));
        Scene scene = new Scene(pane, 200, 50);
        primaryStage.setTitle("Button in a pane"); // Unos naziva pozornice
        primaryStage.setScene(scene); // STavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

Program kreira **StackPane** (linija 12) i dodaje dugme kao dete okvira (linija 13). Metod **getChilidren()** vraća primerak od ***javafx.collections.ObservableList***. ***ObesrvableList*** se ponaša kao ArrayList koji memoriše kolekciju elemenata.

Pozivom ***add(e)*** dodaje se element u listu.***StackPane*** postavlja čvorove u centar okna. U omom primeru se koristi samo jedan element. StackPane koristi predefinisanu veličinu čvora. Zato je dugme prikazano u predefenisanoj veličini.

Na slici 2 prikazan je dobijeni prozor izvršenjem programa ButtonInPane.

Slika 1.2 .2: Dugme je u centru okna [1]

PRIMER: KRUG U CENTRU OKNA

Postavljanje kruga u centru okna i scene.

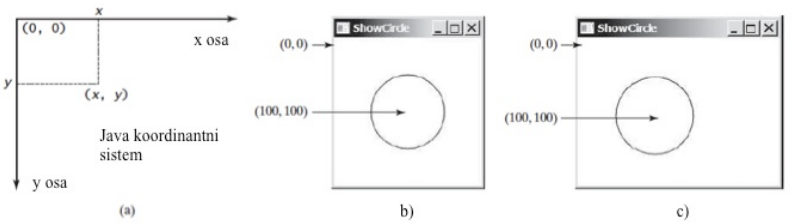
Listing pokazuje program koji postavlja krug u centar okna.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.stage.Stage;

public class ShowCircle extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje kruga i podešavanje njegovoh svojstava
        Circle circle = new Circle();
        circle.setCenterX(100);
        circle.setCenterY(100);
        circle.setRadius(50);
        circle.setStroke(Color.BLACK);
        circle.setFill(Color.WHITE);

        // Kreiranje okna koje sadrži krug
        Pane pane = new Pane();
        pane.getChildren().add(circle);

        // Kreiranje scene i postavljanje scene na pozornicu ge
        Scene scene = new Scene(pane, 200, 200);
        primaryStage.setTitle("ShowCircle"); // Unos naslova pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

Program kreira krug (linja 13) i postavlja ga u centar (100,100) (linije 14-15), što je i centar scene, jer je ona kreirana sa dimenzijama 200x200 (linija 24=. Poluprečnik kruga je 50 (linija 16). Sve jedinice su u pikselima. Linija kruga je crna (linija 17), a popunjen je belom bojom (linija 18).

Program kreira okno (linija 21) i postavlja krug u okno (linija 22). . Okno ima koordinate gornjeg levog ugla (0,0). To je koordinatni početak Java koordinatnog sistema (slika 5.a). . X-kooridnata ide s leva na desno, a z-koordinata ide odozgo nadole.

Okno je postavljeno na scenu (linija 24), a scena na pozornicu (linija 27), Krug je centru pozornice (slika 5.b). Ako se promeni veličina okna, krug onda nije više u njegovom centru.



Slika 1.3 .3: a) Java koordinatni sistem b) Krug u centru okna i scene c) Krug nije u centru promrnom veličine okna [1]

VIDEO - KREIRANJE OKVIRA ZA UZBUNU

JavaFX Java GUI Tutorial - 5 - Creating Alert Boxes (10,53 minuta)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 2

Povezivanje svojstava

POVEZIVANJE SVOJSTAVA CILJNOG I IZVORNOG OBJEKTA

Možete povezati ciljni i izvorni objekat. Svaka promena izvornog objekta odmah se prenosi na promenu i ciljnog objekta.

JavaFX uvodi novi koncept povezivanja svojstava koji omogućava ciljni objekat da bude povezan sa izvornim objektom. Kada se promeni vrednost izvornog objekta, promeni se i vrednost ciljnog objekta. Ciljni objekat se naziva povezanim (***binding***) objektom, a oba objekta se nazivaju povezanim (***bindable***) objektima.

U prethodnom listing, krug nije bio više u centru kada je promenjena dimenzija okna. Ako se povežu **centerX** kruga sa dimenzijom ***width*/2** i **centerZ** kruga sa dimenzijom ***height*/2** okna, kao što je pokazano u sledećem listingu.

Metod ***bind()*** povezuje ciljni objekt sa izvornim objektom:

```
target.bind(source);
```

Metod ***bind()*** je definisan u interfejsu ***javafx.beans.property.*** Property . Svojstvo povezivanja je primerak o klase ***javafx.beans.property.Property.*** Izvorni objekat je primerak interfejsa ***javafx.beans.value.ObservableValue.*** ***ObservableValue*** je entitet koji obuhvata vrednost i dozvoljava da se ona osmatra radi praćenja promena.

JavaFX definiše povezana svojstva za primitivne vrednosti i stringove. Za vrednost tipa double /float /long /int /boolean, njegova povezan tip je ***DoubleProperty /FloatProperty /***

LongProperty /IntegerProperty /BooleanProperty .

Za neki string, povezano svojstvo je ***StringProperty.***

Ova svojstva su takođe potipovi od ***ObservableValue.*** Zato, mogu da budu upotrebljeni i kao izvorni objekat za povezivanje svojstava.

Svako povezano svojstvo ima svoje geter i seter metode. Na primer za ***centerX*** svojstvo, metod je ***getCenterXProperty().***

PRIMER POVEZIVANJA SVOJSTAVA

Program kreira primerak od DoubleProperty upotrebom SimpleDoubleProperty()

Slika 1.a prikazuje konvenciju za definisanje svojstva povezivanja u nekoj klasi, a slika 1.b pokazuje primer u kome je centerX je svojstvo povezivanje tipa ***DoubleProperty***.

```
public class SomeClassName {

  private PropertyType x;

  /** Value getter method */
  public propertyValueType getX() { ... }

  /** Value setter method */
  public void setX(propertyValueType value) { ... }

  /** Property getter method */
  public PropertyType
    xProperty() { ... }
}
```

(a) X je svojstvo povezivanja

```
public class Circle {

  private DoubleProperty centerX;

  /** Value getter method */
  public double getCenterX() { ... }

  /** Value setter method */
  public void setCenterX(double value) { ... }

  /** Property getter method */
  public DoubleProperty centerXProperty() { ... }
}
```

(b) centerX je svojstvo povezivanja

Slika 2.1 .1: Svojstvo povezivanja ima geter i seter vrednosti, i geter metod svojstva [1]

Ovde se prikazuje listing drugog primera povezivanja svojstava ***BindingDemo***. Program kreira primerak od DoubleProperty upotrebom SimpleDoubleProperty(1). Program povezuje **d1** sa **d2** (linija 9). Sada su vrednosti za d1 i d2 iste. Posle unosa vrednosti za d2 od 70.2 (linija 12), i vrednost za d1 postaje 70.2 (linija 14). Ovo je primer **unidirekcionog povezivanja**. Ako se promena svojstva prenosi u oba smera, onda je to **bidirekciono povezivanja** promenom ***metoda bindBidirectional().***

```
import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

public class BindingDemo {

    public static void main(String[] args) {
        DoubleProperty d1 = new SimpleDoubleProperty(1);
        DoubleProperty d2 = new SimpleDoubleProperty(2);
        d1.bind(d2);
        System.out.println("d1 is " + d1.getValue()
                + " and d2 is " + d2.getValue());
        d2.setValue(70.2);
        System.out.println("d1 is " + d1.getValue()
                + " and d2 is " + d2.getValue());
    }
}
```

Dobijen rezultat:

d1 is 2.0 and d2 is 2.0

d1 is 70.2 and d2 is 70.2

VIDEO - KOMUNIKACIJA IZMEĐU PROZORA

JavaFX Java GUI Tutorial - 6 - Communicating Between Windows

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 3

Zajednička svojstva i metodi za čvorove

KLASA NODE

Apstraktna klasa Node definiše mnoga svojstva i metode koji su zajedničke za sve čvorove.

Čvorovi dele mnoga zajednička svojstva. Ovde ćemo analizirati dva takva svojstva stila i svojstva rotacije.

JavaFX svojstva stila su slične sa CSS (cascade style sheets) koji specificiraju stilovi za HTML elemente na veb stranice. Zato, svojstva stilova u JavaFX se nazivaju JavaFX CSS. U JavaFX svojstva stilova se definišu sa prefiksom –fx. Svaki čvor ima svoje sopstvene stilove. Detaljniju informaciju možete naći na adresi: http://docs.oracle.com/javafx/2/api/javafx/scene/doc-files/cssref.html

Sintaksa za postavljanje stila je ***styleName*** vrednost. Višestruka svojstva nekog čvora se može podesiti zajedno, ali odvojeni sa (;). Na primer sledeći iskaz:

```
circle.setStyle("-fx-stroke: black; -fx-fill: red;");
```

Postavlja dva svojstva stilova za krug. Ovaj iskaz je ekvivalentan sa sledećim iskazima:

```
circle.setStroke(Color.BLACK);
circle.setFill(Color.RED);
```

Ako se JavaFX CSS nepravilno koristi, program će raditi, ali će ignorisati stilove.

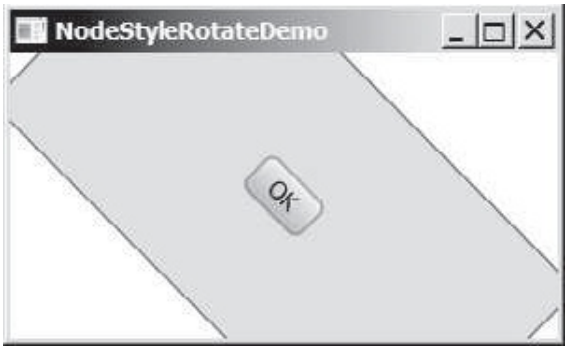
Svojstvo ***rotate*** omogućava vam da specificirate ugao u stepenima za rotaciju čvora oko centra. Pozitivna vrednost ugla označava ugao u smeru sklazaljke na satu. U suprotnom, kreće se suprotno okretanju skazaljke na satu. Na primer, sledeći iskaz rotira dugme za 80 stepeni.

```
button.setRotate(80);
```

PRIMER ZA SVOJSTVO ROTACIJE

Primer prikazuje program koji kreira dugme, postavlja njegov stil, i stavlja ga u okno, a onda ga rotira za 45 stepeni, i postavlja crvenu boju linije na granici i boju pozadine dugmeta.

Sledeći listing prikazuje program koji kreira dugme, postavlja njegov stil, i stavlja ga u okno. Onda se rotira za 45 stepeni i postavlja stil za crvenu boju linije na granici i boju pozadine svetlo sive, kao što je pokazano na slici 1.



Slika 3.1 .1: Postavljen stil okvira i zarotiran za 45 stepeni [1].

Kao što se vidi, rotacijom okna, zarotiralo se i dugme.

Klasa ***Node*** sadrži mnoge korisne metode koji se mogu primeniti na sve čvorove. Na primer, možete iskoristiti metod ***contains(double x, double y)*** za testiranje da li je tačka (x, y) unutar granica čvora.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.stage.Stage;
import javafx.scene.layout.StackPane;

public class NodeStyleRotateDemo extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje scene i postavljanje dugmeta u scenu
        StackPane pane = new StackPane();
        Button btOK = new Button("OK");
        btOK.setStyle("-fx-border-color: blue;");
        pane.getChildren().add(btOK);

        pane.setRotate(45);
        pane.setStyle(
                "-fx-border-color: red; -fx-background-color: lightgray;");

        Scene scene = new Scene(pane, 200, 250);
        primaryStage.setTitle("NodeStyleRotateDemo"); // Unos naziva scene
```

```
        primaryStage.setScene(scene); // Stavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

VIDEO - SVOJSTVA

JavaFX Java GUI Tutorial - 28 - Properties (11,16)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO - POVEZIVANJE SVOJSTAVA

JavaFX Java GUI Tutorial - 29 - Binding (4,35)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO - PRIMER POVEZIVANJE SVOJSTAVA

JavaFX Java GUI Tutorial - 30 - Binding Properties Example (5,12 minuta)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 4

Klasa Color

SVOJSTVA KLASE COLOR

Klasa Color služi za kreiranje boja

JavaFX koristi apstraktnu klasu ***Paint*** za bojenje nekog čvora. Klasa ***javafx.scene.paint.Color*** je konkretna podklasa klase ***Paint*** koja sadrži metode za definisanje boja (slika 1).

Klasa ***Color*** koristi tzv. ***RGBA*** model, koji definiše neku boju kombinacijom crvene (R), plave (B), zelene (G) boje i stepena prozirnosti – alfa vrednost (A). Objekat ***Color*** je nepromenljiv, posle kreiranja, jer mu se svojstva ne mogu menjati. Primenom metoda ***brighter()*** i ***darker()*** menjaju se vrednosti R, G i A, ali se vraća novi ***Color*** objekat, sa istom vrednosti prozračnosti A.

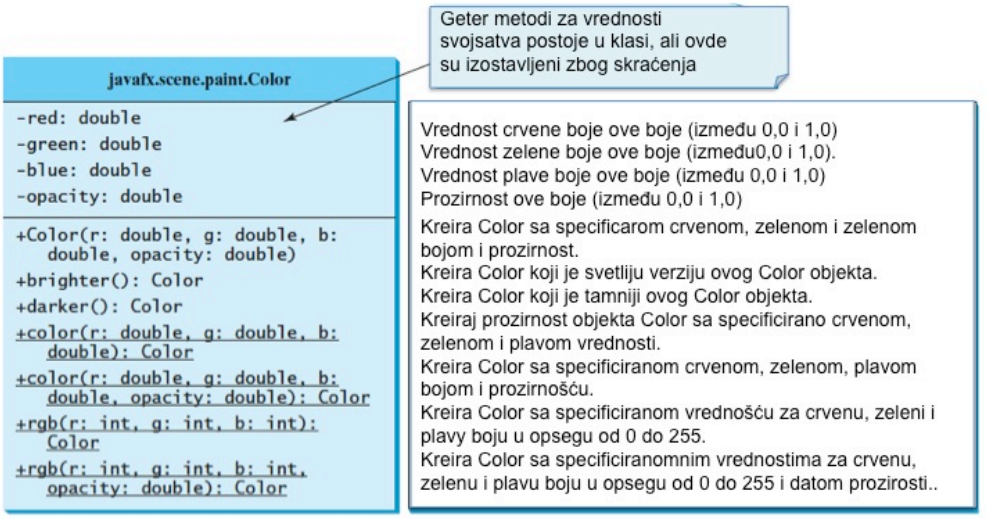
Objekat ***Color*** se može dobiti i sledećim kontruktorima***: color(r, g, b), color(r,g, b, opacity), rgb(r, g, b), and rgb(r, g, b, opacity).***Postoji i 18 unapred definisanih boja. Na primer:

```
circle.setFill(Color.RED);
```

Objekat Color se kreira primenom sledećeg konstruktora:

```
public Color(double r, double g, double b, double opacity);
```

Gde argumenti r, g i b određuju vrednost svojstva za crvenu, zelenu i plavu boju, gde vrednosti se kreću između 0.0 (najtamnija) i 1.0 (najsvetlija). Argument opacity of+značava prozirnost, i kreće se od 0.0 (prozirno) do 1.0 (neprozirno).



Slika 4.1 .1: Klasa Color sadrži informacije o bojama [1]

Poglavlje 5

Klasa Font

SVOJSTVA KLASE FONT

Klasa Font definiše ime fonta (vrste slova) i njegovu postavku i veličinu

Klasa ***javafx.scene.text.Font*** služi za kreiranje fontova. Objekat klase Font se kreira konstruktorima ili statičkim metodima. Statički metod ***getFamilies()*** daje listu raspoloživih fontova (npr. ***Ariel, Times New Roman, Courier***), u vidu objekta ArrayList.

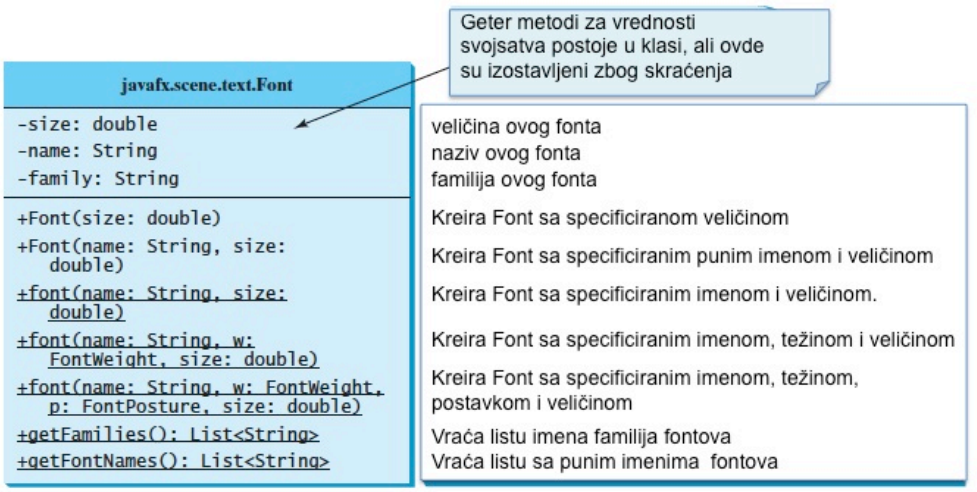
Objekat ***Font***, pored naziva fonta i njegove veličine, definiše i njegovu postavku (posture) te može biti:sa:

- *kosim slovima* ili *kurzivom* ***(italic***)
- **izraženim slovima** (***bold***) i
- normalnim slovima (***regular***)

Postavka slova se definiše izrazom: ***FontPosture.ITALIC*** ili ***FontPosture.BOLD*** ili ***FontPosture.REGULAR.***

Na primer, sledeći iskazi kreiraju dva fonta:

```
Font font1 = new Font("SansSerif", 16);
 Font font2 = Font.font("Times New Roman", FontWeight.BOLD,
      FontPosture.ITALIC, 12);
```



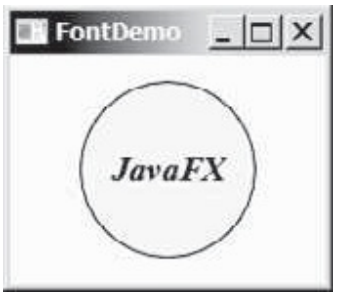
Slika 5.1 .1: Klasa Font sadrži informacije o fontovima [1]

PRIMER DEFINISANJA FONTOVA

Postavljanje kruga sa nalepnicom "JavaFX" u centar scene.

Na slici 2 prikazan je krug sa nalepnicom JavaFX koji je postavljen u centar scene. Treba koristiti sledeći font: Times New Roman, izraćena slova, kurziv, i veličina 20.

.



Slika 5.2 .2: Nalepnica je stavljena preko kruga kojije u centru scene [1]

Program kreira okvir StackPane (linija 15) i dodaje mu krug i nalepnicu (oznaku) (linije 22 i 28). Ta dva iskaya se mogu zameniti i sa jednim iskazom:

```
pane.getChildren().addAll(circle, label);
```

StackPlan postavlja čvorove u centar, a njih stavlja jedan na drugi. Postavlja se popuna kruga bojom (linija 21). Kreira se nalepnica i njen font (linija 26). Promenom dimenzija prozora ostaju u centru jer je ***StackPane*** automatski postavlja čvorove u centar.

Daje se listing programa koji to realizuje

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.*;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.text.*;
import javafx.scene.control.*;
import javafx.stage.Stage;

public class FontDemo extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna u koje se smešta krug
        Pane pane = new StackPane();

        // Kreiranje kruga i unos njegovih svojstava
        Circle circle = new Circle();
        circle.setRadius(50);
        circle.setStroke(Color.BLACK);
```

```
        circle.setFill(new Color(0.5, 0.5, 0.5, 0.1));
        pane.getChildren().add(circle); // Dodavanje kruga u okno

        // Kreiranje nalepnice i unos njenih svojstava
        Label label = new Label("JavaFX");
        label.setFont(Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.ITALIC, 20));
        pane.getChildren().add(label);

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane);
        primaryStage.setTitle("FontDemo"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Stavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

Poglavlje 6

Klase Image i ImageView

SVOJSTVA KLASA IMAGE I IMAGEVIEW

Klasa Image predstavlja grafičku sliku, a klasa ImageView prikazuje sliku.

Klasa ***javafx.scene.image.Image*** predstavlja grafičku sliku i koristi se za unošenje slike sa specificirane datoteke ili sa URL. Primeri:

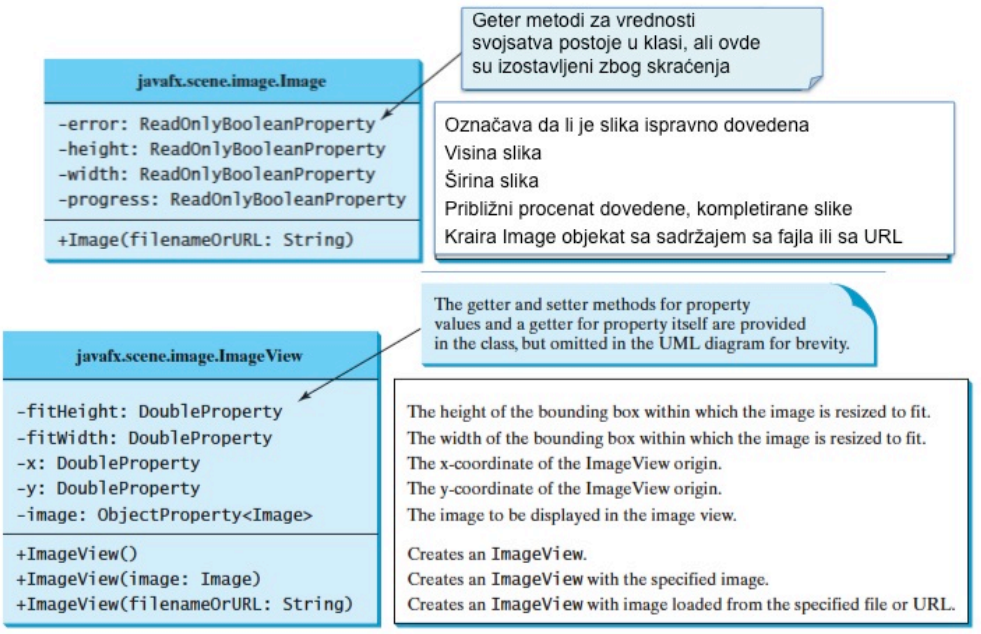
- ***Image***("image/us.gif")
- ***Image***("http://www.cs.armstrong.edu/liang/image/us.gif")

Klasa ***javafx.scene.image.ImageView*** je čvor za prikazivanje neke slike. ***ImageView*** se može kreirati iz ***Image*** objekta. Na primer:

```
Image image = new Image("image/us.gif");
 ImageView imageView = new ImageView(image);
```

Alternativno, može se kreirati ***ImageView*** direktno iz datoteke ili sa URL:

```
ImageView imageView = new ImageView("image/us.gif");
```

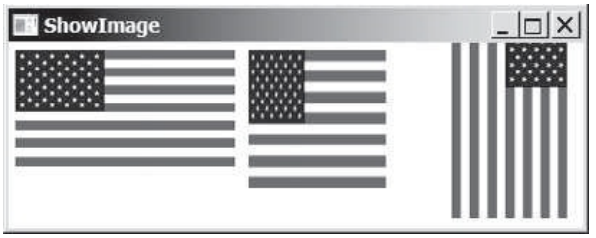


Slika 6.1 .1: Klase Image i ImageView [1]

PRIMER PRIKAZA SLIKE

Prikazuju se tri ImageView objekta koji prikazuju istu sliku, tj. objekat Image.

Na slici 2 prikazan je prozor sa tri slike. Prikazan listing prikazuje program koji realizuje prikaz ove tri slike.



Slika 6.2 .2: Slika koja sadrži tri prikaza slika u oknu [1]

Program kreira ***Hbox*** (linija 14) . To ja okno koji horizontalno postavlja čvorove u jednom redu. Program kreira ***Image*** i onda i ***ImageView*** za prikaz***Image***. I postavlja ***ImageView*** u ***Hbox*** (linija 17) Program kreira i drugi ***ImageView*** (linija 19), postavlja svojstva ***fitHight*** i ***fitWidth*** (linije 20-21) i postavljanja ***ImageView*** u ***Hbox*** (linija 22). Program kreira i treći ***ImageView*** (linija 24), rotira ga za 90 stepeni (linija 25) i postavlja ga u ***Hbox*** (linija 26). Metod ***setRotate*** je definisan u klasi ***Note*** i može se koristite za bilo koji čvor. Image se ovde deli od strane tri ***ImageView***, ali se čvor ***ImageView*** ne može da deli. Ne može se postaviti više puta.

Datoteka sa slikom mora da bude u istim direktorijumu sa class datotekom. UL mora da sadrži i http:// deo adrese.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.geometry.Insets;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
public class ShowImage extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna za smeštaj slike
        Pane pane = new HBox(10);
        pane.setPadding(new Insets(5, 5, 5, 5));
        Image image = new Image("image/us.gif");
        pane.getChildren().add(new ImageView(image));

        ImageView imageView2 = new ImageView(image);
        imageView2.setFitHeight(100);
```

```
        imageView2.setFitWidth(100);
        pane.getChildren().add(imageView2);

        ImageView imageView3 = new ImageView(image);
        imageView3.setRotate(90);
        pane.getChildren().add(imageView3);

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowImage"); // Unos naslova pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

VIDEO - ZATVARANJE SVOJSTAVA PROGRAMA

JavaFX Java GUI Tutorial - 7 - Closing the Program Properly (8,21 minuta)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 7

Okna rasporeda

UVOD

JavaFX obezbeđuje više okna koja automatski raspoređuju čvorove na željene lokacije sa željenom veličinom.

JavaFX obezbeđuje puno tipova okna za organizovanje čvorova u kontejneru:

- ***Pane*****:** Osnovna klasa za okna za raspoređivanje. Sadrži metod ***getChilidren*** koji vraža listu čvorova u okviru.
- ***StackPane*****:** Postavlja čvorove jedan iznad drugog u centru okvira.
- ***FlowPane*****:** Postavlja čvorove iz reda u red po horizontali ili kolona po kolona po vertikali.
- ***GridPane*****:** Postavlja čvorove u ćelije dvodimenzione rešetke (matrice).
- ***BorderPane***: Postavlja čvorove na vrh, desno, dole, levo i u centar okna.
- ***Hbox*****:** Postavlja čvorove u jedan red (horizontalu).
- ***Vbox***: Postavlja čvorove u jednu kolonu (vertikalu)

Okno (***pane***) se koristi kao platno (canvas) za prikazivanje **oblika** (shapes). ***Pane*** je osnovna klasa za sve ostale specijalizovane (pot)klase za okna. Svaki okvir sadrži listiučvorova koje sadrži. Ta lista je primerak klase ***ObservableList*** koji se dobija pozivom metoda ***getChilidren()***. Sa metodom *add(node)* mogu se čvorovi dodavati u okno, tj. u listu, a sa metodom *addAll(node1, node2, ...)* dodaje promenljivi broj čvorova u okvir.

7.1 FlowPane

KLASA FLOWPANE

FlowPane je okvir koji ređa čvorove po horizontali (red po red) ili po vertikali (kolona po kolona).

FlowPane ređa čvorove u oknu horizontalno, sleva udesno, ili vertikalno, odozgo nadole, po redosledu njihovog dodavanja. Kada se popuni jedan red ili kolona, počinje ređanje u drugom redu ili koloni. Primenom dve konstante:

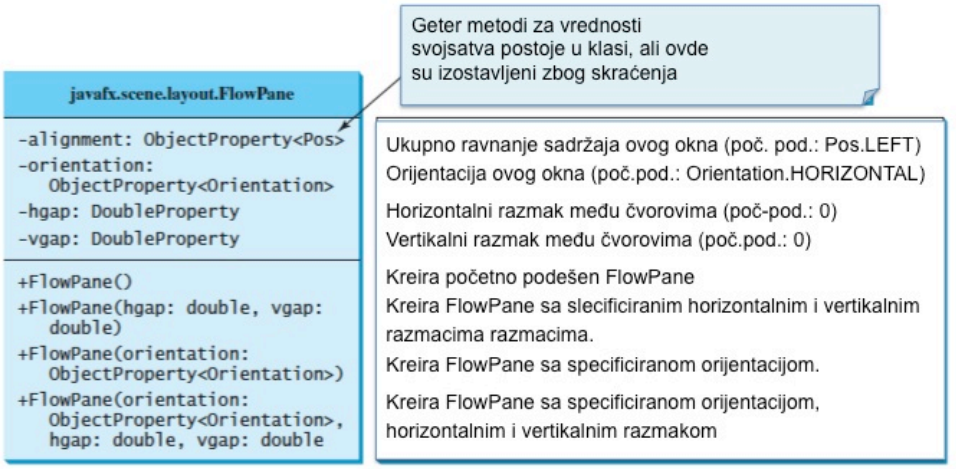
- ***Orientation.HORIZONTAL***
- ***Orientation.VERTICAL***

Određujete da li želite ređanje po horizontali ili vertikali.

Možete da specificirate i razmak između čvorova u pikselima (i po horizontali i po vertikali.

Ravnanje polja sa podacima, razmaci među njima **hgap** i **vgap**, su povezana svojstva (binding properties). Svaki od njih ima svoj getter metod, na primer, ***getHgap()*** koji vraća svoju vrednost, a setter metod, npr., ***setGap(double***) unosi željenu vrednost, a geter metod vraća

vrednost svojstva (na primer, ***hGapeProperty().*** Za polje podataka tipa ObjectProperty<T> vrednosni geter metod vraća vrednost tipa T , a geter metod za svojstva, vraća vrednost svojstva tipa ObjectProperty<T>.



Slika 7.1.1 .1: FlowLayout raspoređuje čvorove po horizontali ili po vertikali [1]

PRIMER PRIMENE FLOWPANE

Zadatak je da se dodaju natpisi i tekstualna polja u okno FlowPane

Potrebno je dodati natpise i tekstualna polja u ***FlowPane*** okno (slika 2).

Slika 7.1.2 .2: Čvorivi se ređaju horizontalno, red po red [1]

Program kreira***FlowPane*** (linija 13) i postavlja svoje svojstvo ispunjenja (padding) sa objektoma***Insets*** (linija 14). On specificira veličinu i granicu okvira.

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.FlowPane;
```

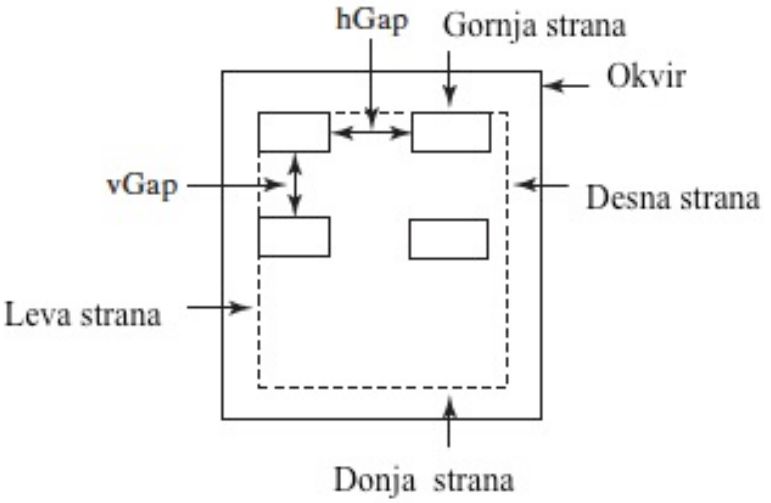
```
import javafx.stage.Stage;
public class ShowFlowPane extends Application {

    @Override // Redefinisanje start metoda klase Aplikacije
    public void start(Stage primaryStage) {
        // Kreiranje okvira i unos njegovih svojstava
        FlowPane pane = new FlowPane();
        pane.setPadding(new Insets(11, 12, 13, 14));
        pane.setHgap(5);
        pane.setVgap(5);

        // Postavljanje čvorova u okviru
        pane.getChildren().addAll(new Label("First Name:"),
                new TextField(), new Label("MI:"));
        TextField tfMi = new TextField();
        tfMi.setPrefColumnCount(1);
        pane.getChildren().addAll(tfMi, new Label("Last Name:"),
                new TextField());

        // Kreiranje scene i njeno postavljenje  na pozornici
        Scene scene = new Scene(pane, 200, 250);
        primaryStage.setTitle("ShowFlowPane"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

NAČIN RASPOREĐIVANJA ČVOROVA U OKNU

Objekat Insets definiše veličinu granice okna. Svojstva hGap i vGap definišu razmake susednih čvorova u horizontalnom i vertikalnom pravcu.

Konstruktor za***Insets(11,12,13,14***) kreira ***Insets*** sa veličinom granice gore (11), desno (12), dole (13) i levo (14) u pikselima, kao što je pokazano na slici 3. Možete koristiri i konstruktor ***Insets***(value) koji kreira okno sa jednakom granicom na sve četiri strane. Svojstva ***hGap*** i ***vGap*** u linijama 15-16 specificiraju horizontalni i vertikalni razmak između dva čvora, kao što je pokazano na slici 3.

.

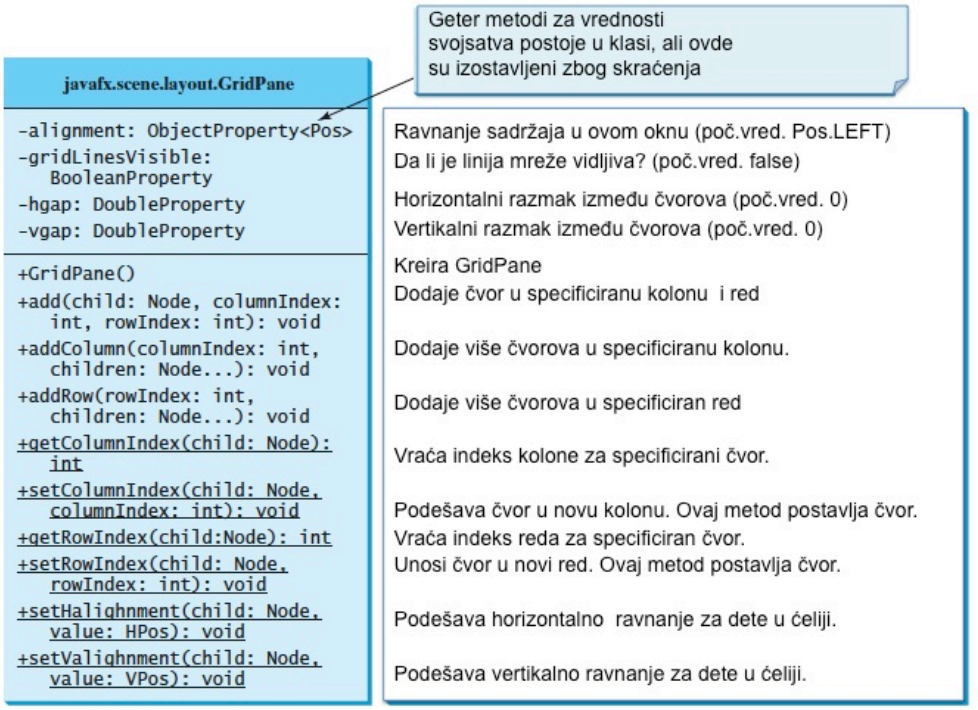


Slika 7.1.3 .3: Razmaci hGap i vGap između čvorova. [1]

7.2 GridPane

KLASA GRIDPANE

GridPane uređuje čvorove u matričnom obliku



Slika 7.2.1 .1: GridPane rsporeduje čvorove u ćelije matrice [1]

PRIMER KORIŠĆENJA GRIDPANE OKNA

Program kreira dva okna, a po dve kolone i četri reda primenom GridPane okna.

Na slici 2 su prikazana dva okna. Svaki sa po tri natpisa i tri tekstualna polja i sa jednim dugmetom. Listing prokazuje program koji ih kreira.

Slika 7.2.2 .2: GridPane raspoređuje čvorove u kolone i redove [1]

Program kreira GridPane (linija 16) i postavlja svojstva (linije 17-20). Ravnanje se postavlja u centar (linija 27), te i čvor se postavlja u centar okna. Ako promenite veličinu prozora, čvor ostaje u centru.

Program dodaje natpis u kolonu 0 i u red 0 (linija 23). Indeksi kolone i redova počinju od 0. Ne mora se svaka ćelija matrice da se popuni. Dugme se stavlja u kolonu 1 i red 3 (linija 30), a nema čvorova u kolonu 0 i redu 3. Uklanjanje čvora bi se radilo metodom pane.getChilidren().remove(node). Uklanjenje svih čvorova se radi metodom pane.getChilidren().removeAll().

Program poziva statički metod setHalignment kojim se dugme ravna u ćeliji (linija 30).

Veličina scene nije uneta (linija 34), već se automatski određuje.

```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;
public class ShowGridPane extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna i unos njegovih svojstava
        GridPane pane = new GridPane();
        pane.setAlignment(Pos.CENTER);
        pane.setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
        pane.setHgap(5.5);
```

```
        pane.setVgap(5.5);

        // Postavljanje čvorova u okno
        pane.add(new Label("First Name:"), 0, 0);
        pane.add(new TextField(), 1, 0);
        pane.add(new Label("MI:"), 0, 1);
        pane.add(new TextField(), 1, 1);
        pane.add(new Label("Last Name:"), 0, 2);
        pane.add(new TextField(), 1, 2);
        Button btAdd = new Button("Add Name");
        pane.add(btAdd, 1, 3);
        GridPane.setHalignment(btAdd, HPos.RIGHT);

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowGridPane"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

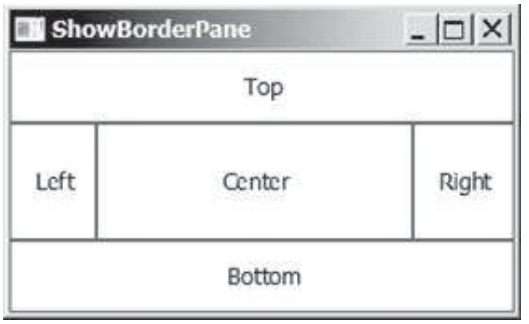
7.3 BorderPane

KLASA BORDERPANE

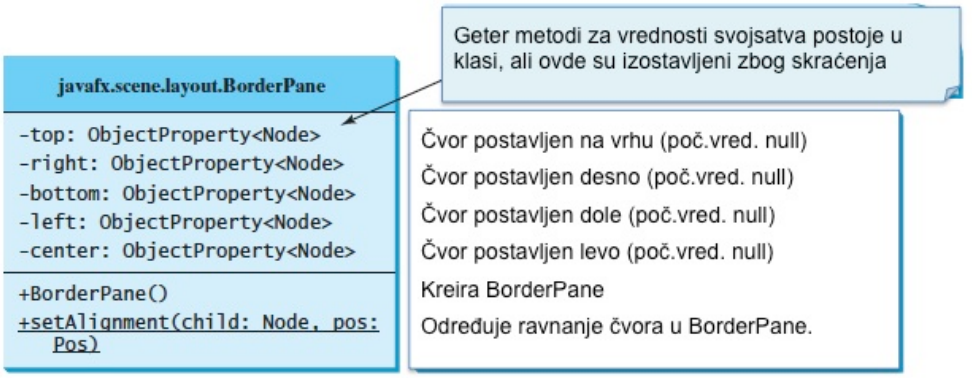
BorderPane postavlja čvorove u pet delova okna: gore, dole, levo, desno i u centar

BorderPane postavlja čvorove u pet delova okna: gore, dole, levo, desni u centar (slika 1), upotrebom metoda: ***setTop(node) , setBottom(node) , setLeft(node) , setRight(node)*** , i***setCenter(node***)

UML dijagram klase je dat na slici 2.



Slika 7.3.1 .1: BorderPane raspoređuje čvorove gore, dole, levo, desno i u centar



Slika 7.3.2 .2: Klasa BorderPane

PRIMER PRIMENE OKNA BORDERPANE

Listing programa ShowBorderPane pokazuje upotrebu okna BorderPane raspoređujući pet dugmeta u pet delova okna

Listing programa **ShowBorderPane** pokazuje upotrebu okna ***BorderPane*** raspoređujući pet dugemta u pet delova okna (slika 1)

Program definiše ***CustomPane*** koji proširuje***StackPane*** (linija 31) . Konstruktor za ***CustomPane*** dodaje natpis sa specificiranim nazivom (linija 33), definiše stilove za boju granice. I postavlja popunjenost (padding) upotrebom objekta Insets (linija 35).

Program kreira ***BorderPane*** (line 13) i postavlja per primeraka objekta ***CustomPane*** u pet delova okna s granicom (linije 16-20). Vidi se da je okno takođe i čvor. Zato, okno se može dodati u drugo okno. Za uklanjanje čvora iz gornjeg dela, treba pozvati ***setTop(null***). Ako neki deo (region) nije zauzet, ne rezerviše se prostor za taj deo (region).

.

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
public class ShowBorderPane extends Application {

    @Override //Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje BorderPane objekta
        BorderPane pane = new BorderPane();

        // Postavlja čvor u okno
        pane.setTop(new CustomPane("Top"));
```

```
        pane.setRight(new CustomPane("Right"));
        pane.setBottom(new CustomPane("Bottom"));
        pane.setLeft(new CustomPane("Left"));
        pane.setCenter(new CustomPane("Center"));

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowBorderPane"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage
    }
    // Definisanje okna koji sadrži natpis u centru okvira.
    class CustomPane extends StackPane {

        public CustomPane(String title) {
            getChildren().add(new Label(title));
            setStyle("-fx-border-color: red");
            setPadding(new Insets(11.5, 12.5, 13.5, 14.5));
        }

    }

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

7.4 HBox i VBox

KLASE HBOX I VBOX

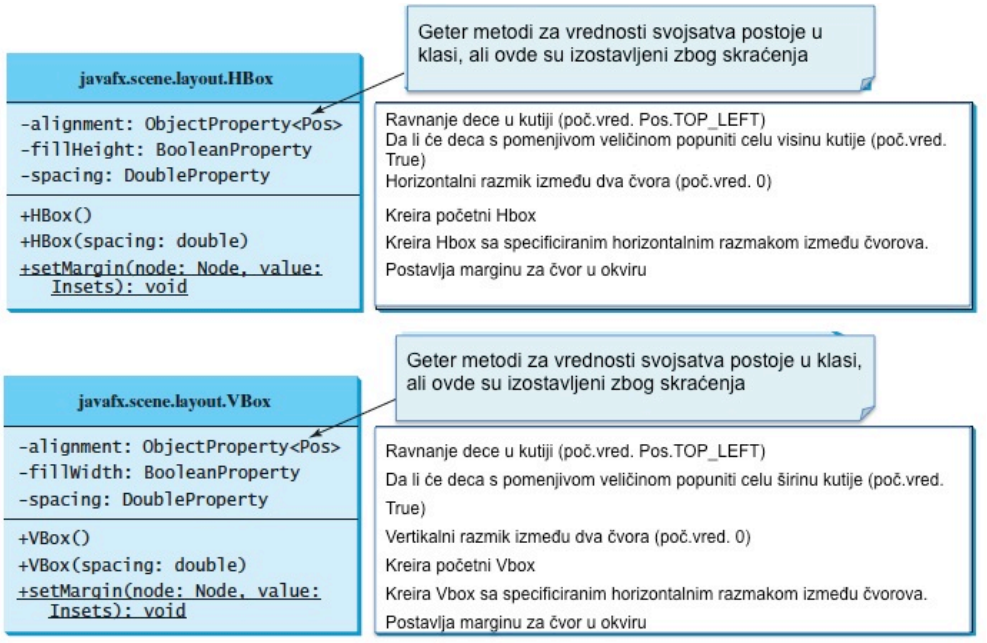
HBox i VBox sve čvorove stavljaju u jedan red, odnosno u jednu kolonu

HBox i VBox su okna koja sadrže, kao kontejneri, čvorove, koja se često nazivaju i decom, jer su vezani sa oknom vezom tipa “sadrži”.

HBox postavlja svoju decu u jedan horizontalni red.***VBox*** postavlja svoju decu i jednu vertikalnu kolonu.

UML klasni dijagrami za ***HBox*** i ***VBox*** dati su na slici 1.

.



Slika 7.4.1 .1: Klase HBox i VBox koje postavljaju čvorove u jedan red, odn. U jednu kolonu [1]

PRIMER PRIMENE HBOX I VBOX OKANA

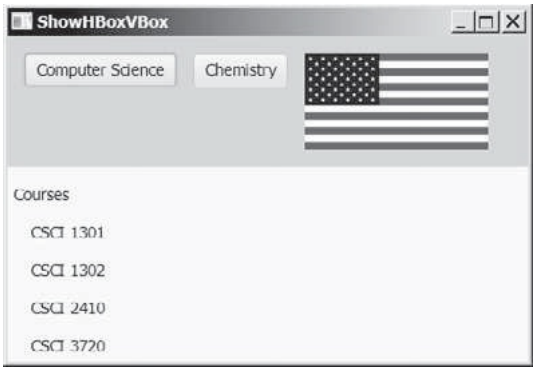
Primer dobijanja prozora sa dva okna. HBox okno treba da obezbedi dva dugmeta i prikaz zastave. Okno VBox treba da prikaže pet natpisa u jednoj koloni.

Potrebno je napraviti program čijim izvršenjem dobija se prozor sa Hbox oknom sa dva dugmeta i Vbox okno sa pet natpisa, kao što je pokazano na slici 2.

Program definiše metod getHBox() metod. On vraža Hbox koji sadrži dva dugmeta i prikaz sliku (linije 30-39). Boja pozadine okna Hbox je zlatna, a određena je upotrebom Java CSS (linija 33).

Program definiše metod getVBox() koji vraća Vbox okno koji sadrži pet natpisa (linije 41-55). Prvi natpis se dodaje u Vbox okno u liniji 44, a drugih četiri natpisa se dodaju u liniji 51.

Metod setMargin se upotrebljava za postavljenje margine čvora kada se postavlja unutar Vbox (linija 50).



Slika 7.4.2 .2: HBox postavljaju čvorove u jedan red, a VBox - u jednu kolonu [1]

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
public class ShowHBoxVBox extends Application {

    @Override // Redefiniše se start metod u klasi Application
    public void start(Stage primaryStage) {
        // Kreira okno sa granicom e
        BorderPane pane = new BorderPane();

        // Postavljanje čvorova u okbu
        pane.setTop(getHBox());
        pane.setLeft(getVBox());

        // Kreira scenu i stavlja je na pozornicu
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowHBoxVBox"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Postavlja scenu
        primaryStage.show(); // Prikazuje pozornicu
    }

    private HBox getHBox() {
        HBox hBox = new HBox(15);
        hBox.setPadding(new Insets(15, 15, 15, 15));
        hBox.setStyle("-fx-background-color: gold");
        hBox.getChildren().add(new Button("Computer Science"));
        hBox.getChildren().add(new Button("Chemistry"));
        ImageView imageView = new ImageView(new Image("image/us.gif"));
        hBox.getChildren().add(imageView);
        return hBox;
    }
```

```
    private VBox getVBox() {
        VBox vBox = new VBox(15);
        vBox.setPadding(new Insets(15, 5, 5, 5));
        vBox.getChildren().add(new Label("Courses"));

        Label[] courses = {new Label("CSCI 1301"), new Label("CSCI 1302"),
            new Label("CSCI 2410"), new Label("CSCI 3720")};

        for (Label course : courses) {
            VBox.setMargin(course, new Insets(0, 0, 0, 15));
            vBox.getChildren().add(course);
        }

        return vBox;
    }

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

VIDEO - RAD SA RASPOREDIMA

JavaFX Java GUI Tutorial - 8 - Embedding Layouts (6,25 minuta)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO - GRIDPANE

JavaFX Java GUI Tutorial - 9 - GridPane (11,15 minuta)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

Poglavlje 8

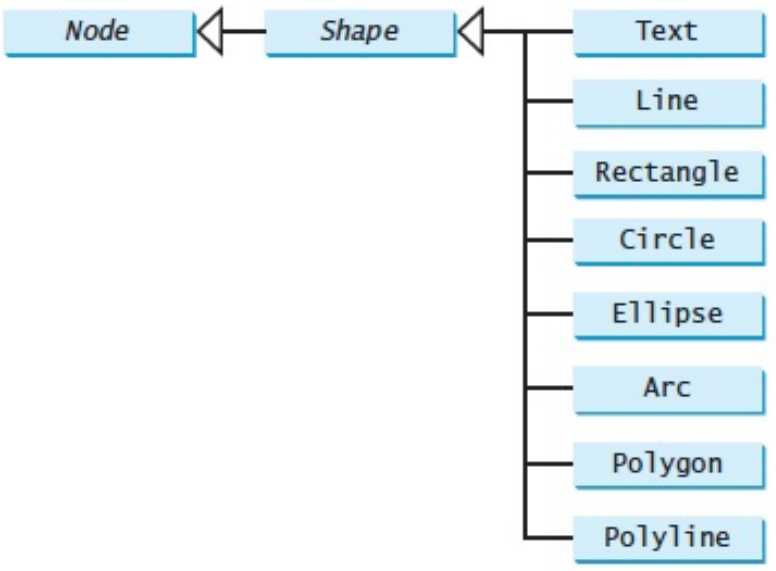
Klasa Shapes

UVOD

JavaFX obezbeđuje puno klasa oblika (shapes) koje crtaju tekstove, linije, krugove, pravougaonike, elipse, lukove, poligone i polilinije

Klasa Shape je apstrkna klasa koja definiše zajednička svojstva za sve oblike. Ta svojstva određuju boju popune površine oblika, boju i izgled granične linije. Svojstvo popune površine specificira boju koja popunava unutrašnjost oblika. Svojstvo izgleda granične linije (stroke) specificira boju linije obika. Svojstvo strokeWith specificira širinu linije oblika.

Ovde će se prikazati klase ***Text***,***Line***,***Rectangle***,***Circle***,***Elipse***,***Arc***,***Polygon*** i***Polyline*** za crtanje tekstova i jednostavne oblike, UML dijagram ovih klasa je prikazan na slici 1. .

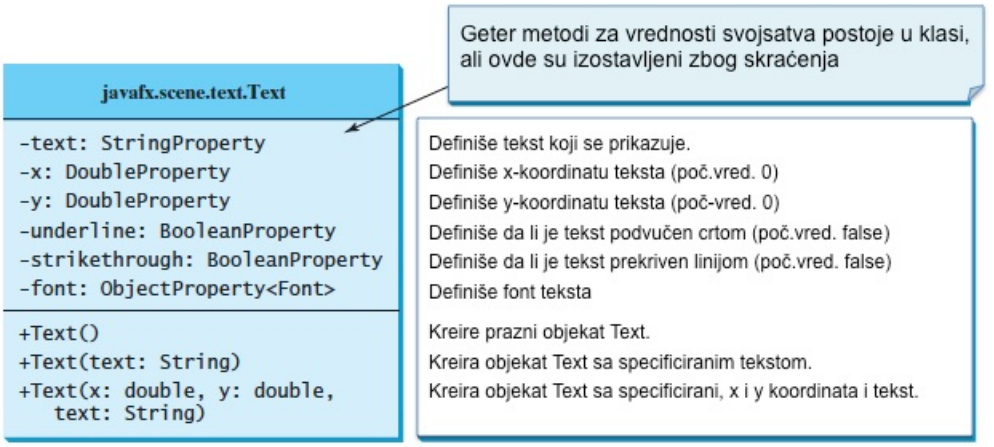


Slika 8.1.1 .1: UML dijagram klasa koje definišu osnovne oblike čvorova [1]

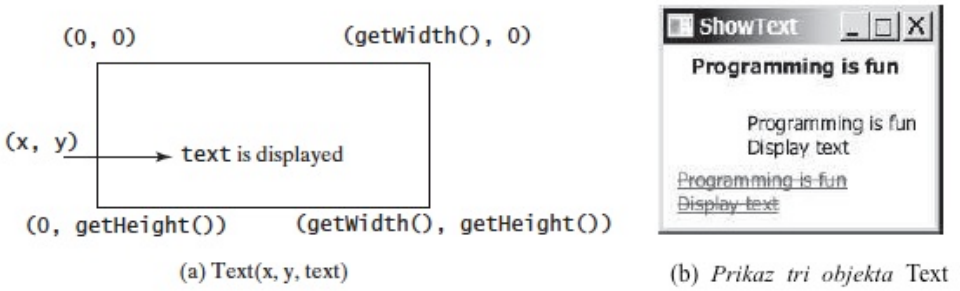
8.1 Klasa Text

SVOJSTVA KLASE TEXT

Klasa Text prikazuje željeni tekst u određenoj poziciji okna.



Slika 8.2.1 .1: UML dijagram klase Text [1]



Slika 8.2.2 .2: Objekat Text prikazuje željeni tekst [1]

PRIMER KORIŠĆENJA KLASE TEXT

Program treba da prikaže u oknu četiri teksta sa različitim svojstvima (naglašen, u više redova, podvučen, prekriven crtom).

Ovde je prikazan listing programa **ShowText** čijim izvršenjem se dobija prikaz tekstana slici 2.b.

Program kreira **Text** objekat (linija 18), određuje mu font (linija 19) i postavlja ga u okno (linija 21).

Program kreira drugi Text objekat sa više linija (linija 23) i postavlja ga u okno (linija 24).

Program kreira treći Text objekat (linja 26), određuje njegovu boju (linija 27), određuje tekst podvučen donjom crtom i tekst sa linijiom preko teksta (linje 28-29) i postavlja ga u okno (linija 30).

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.geometry.Insets;
import javafx.stage.Stage;
```

```
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;
import javafx.scene.text.FontPosture;
public class ShowText extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna koji treba da sadrži tekst
        Pane pane = new Pane();
        pane.setPadding(new Insets(5, 5, 5, 5));
        Text text1 = new Text(20, 20, "Programming is fun");
        text1.setFont(Font.font("Courier", FontWeight.BOLD,
                FontPosture.ITALIC, 15));
        pane.getChildren().add(text1);

        Text text2 = new Text(60, 60, "Programming is fun\nDisplay text");
        pane.getChildren().add(text2);

        Text text3 = new Text(10, 100, "Programming is fun\nDisplay text");
        text3.setFill(Color.RED);
        text3.setUnderline(true);
        text3.setStrikethrough(true);
        pane.getChildren().add(text3);

        // Kreiranje scene i njeno postavljene na pozornicu
        Scene scene = new Scene(pane);
        primaryStage.setTitle("ShowText"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Postavljenje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

VIDEO - UNOŠENJE TEKSTA

JavaFX Java GUI Tutorial - 10 - Extract and Validate Input (8,37)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

8.2 Klasa Line

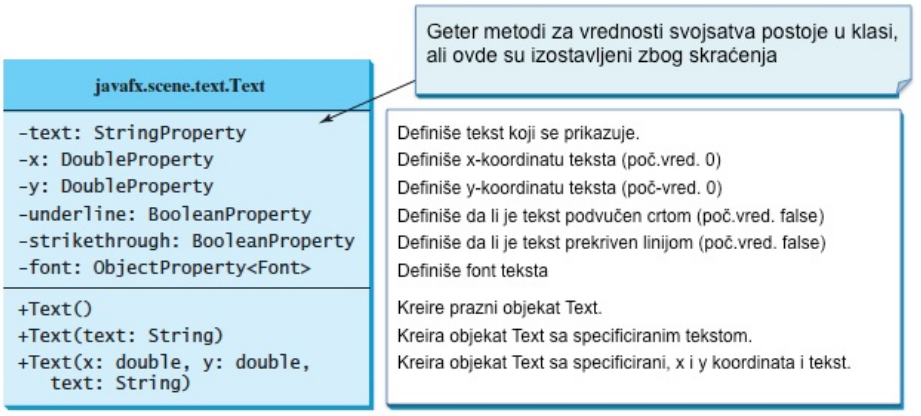
SVOJSTVA KLASE LINE

Klasa Line definiše liniju određenu početnom i krajnjom tačkom.

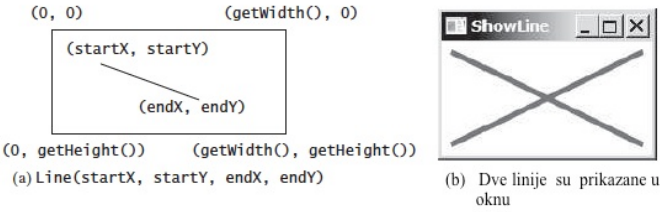
Slika 1 prikazuje UML dijagram klase ***Line*** koja definiše liniju.

Klasa Line kreira liniju koja povezuje dve tačke koje određene parametrima ***startX*** ,***startY*** ,***endX*** i***endY*** , kao što je prikazano na slici 2.a.

.



Slika 8.3.1 .1: UML dijagram klase Line koja prikazuje liniju između dve tačke



Slika 8.3.2 .2: Kreira se objekat Line radi prikaza linije [1]

PRIMER KORIŠĆENJA KLASE LINE

Program treba da prikaže dve ukrštene linije u oknu

Listing prikazuje program koji pokazuje kako se dobija prikaz dve ukrštene linije na slici 2.b.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Line;
```

```
public class ShowLine extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreira scenu i postavlja je na pozornicu
        Scene scene = new Scene(new LinePane(), 200, 200);
        primaryStage.setTitle("ShowLine"); // Unosi naziv pozornice
        primaryStage.setScene(scene); // Postavlja scenu na pozornicu
        primaryStage.show(); // Prikazuje pozornicu
    }

    class LinePane extends Pane {

        public LinePane() {
            Line line1 = new Line(10, 10, 10, 10);
            line1.endXProperty().bind(widthProperty().subtract(10));
            line1.endYProperty().bind(heightProperty().subtract(10));
            line1.setStrokeWidth(5);
            line1.setStroke(Color.GREEN);
            getChildren().add(line1);

            Line line2 = new Line(10, 10, 10, 10);
            line2.startXProperty().bind(widthProperty().subtract(10));
            line2.endYProperty().bind(heightProperty().subtract(10));
            line2.setStrokeWidth(5);
            line2.setStroke(Color.GREEN);
            getChildren().add(line2);
        }
    }

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

8.3 Klasa Rectangle

SVOJSTVA KLASE RECTANGLE

Klasa Rectangle definiše pravougaonik određen gornjim levim temenom, širinom i visinom pravougaonika.

Slika 1 prikazuje UML dijagram klase ***Rectangle Ie*** koja definiše pravougaonik.

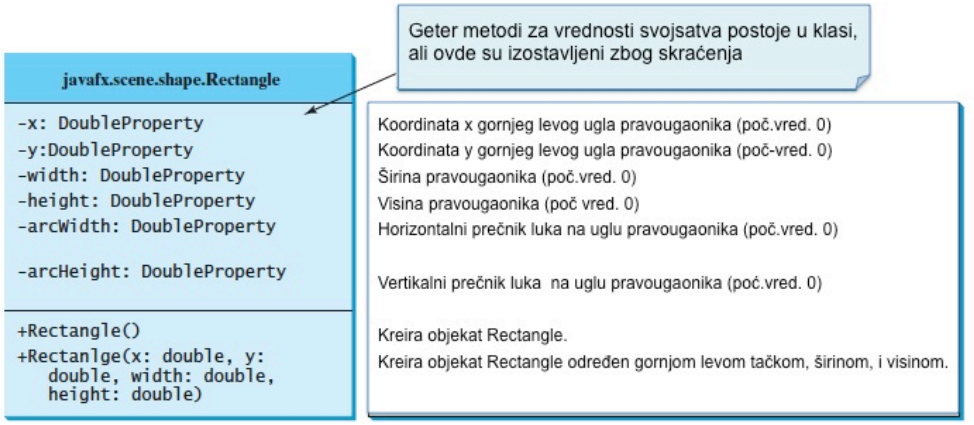
Klasa Rectangle kreira pravougaonik koji je definisan parametrima x i y (koordinatama gorenjeg levog temena), širinom, visinom, širinom i visinom luka u njegovim temenima (slika 2.a) , tj. parametrima: ***x , y , width , height , arcWidth , i arcHeight.***

Koordinate ***x*** i ***y*** određuju položaj gornjeg levog temena pravougaonika***, arcgWith*** i ***ArcHeight*** određuju horizontalni, odnosno vertikalni prečnik ugaonih lukova pravougaonika.

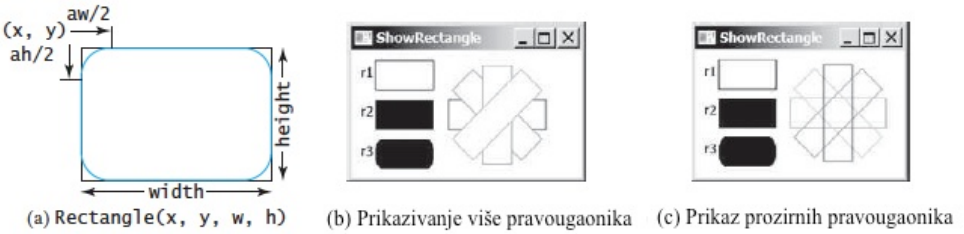
.

.

.



Slika 8.4.1 .1: UML dijagram klase Rectangle



(a) Rectangle(x, y, w, h) (b) Prikazivanje više pravougaonika (c) Prikaz prozirnih pravougaonika

Slika 8.4.2 .2: Objekat Rectangle prikazuje pravougaonik [1]

PRIMER KORIŠĆENJA KLASE RECTANGLE

Program prikazuje crtanje više pravougaonika, zarotiranih oko centra scene za određeni ugao.

Listing programa ShowRectangle demonstrira crtanje pravougaonika prikazanih na slici 2.b.

Program kreira vipe pravougaonika. Njihova površina ima unapred definisanu boju, a to je crna. Isto tako, početno podešena boja okvirne linije pravougaonika je crna.

Boja okvirne linije pravougaonika r1 je crna (linija 17).

Program kreira pravougaonik r3 (linija 26) i određuje širine i visine njegovih lukova u uglovima (linije 27-28). Zato, pravougaonik r3 ima zaobljenna temena.

Program više puta ponavlja kreiranje pravougaonika (linija 33), rotira ga (linija 34), slučajno postavlja boju njegove okvirne linije (linije 35-36), boju njegove unutrašnjosti koja je bela (linija 37) i dodaje pravougaonik u okno (linija 38),ređajuži ih jedan preko drugoga.

Ako se linija 37 zameni se linjom:

r.setFill(null);

Pravougaonik neće biti popunjen bojom. Zako se dobija prikaz prozirnih pravougaonika na slici 2.c.

.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.text.Text;
import javafx.scene.shape.Rectangle;
public class ShowRectangle extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna
        Pane pane = new Pane();

        // Kreiranje pravougaonika i njihovo dodavanje u okno
        Rectangle r1 = new Rectangle(25, 10, 60, 30);
        r1.setStroke(Color.BLACK);
        r1.setFill(Color.WHITE);
        pane.getChildren().add(new Text(10, 27, "r1"));
        pane.getChildren().add(r1);

        Rectangle r2 = new Rectangle(25, 50, 60, 30);
        pane.getChildren().add(new Text(10, 67, "r2"));
        pane.getChildren().add(r2);

        Rectangle r3 = new Rectangle(25, 90, 60, 30);
        r3.setArcWidth(15);
        r3.setArcHeight(25);
        pane.getChildren().add(new Text(10, 107, "r3"));
        pane.getChildren().add(r3);

        for (int i = 0; i < 4; i++) {
            Rectangle r = new Rectangle(100, 50, 100, 30);
            r.setRotate(i * 360 / 8);
            r.setStroke(Color.color(Math.random(), Math.random(),
                    Math.random()));
            r.setFill(Color.WHITE);
            pane.getChildren().add(r);
        }

        // Kreiranje scene i njeno postavljanje u pozornicu
        Scene scene = new Scene(pane, 250, 150);
        primaryStage.setTitle("ShowRectangle"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Stavljanje scene na pozornicu
```

```
        primaryStage.show(); // Prikaz pozornice
    }

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

8.4 Klase Circle i Ellipse

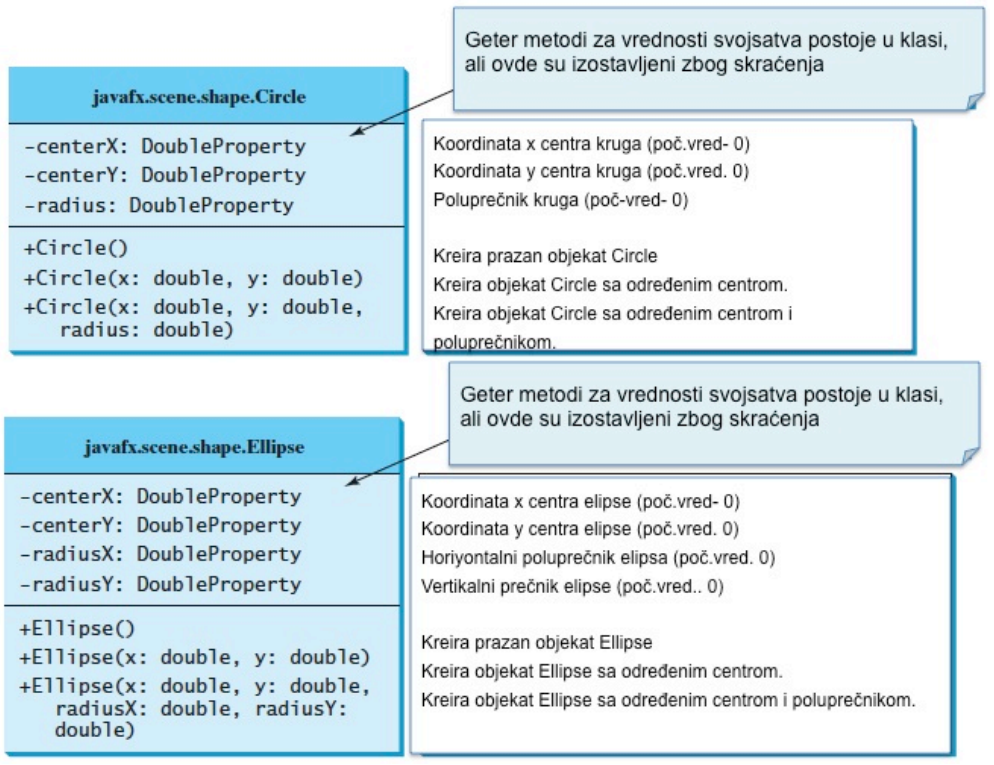
SVOJSTAVA KLASA CIRCLE I ELLIPSE

Klase Circle i Ellipse definišu krug, odnosno elipsu

Na slici 1 prikazani su UML dijagrami klasa ***Circle*** i ***Ellipse***, koje definišu krug, odn. elipsu.

Klasa Circle kreira krug koji je definisam svojm cenrom sa parametrima ***centerX , centerY*** , polupečnikom ***(radius***) .

Klasa Ellipse kreira elipsu koja je definisana svojm centro, tj. parametrima .***centerX , centerY*** , i horizontalnim i vertikalnim polupračnikom, tj. parametrima ***radiusX*** , and ***radiusY*** , kao što je prikazano na slici 2.a.



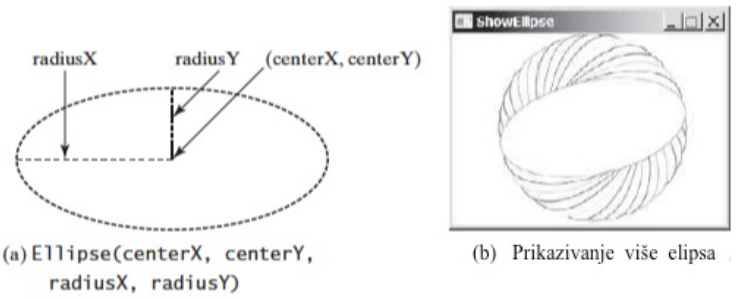
Slika 8.5.1 .1: UML dijagrami klasa Circle i Ellipse [1]

PRIMER KORIŠĆENJA KLASE ELLIPSE

Program treba da nacrta niz zarotiranih elipsi oko centra okna i scene, a svaka elipsa ima slučajno određena svojstva (boja granične linije i popune).

Potrebno je da se nacrta niy yarotiranih elipsa u oknu, kao što je to prikazano na slici 2.b. Listing programa ShowEllipse prikazuje progra, čijim izvrešenjem se dobija skup zarotiranih elipsi oko centra okna i scene (slika 2.b).

Program više puta ponavlja crtanje elipse (linija 16), slučajno određuje boju granične linije (linije 17-18), postavlja boju njihove popune (linija 19) i dodaje nacrtan pravougaonik u okno (linija 21).



Slika 8.5.2 .2: (a) Elipsa (b) Prikaz više elipsa [1]

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Ellipse;
public class ShowEllipse extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna
        Pane pane = new Pane();

        for (int i = 0; i < 16; i++) {
            // Kreiranje elipse i njeno dodavanje u okno
            Ellipse e1 = new Ellipse(150, 100, 100, 50);
            e1.setStroke(Color.color(Math.random(), Math.random(),
                    Math.random()));
            e1.setFill(Color.WHITE);
            e1.setRotate(i * 180 / 16);
            pane.getChildren().add(e1);
        }

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane, 300, 200);
```

```
        primaryStage.setTitle("ShowEllipse"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

8.5 Klasa Arc

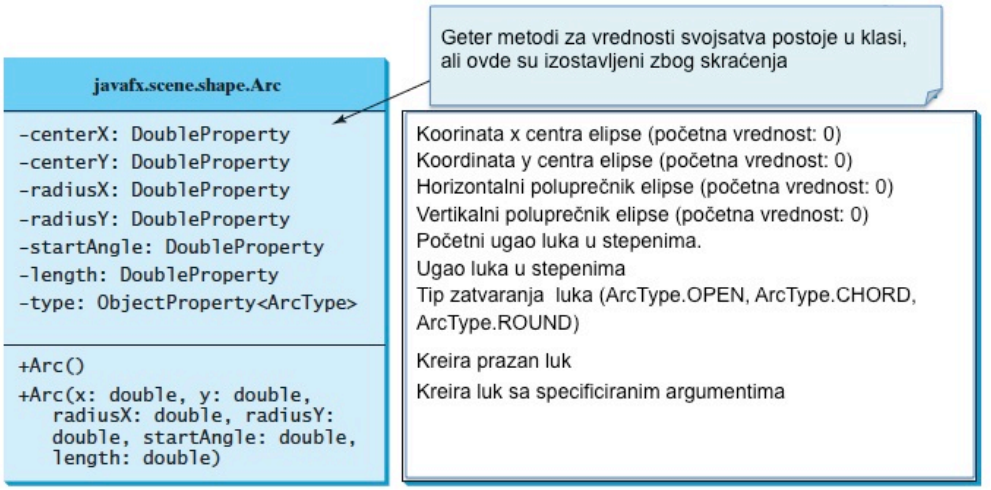
SVOJSTVA KLASE ARC

Klasa Arc definiše luk.

Klasa ***Arc*** definiše luk. UML dijagram klase***Arc*** je prikazan na slici 1.

Klasa Arc kreira luk koji je je deo elpse definisan parametrima: ***centerX*** ,***centerY*** ,***radiusX*** ,***radiusY*** ,***startAngle*** ,***length*** , i tip luka ***type (ArcType.OPEN*** ,***ArcType.CHORD*** ili ***ArcType.ROUND***).

Parametar ***startAngle*** je početni ugao, a ***length*** je ugao prostiranja luka (tj. Ugao obuhvaćen lukom). Uglovi se mere u stepenima i primenjuju uobičajenu matematičku knvenciju (tj. 0 stepeni je u pravcu istoka, a pozitivan ugao znači ugao u smeru suprotnom od kretanja kazaljke na satu), kao što je pokazano na slici 2.a.

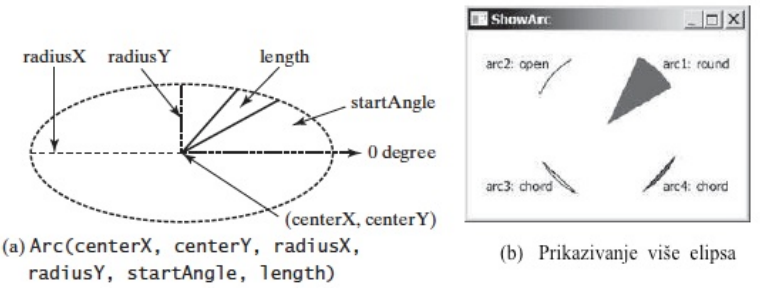


Slika 8.6.1 .1: UML dijagram klase Arc koja crta luk [1]

PRIMER CRTANJA LUKOVA

Program crta četiri luka i odsečka luka koji su prikazani na slici 2b.

Na slici 2a prikazani su prikazani parametri elipse, a na slici 2b primer korišćenja klase Arc. Prikazan listing programa **ShowArc** prikazuje kod koji ostvaruje rezultat prikazan na slici 2b.



Slika 8.6.2 .2: Objekat Arc prikazuje jedan luk [1]

Program kreira luk arc1 sa centrom (150,100) sa poluprečniko, radiusX 80 i radiusY 80. Početni ugao je 30, a prelazni ugao je 35 (linija 15). Tip luka arc1 je ArcTzpe.ROUND (linija 18). Arc1 je popunjen crvenom bojom, kao i njegova granična linija

Program kreira i luk arc3, sa centrom (150,100) poluprečnikom radiusX 80 i radiusZ 80. Početni ugao je 30 + 180 sa prelaznim uglom 35 (linija 29(, Tip luka arc3 je . ArcTzpe.CHORD (linija 31). Boja popune luka arc3 je bela i boja granične linije odesčka luka je crna.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Arc;
import javafx.scene.shape.ArcType;
import javafx.scene.text.Text;
public class ShowArc extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje okna
        Pane pane = new Pane();

        Arc arc1 = new Arc(150, 100, 80, 80, 30, 35); // Kreiranje luka
        arc1.setFill(Color.RED); // Unos boje popunjavanja segmenta luka
        arc1.setType(ArcType.ROUND); // Unos tipa lika
        pane.getChildren().add(new Text(210, 40, "arc1: round"));
        pane.getChildren().add(arc1); // Dodavanje luka u okno

        Arc arc2 = new Arc(150, 100, 80, 80, 30 + 90, 35);
        arc2.setFill(Color.WHITE);
        arc2.setType(ArcType.OPEN);
        arc2.setStroke(Color.BLACK);
        pane.getChildren().add(new Text(20, 40, "arc2: open"));
        pane.getChildren().add(arc2);

        Arc arc3 = new Arc(150, 100, 80, 80, 30 + 180, 35);
        arc3.setFill(Color.WHITE);
```

```
        arc3.setType(ArcType.CHORD);
        arc3.setStroke(Color.BLACK);
        pane.getChildren().add(new Text(20, 170, "arc3: chord"));
        pane.getChildren().add(arc3);

        Arc arc4 = new Arc(150, 100, 80, 80, 30 + 270, 35);
        arc4.setFill(Color.GREEN);
        arc4.setType(ArcType.CHORD);
        arc4.setStroke(Color.BLACK);
        pane.getChildren().add(new Text(210, 170, "arc4: chord"));
        pane.getChildren().add(arc4);

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane, 300, 200);
        primaryStage.setTitle("ShowArc"); // Unos naslova pozornice
        primaryStage.setScene(scene); // postavljanje scene pozornice
        primaryStage.show(); // Prikaz pozornice
    }

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

DEFINISANJE UGLOVA LUKA

Ugao je negativan ako se dobija okretanjem poluprečnika luka u smeru okretanja kazaljke na satu, počev od istočnog pravca.

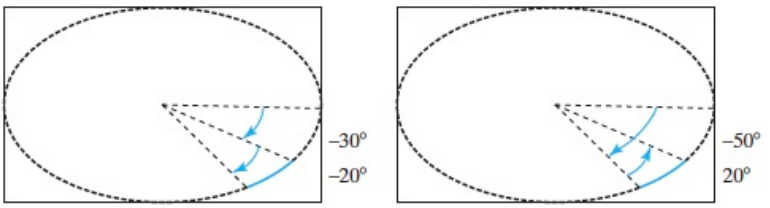
Uglovi mogu biti i negativni. To znači da se kreće u smeru kazaljke na satu, počev od istočnog pravca (slika 3). Negativni prelazni ugao označava ugao koji se formira kretanje u smeru obrtanja skazaljke na satu počev od početnog ugla. Sledeća dva iskaza prikazuju definišu isti ugao.

:

```
new Arc(x, y, radiusX, radiusY, -30, -20);
 new Arc(x, y, radiusX, radiusY, -50, 20);
```

Prvi iskaz upotrebljava početni ugao -30 i negativni prelazni ugao -20, kao što je prikazano na slici 3.a.,

Drugi iskaz upotrebljava negativni početni ugao -50 i pozitiv prelazni ugao 20, kao što je prikazano na slici 3.b.



(b) Negativni početni ugao -30° i egativni ugao prostiranja -20°

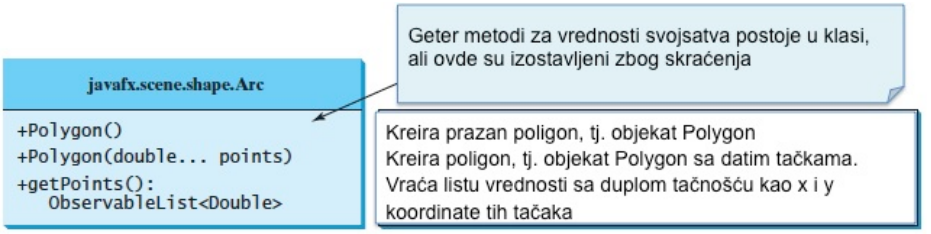
(b) Negativni početni ugao -50° i positivni ugao prostiranja 20°

Slika 8.6.3 .3: Ugao može biti i negativan [1]

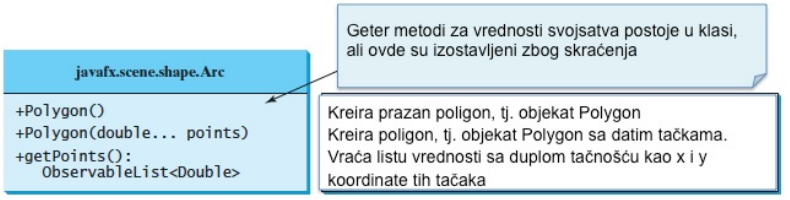
8.6 Klase Polygon i Polyline

SVOJSTVA KLASE POLYGON

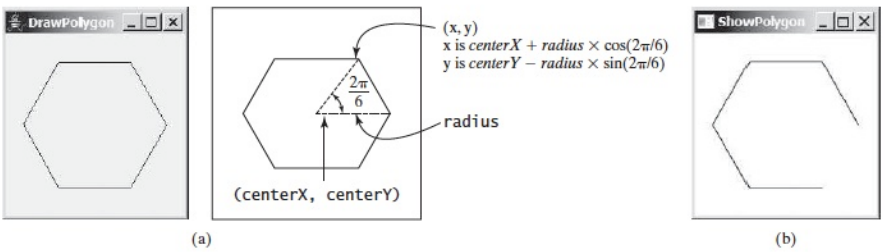
Klasa Polygon definiše poligon koji povezuje niz tačaka.



Slika 8.7.1 .1: Klasa Polygon [1]



Slika 8.7.2 .2: Polygon crta zatvorenu liniju,a Polyline otvorenu liniju kroz niz tačaka [1]



Slika 8.7.3 .3: Parametri klasa Polygon i Polyine [1]

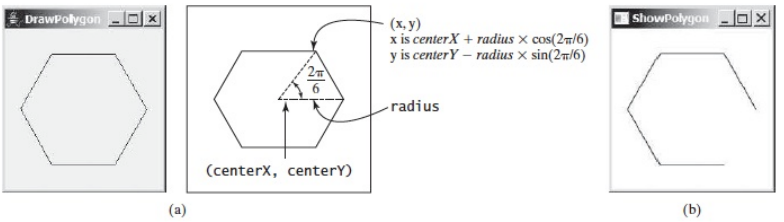
PRIMER PRIMENE KLASE POLYGON

Program crta heksagon, uz pomoć klase Polygon.

Program kreira Polygon objekat (linija 14) i dodaje ga u okvir (linija 15). Metod ***getPoints()*** vraća listu ***aObservableList<Double>*** (linija 18), koja sadrži add() metod za dodavanje elemenata u listu (linije 26-27). Parametar metoda add(vlue) mora bite duple tačnosti, tj. tipa ***Double.***

Petlja dodaje šest tačaka poligona (linije 25-28). Svaka tačka je predstavljena svojim x- i y-koordinate heksagona na slici Koordinate heksagona se računaju prema formulama datim na slici 3.a.

Ako se objekat Polygon zameni sa objektom Polyline, program onda prikazuje poliliniju, kao na slici 3.b.



Slika 8.7.4 .4: Parametri klasa Polygon i Polyine [1]

```
import javafx.application.Application;
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;
import javafx.scene.shape.Polygon;
public class ShowPolygon extends Application {

    @Override // Redefinisanje metoda start klase Application
    public void start(Stage primaryStage) {
        // Create a pane, a polygon, and place polygon to pane
        Pane pane = new Pane();
        Polygon polygon = new Polygon();
        pane.getChildren().add(polygon);
        polygon.setFill(Color.WHITE);
        polygon.setStroke(Color.BLACK);
        ObservableList<Double> list = polygon.getPoints();

        final double WIDTH = 200, HEIGHT = 200;
        double centerX = WIDTH / 2, centerY = HEIGHT / 2;
        double radius = Math.min(WIDTH, HEIGHT) * 0.4;

        // unos tačaka poligona
        for (int i = 0; i < 6; i++) {
            list.add(centerX + radius * Math.cos(2 * i * Math.PI / 6));
```

```
            list.add(centerY - radius * Math.sin(2 * i * Math.PI / 6));
        }

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane, WIDTH, HEIGHT);
        primaryStage.setTitle("ShowPolygon"); // Unos naslova pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

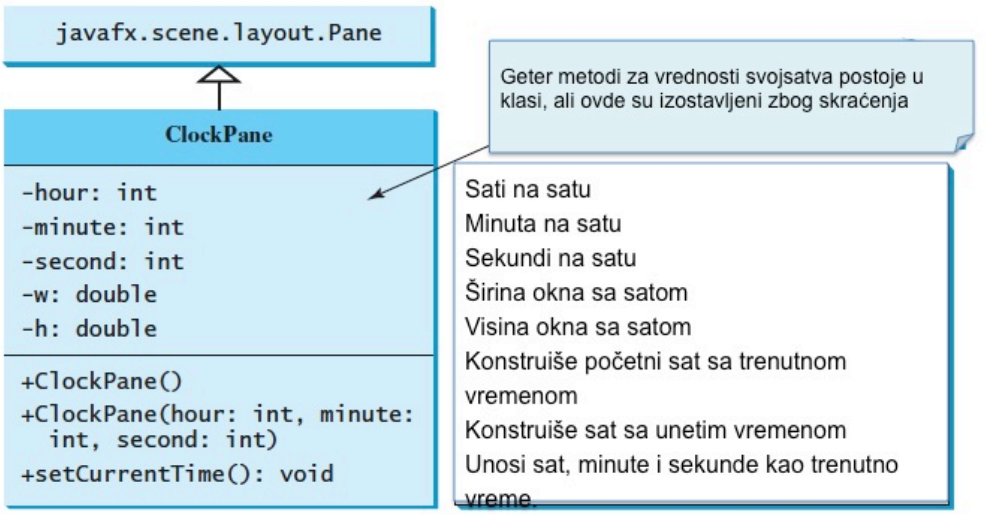
Poglavlje 9

Studija slučaja: Klasa ClockPane

KLASA CLOCKPANE

Klasa ClockPane kreira, crta i boji sat.

UML dijagram klase ClockPane je prikazan na slici 1.



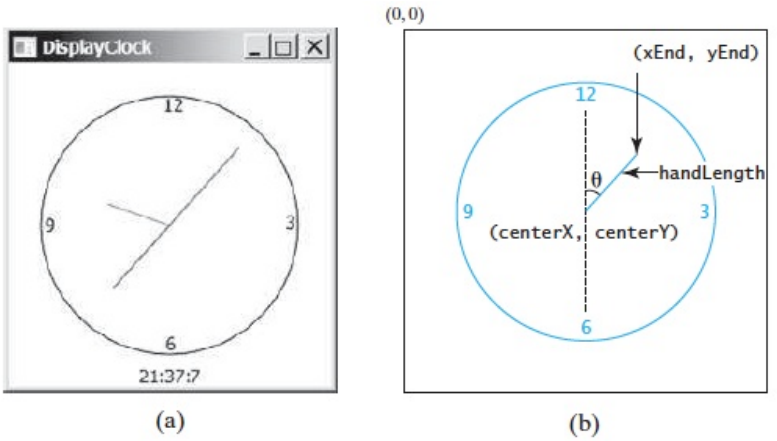
Slika 9.1 .1: Klasa ClockPane prikazuje analogni sat [1]

LISTING PROGRAMA DISPLAYCLOCK

Program prikazuje analogni sat sa oznakama sata, minuta i sekundi.

Pod pretpostavkom da je klasa **ClockPane** raspoloživa (biće opisana kasnije), ovde se prikazuje listing programa za prikaz analognog sata i za upotrebu natpisa za prikaz sata, minuta i sekundi kao što je prikazano na slici 2.

.



Slika 9.2 .2: Izgled sata koji prikazuje program ClockPane [1]

Koordinate vrha skazaljke se računa izrazima:

endX = centerX + handLength × sin(θ)

endY = centerY - handLength × cos(θ)

```
import javafx.application.Application;
import javafx.geometry.Pos;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.BorderPane;
public class DisplayClock extends Application {

    @Override // Predefinisanje metoda start u klasi Application
    public void start(Stage primaryStage) {
        // Kreiranje sata i naslova
        ClockPane clock = new ClockPane();
        String timeString = clock.getHour() + ":" + clock.getMinute()
                + ":" + clock.getSecond();
        Label lblCurrentTime = new Label(timeString);

        // Postavljanje sata i nalepnice nu okno  BorderPane
        BorderPane pane = new BorderPane();
        pane.setCenter(clock);
        pane.setBottom(lblCurrentTime);
        BorderPane.setAlignment(lblCurrentTime, Pos.TOP_CENTER);

        // Kreiranje scene i postavljanje je na pozornicu
        Scene scene = new Scene(pane, 250, 250);
        primaryStage.setTitle("DisplayClock"); // Unos imena pozornice
        primaryStage.setScene(scene); // STavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

```
    }
}
```

LISTING KLASE CLOCKPANE

Klasa ClockPane crta analogni sat u oknu

Jedan minut ima 60 sekundi, te je ugao okretanje skazaljke sa sekundama:

second x (2π/60)

Položaj skazaljke sa minutima zavisi od minuta i sekundi. Vrednost minuta se određuje je: minute + second/60.

Ugao skazaljke sa minutima se određuje izrazom:

(minute + second/60) x (2π/60)

Kako u jednom krugu postoji 12 sati, ugao skazaljke sa satom se računa pomoću izraza:

(hour + minute/60 + second/(60 x 60))x (2π/12)

Ako se zanemari uticaj na sekundi, mogu se koristiti sledeće pojednostavljeni izrazi:

secondX = centerX + secondHandLength × sin(second × (2π/60))
secondY = centerY - secondHandLength × cos(second × (2π/60))
minuteX = centerX + minuteHandLength × sin(minute × (2π/60))
minuteY = centerY - minuteHandLength × cos(minute × (2π/60))
hourX = centerX + hourHandLength × sin((hour + minute/60) × (2π/12))
hourY = centerY - hourHandLength × cos((hour + minute/60) × (2π/12))

Slika 9.3 .3: Izrazi za obračun koordinata vrhova sklazaljki [1]

```
import java.util.Calendar;
import java.util.GregorianCalendar;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Line;
import javafx.scene.text.Text;
public class ClockPane extends Pane {

    private int hour;
    private int minute;
    private int second;

    // Širina i visina okna sata
    private double w = 250, h = 250;

    /**
     * Konstruisanje početnog sata sata sa trenutnim vremenom
     */
    public ClockPane() {
```

```
        setCurrentTime();
    }

    /**
     * Konstruisanje sata sa specificiranim satim, minutima i sekundama
     */
    public ClockPane(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
        paintClock();
    }

    /**
     * Vraća hour
     */
    public int getHour() {
        return hour;
    }

    /**
     * Unisi novu vrednost za hour
     */
    public void setHour(int hour) {
        this.hour = hour;
        paintClock();
    }

    /**
     * vraća minute
     */
    public int getMinute() {
        return minute;
    }

    /**
     * Unosi novu vrednost za minute
     */
    public void setMinute(int minute) {
        this.minute = minute;
        paintClock();
    }

    /**
     * Vraća second
     */
    public int getSecond() {
        return second;
    }

    /**
     * Unosi novu vrednost za second
     */
```

```
    public void setSecond(int second) {
        this.second = second;
        paintClock();
    }

    /**
     * Vraća širinu okna za sat
     */
    public double getW() {
        return w;
    }

    /**
     * Unosi širinu okna za sat
     */
    public void setW(double w) {
        this.w = w;
        paintClock();
    }

    /**
     * Vraća visinu okna za sat
     */
    public double getH() {
        return h;
    }

    /**
     * Podešava visinu okna sata
     */
    public void setH(double h) {
        this.h = h;
        paintClock();
    }

    /* Unosi sadašnje vreme u sat  */
    public void setCurrentTime() {
        // Konstruiše kalendar ya sadašnji datum i vreme
        Calendar calendar = new GregorianCalendar();

        // Unos trenutnog vremena sata, minuta i sekundi
        this.hour = calendar.get(Calendar.HOUR_OF_DAY);
        this.minute = calendar.get(Calendar.MINUTE);
        this.second = calendar.get(Calendar.SECOND);

        paintClock(); // nacrtaj sat
    }

    /**
     * Crtanje sata
     */
    protected void paintClock() {
        // Inicijalizacija parametra sata
```

```
        double clockRadius = Math.min(w, h) * 0.8 * 0.5;
        double centerX = w / 2;
        double centerY = h / 2;
        // Crtanje kruga
        Circle circle = new Circle(centerX, centerY, clockRadius);
        circle.setFill(Color.WHITE);
        circle.setStroke(Color.BLACK);
        Text t1 = new Text(centerX - 5, centerY - clockRadius + 12, "12");
        Text t2 = new Text(centerX - clockRadius + 3, centerY + 5, "9");
        Text t3 = new Text(centerX + clockRadius - 10, centerY + 3, "3");
        Text t4 = new Text(centerX - 3, centerY + clockRadius - 3, "6");

        // Crtanje skazaljke za prikaz sekundi
        double sLength = clockRadius * 0.8;
        double secondX = centerX + sLength
                * Math.sin(second * (2 * Math.PI / 60));
        double secondY = centerY - sLength
                * Math.cos(second * (2 * Math.PI / 60));
        Line sLine = new Line(centerX, centerY, secondX, secondY);
        sLine.setStroke(Color.RED);

        // Crtanje velike skazalje za prikaz minuta
        double mLength = clockRadius * 0.65;
        double xMinute = centerX + mLength
                * Math.sin(minute * (2 * Math.PI / 60));
        double minuteY = centerY - mLength
                * Math.cos(minute * (2 * Math.PI / 60));
        Line mLine = new Line(centerX, centerY, xMinute, minuteY);
        mLine.setStroke(Color.BLUE);

        // Crtanje sklayalje ya prikaz sati
        double hLength = clockRadius * 0.5;
        double hourX = centerX + hLength
                * Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12));
        double hourY = centerY - hLength
                * Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12));
        Line hLine = new Line(centerX, centerY, hourX, hourY);
        hLine.setStroke(Color.GREEN);

        getChildren().clear();
        getChildren().addAll(circle, t1, t2, t3, t4, sLine, mLine, hLine);
    }
}
```

OBJAŠNJENJE LISTINGA PROGRAMA CLOCKPANE

Metod paintClock se poziva uvek kada se promeni bilo koje svojstvo sata (hour, minute, second, w i h)

Program prikazuje analogni sat sa trenutnom vremenom korišćenjem početnog konstruktora bez argumenata (linije 18-20) . Prikazuje se sat sa specificiranim satom, minutom i sekundom upotrebom konstruktora (linije 23-28). Konstruktor sa trenutnim vremenom je dat na linijama 86-96).

Klasa definiše svojstva: sat, minut i sekunda koja se memoriše kao trenutno vreme (linije 10-12). Širina i visina okvira je definisano u linijama 15). Početne vrednosti za ***w*** i ***H*** su 250. To se može promeniti metodima ***setW*** i ***setH*** (linije 69, 80). Te veličine se koriste za crtanja sata u oknu sa metodom ***paintClock().***

Metod ***paintClock()*** boji i crta sat (linije 99-143). Poluprečnik sata je proporcionalan širini i visini okna (linija 101). Krug sata je postavljen u centar okna (linija 106). Tekst koji pokazuje sate: 12,, 3, 6, 9 je kreiran u linijama 109-112.

Skazaljka-sekundara, skazaljka za minute i skazaljka za satove su linije kreirane u linijama 114-139. Metod ***paintClock()*** postavlja sve ove oblike u okbu upotrebom addAll metod u listu (linija 142). Metod ***paintCloick*** se poziva uvek kada se promeni bilo koje svojstvo sata (hour, minute, second, w i h) u linijama 27, 38, 49, 60, 71, i 82, i 95. Pre ubacivanja novih svojstava u sadržaj okna, briše se prethodni (linija 141)

Poglavlje 10

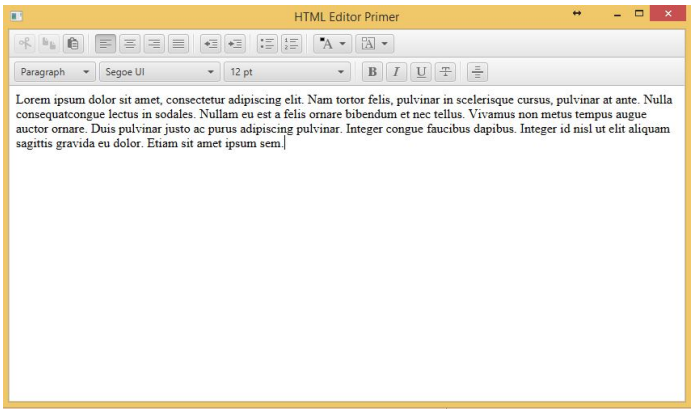
Vežba – Pokazni primeri

PRIMER 1

Cilj ovog primera je prikaz korišćenja HTML Editor komponente u Java FX-u

Napraviti formu kao na sledećoj slici koristeći HTMLEditor komponentu u JavaFX-u.

Potrebno vreme: 10 minuta



Slika 10.1 .1: HTML editor

JavaFX nam obezbeđuje gotovu GUI komponentu koja se naziva **HTMLEditor** i koja nam omogućava osnovne funkcije manipulacije sa tekstom (formatiranje teksta, copy, cut, paste, itd).

Potrebno je samo da kreiramo instancu klase HTMLEditor, nakon čega joj možemo definisati dimenzije i inicijalni tekst. Kreiranu instancu moramo dodati na scenu. U prikazanom primeru root komponenta aplikacije je VBox kontejner u koji ćemo smestiti naš editor, a zatim ćemo VBox objekat proslediti sceni.

Klasa Main.java

```
import javafx.application.Application;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.scene.web.HTMLEditor;
import javafx.stage.Stage;

public class Main extends Application {
```

```
    private final String TEKST_ZA_PRIKAZ = "Lorem ipsum dolor sit "
            + "amet, consectetur adipiscing elit. Nam tortor felis, pulvinar "
            + "in scelerisque cursus, pulvinar at ante. Nulla consequat"
            + "congue lectus in sodales. Nullam eu est a felis ornare "
            + "bibendum et nec tellus. Vivamus non metus tempus augue auctor "
            + "ornare. Duis pulvinar justo ac purus adipiscing pulvinar. "
            + "Integer congue faucibus dapibus. Integer id nisl ut elit "
            + "aliquam sagittis gravida eu dolor. Etiam sit amet ipsum "
            + "sem.";

    @Override
    public void start(Stage stage) {
        stage.setTitle("HTML Editor Primer");
        stage.setWidth(500);
        stage.setHeight(500);
        Scene scene = new Scene(new Group());
        VBox root = new VBox();
        final HTMLEditor htmlEditor = new HTMLEditor();
        htmlEditor.setPrefHeight(600);
        htmlEditor.setHtmlText(TEKST_ZA_PRIKAZ);
        root.getChildren().addAll(htmlEditor);
        scene.setRoot(root);
        stage.setScene(scene);
        stage.show();
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String[] args) {
        launch(args);
    }

}
```

PRIMER 2

Cilj ovog primera je prikaz rada Media i MediaPlayer JavaFX komponenti

Pustiti MP3 fajl preko JaveFX.

Potrebno vreme: 5 minuta

Klasa Main:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.scene.media.Media;
```

```
import javafx.scene.media.MediaPlayer;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage primaryStage) {
        final Media media = new Media("file:///C:/a.mp3");
        final MediaPlayer mediaPlayer = new MediaPlayer(media);
        mediaPlayer.play();
        primaryStage.setTitle("Pustanje MP3 ");
        primaryStage.setWidth(200);
        primaryStage.setHeight(200);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

JavaFX nam takođe obezbeđuje GUI komponentu koja ima mogućnost reprezentacije audio datoteka.
Potrebno je kreirati objekat tipa **MediaPlayer** i proslediti mu putanju do audio datoteke koju želimo da pustimo. Pre toga moramo da podatke iz fajla smestimo u **Media** objekat. U primeru je korišćen fajl C:/a.mp3. Potrebno je da se promeni lokacija datoteke na lokaciju neke mp3 datoteke koja se nalazi na računaru na kom se program pokreće.

Ubaciti dugme koje ce zaustaviti pesmu.

PRIMER 3

Cilj ovog primera je prikaz korišćenja Media, MediaPlayer i MediaView JavaFX komponenti

Pustiti video fajl preko JaveFX.

Potrebno vreme: 5 minuta

Klasa Main:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.scene.Scene;
import javafx.scene.layout.VBox;
import javafx.scene.media.Media;
```

```
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.stage.Stage;

public class Main extends Application {

    @Override
    public void start(Stage stage) {
        VBox box = new VBox();
        Scene scene = new Scene(box, 1000, 500);
        stage.setScene(scene);
        Media media = new Media("http://techslides.com/demos/sample-videos/
small.mp4");
        MediaPlayer mediaPlayer = new MediaPlayer(media);
        mediaPlayer.setAutoPlay(true);
        MediaView mediaView = new MediaView(mediaPlayer);
        ((VBox) scene.getRoot()).getChildren().add(mediaView);
        stage.show();
    }

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

}
```

MediaPlayer je JavaFX komponenta koja omogućava reprezentaciju video snimaka. Potrebno je da prvo kreiramo objekat tipa **Media** koji prima putanju do video resorsa koji želimo da prikažemo, a zatim taj objekat da prosledimo objektu klase MediaPlayer. Kako bi se video vizuelno prikazao potrebno je da objekat MediaPlayer prosledimo komponenti **MediaView** koja će nam prikazati reprezentaciju videa u prozoru.

Postavljanjem **autoPlay** atributa objekta klase MediaPlayer na true, obezbeđujemo da se video pušta automatski. Moguće je dodati dugme koje preko kog ćemo aktivirati puštanje video snimka.

Moguće je prikazati bilo koji video koji se nalazi lokalno na računaru ili na nekoj internet lokaciji. Važno je da se navede putanja do resorsa, što znači da youtube linkovi neće moći da se reprezentuju.

Zašto ne može da se navede link sa youtube-a?

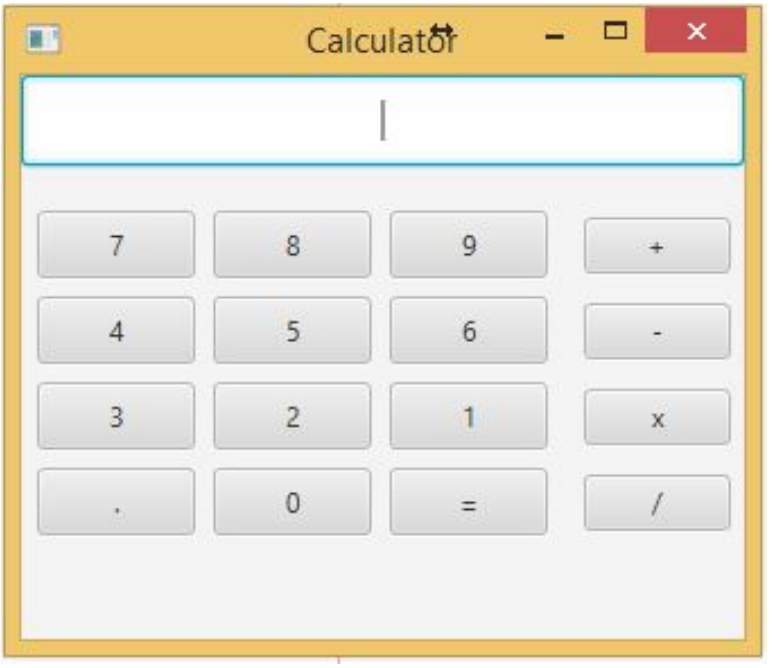
Pokrenuti snimak na dugme.

PRIMER 4

Pravljenje digitrona koristeći odgovarajuće kontejnere iz Jave FX

Napraviti interfejs za digitron koristeći Javu FX po sledećoj slici

Potrebno vreme: 10 minuta



Slika 10.2 .2: izgled kalkulatora

PRIMER 4 - REŠENJE

Prikaz programskog koda za primer 4

Klasa CalcButton:

```
import javafx.scene.control.Button;

public class CalcButton extends Button {

    private String sign;

    public CalcButton() {
        super();
    }

    public CalcButton(String sign) {
        super();
        setText(sign);
        setMinSize(70, 30);
        setMaxSize(70, 30);
    }

    public String getSign() {
        return sign;
    }

    public void setSign(String sign) {
        this.sign = sign;
```

```
    }

}
```

Klasa CalculatorStage:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.TextField;
import javafx.scene.layout.GridPane;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class CalculatorStage extends Application {

    @Override
    public void start(Stage primaryStage) {
        // postavljamo VBox layout i postavljamo verikalno rastojanje izmedju
cvorova unutar njega
        VBox root = new VBox();
        root.setSpacing(20);
        String[] signArr = new String[]{"7", "8", "9",
            "4", "5", "6",
            "3", "2", "1",
            ".", "0", "="};
        String[] opArr = new String[]{
            "+", "-", "x", "/"
        };
        // kreiramo textField i postavljamo mu visinu i poravnanje
        TextField display = new TextField();
        display.setMinHeight(40);
        display.setMaxHeight(40);
        display.setAlignment(Pos.CENTER);
        // kreiramo gridPane container
        GridPane grid = new GridPane();
        grid.setAlignment(Pos.CENTER);
        grid.setHgap(8);
        grid.setVgap(8);
        int signInd = 0;

        // dodavanje dugmica kroz for petlju
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 3; j++) {
                CalcButton bTmp = new CalcButton(signArr[signInd]);
                grid.add(bTmp, j, i);
                signInd++;
            }
        }
        for (int i = 0; i < opArr.length; i++) {
            Button bTmp = new Button(opArr[i]);
```

```
            bTmp.setMinSize(65, 25);
            bTmp.setMaxSize(65, 25);
            grid.add(bTmp, 4, i);
        }

        // stavljanje textField-a u root container
        root.getChildren().add(display);
        // stavljanje grid container unutar root-a
        root.getChildren().add(grid);
        // podesavamo scenu tako sto joj prosledjujemo koreni container i dimenzije
        Scene scene = new Scene(root, 320, 250);

        // podasavanje glavnog stage-a
        primaryStage.setTitle("Calculator");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

PRIMER 4 - OBJAŠNJENJE

Cilj sekcije je da se pojasni kod primera 4

U ovom zadatku smo kreirali svoju klasu koja će **CalcButton** koja predstavlja jedno dugme na kalkulatoru. Jedini atribut koji ima ova klasa je znak.

Izdvajanje dugmeta u posebnu klasu može da bude korisno ukoliko želimo da stilizujemo dugme kalkulatora kako bi smo sav taj kod koji se odnosi na stilizovanje definisali samo jednom u okviru klase. Takođe, sve zajedničke funkcionalnosti dugmića na kalkulatoru možemo definisati u ovoj klasi.

Za prikaz dugmića kao na slici je najadekvatniji **GridPane** kontejner s obzirom da su dugmići razvrstani po vrstama i kolonama. Dakle, definisali smo u nizu **singArr** sve brojeve koje treba prikazati, a u nizu **opArr** sve operacije koje se nalaze na kalkulatoru. Zatim ćemo prvo kroz ugnježdenu for petlju dodati prve tri kolone koje se odnose na brojeve, a zatim proći kroz niz sa operacijama i dodati poslednju kolonu dugmića. Svako dugme je objekat klase CalcButton koju smo malopre pomenuli.

S obzirom da kalkulator sadrži i display, potrebno je definisati određenu root komponentu u koju ćemo smestiti GridPane sa dugmićima i display koji je predstavljen kao **TextField** objekat. Root komponenta će biti instanca **VBox** kontejnera i u nju ćemo dodati prvo display, zatim GridPane i nakon toga ćemo je postaviti u scenu.

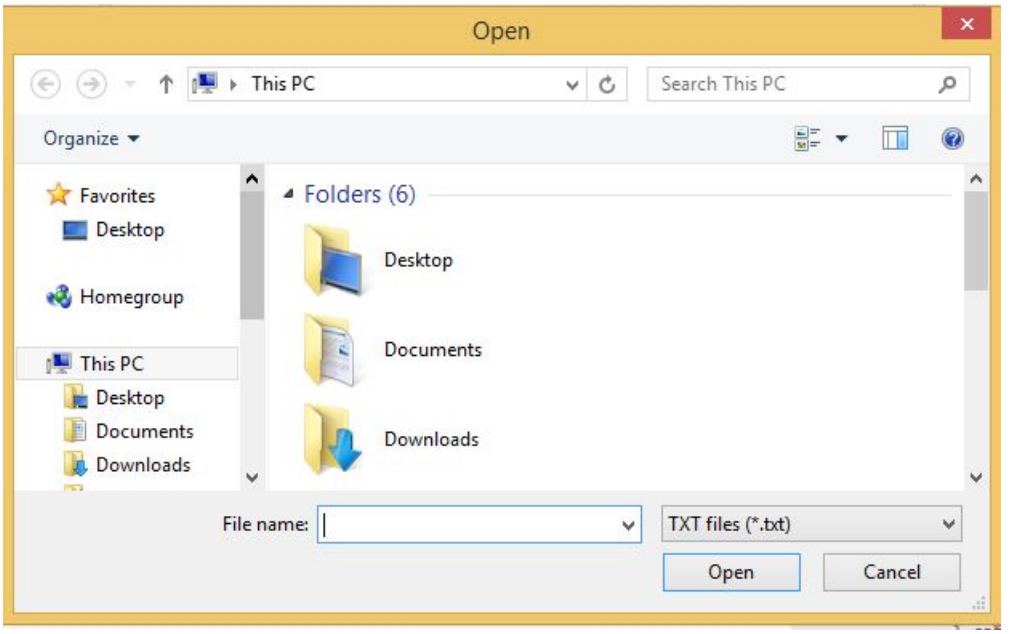
Razmisliti kako biste dodali funkcionalnost ovom kalkulatoru. Koji problemi mogu da nastanu, kako biste ih rešili?

PRIMER 5

Korišćenje FileChooser komponente u JavaFX-u.

Napraviti aplikaciju koja otvara txt fajl po izboru korisnika, a potom menja svako slovo **a** u slovo **b** unutar fajla.

Potrebno vreme: 10 minuta



Slika 10.3 .3: Prikaz grafičkog interfejsa za izbor fajlova

PRIMER 5 - REŠENJE

Prikaz programskog koda primera 5

Klasa Main:

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.scene.Group;
```

```
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.layout.VBox;
import javafx.scene.layout.VBoxBuilder;
import javafx.stage.FileChooser;
import javafx.stage.Stage;

public class Main extends Application {

    private BufferedWriter bw;
    private BufferedReader br;
    private File file;

    @Override
    public void start(Stage primaryStage) {
        Group root = new Group();
        Button buttonLoad = new Button("Ucitaj fajl");
        buttonLoad.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent arg0) {
                FileChooser fileChooser = new FileChooser();
                FileChooser.ExtensionFilter extFilter = new
FileChooser.ExtensionFilter("TXT files (*.txt)", "*.txt");
                fileChooser.getExtensionFilters().add(extFilter);
                file = fileChooser.showOpenDialog(primaryStage);
                try {
                    br = new BufferedReader(new FileReader(file));
                    //
                    String forReplace = readFromFile();
                    bw = new BufferedWriter(new FileWriter(file));
                    forReplace = forReplace.replace("a", "b");
                    writeInFile(forReplace);
                } catch (Exception e) {
                }
            }
        });
        VBox vBox = VBoxBuilder.create()
                .children(buttonLoad)
                .build();
        root.getChildren().add(vBox);
        primaryStage.setScene(new Scene(root, 500, 400));
        primaryStage.show();
    }

    public void writeInFile(String text) throws IOException {
        bw.write(text);
        bw.flush();
        bw.close();
    }

    public String readFromFile() throws IOException {
        String line = null;
        String result = "";
```

```
        while ((line = br.readLine()) != null) {
            result += line + "\r\n";
        }
        br = new BufferedReader(new FileReader(file));
        return result;
    }

    /**
    * @param args the command line arguments
    */
    public static void main(String[] args) {
        launch(args);
    }

}
```

PRIMER 5 - OBJAŠNJENJE

Cilj sekcije je da se pojasni kod primera 5

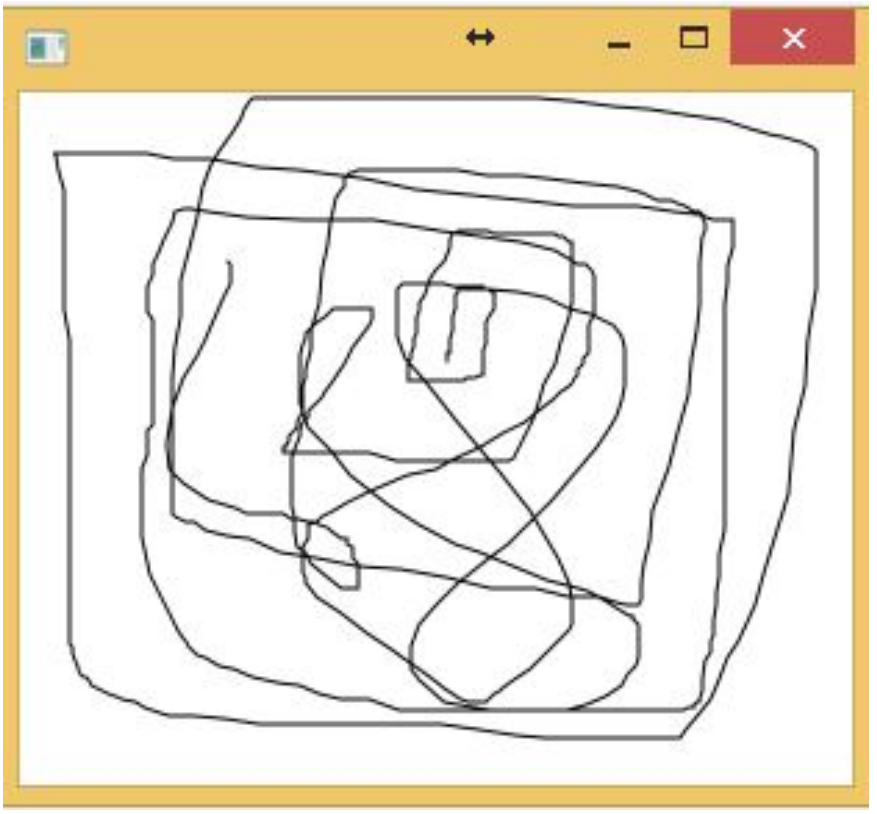
Dodati .csv ekstenziju (Comma-separated values)u filter FileChooser-a.

PRIMER 6

Cilj ovog primer je da prikaže rad sa događajima miša i crtanje pomoću Path-a u Javi FX

Napraviti aplikaciju koja omogućava korisniku da crta po ekranu tako što drži levi klik i prevlači miša po ekranu. Kada korisnik hoće da poništi ono što je nacrtao treba da klikne jednom desni klik.

Potrebno vreme: 10 minuta



Slika 10.4 .4: - Crtanje pomoću miša

PRIMER 6 - REŠENJE

Prikaz programskog koda primera 6

Klasa MainCrtanje:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.event.EventHandler;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.input.MouseButton;
import javafx.scene.input.MouseEvent;
import javafx.scene.paint.Color;
import javafx.scene.shape.LineTo;
import javafx.scene.shape.MoveTo;
import javafx.scene.shape.Path;
import javafx.stage.Stage;

public class MainCrtanje extends Application {

    Path path;

    @Override
    public void start(Stage primaryStage) {
```

```
        Group root = new Group();
        Scene scene = new Scene(root, 300, 250);

        path = new Path();
        path.setStrokeWidth(1);
        path.setStroke(Color.BLACK);

        scene.setOnMouseClicked(mouseHandler);
        scene.setOnMouseDragged(mouseHandler);
        scene.setOnMouseEntered(mouseHandler);
        scene.setOnMouseExited(mouseHandler);
        scene.setOnMouseMoved(mouseHandler);
        scene.setOnMousePressed(mouseHandler);
        scene.setOnMouseReleased(mouseHandler);

        root.getChildren().add(path);
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    EventHandler<MouseEvent> mouseHandler = new EventHandler<MouseEvent>() {

        @Override
        public void handle(MouseEvent mouseEvent) {
            if (mouseEvent.getEventType() == MouseEvent.MOUSE_PRESSED) {
                if (mouseEvent.getButton() == MouseButton.SECONDARY) {
                    path.getElements().clear();
                }
                path.getElements()
                        .add(new MoveTo(mouseEvent.getX(), mouseEvent.getY()));
            } else if (mouseEvent.getEventType() == MouseEvent.MOUSE_DRAGGED) {
                path.getElements()
                        .add(new LineTo(mouseEvent.getX(), mouseEvent.getY()));
            }

        }

    };

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

PRIMER 6 - OBJAŠNJENJE

Cilj sekcije je da se pojasni kod primera 6

Za rešenje ovog zadatka koristimo **Path** klasu koja nam omogućava da spajamo linije u jednu putanju. Zatim zadajemo debljinu linije, kao i boju.

Kako bi korisnik mogao mišem da crta po sceni, moramo da uhvatimo sve događaje koje miš može da izazove. Takve događaje zatim obrađujemo pomoću objekta **EventHandler<MouseEvent>** i njegovog metoda **handle**. MouseEvent klasa nam omogućava da proverimo koji se tip događaja desio sa **getEventType**, kao i koje dugme na mišu je izazvalo događaj sa **getButton**. Pored toga možemo da dobijemo koordinate klika miša sa **getX** i **getY**. Kada imamo koordinate, trebamo da dodamo odgovarajuću liniju u putanju. Liniju nadovezujemo na putanju sa **LineTo** objektom kojem prosleđujemo koordinate novog klika. Zatim taj deo putanje dodajemo u elemente putanje. To radimo kada korisnik vuče miša po sceni (MouseEvent.MOUSE_DRAGGED) . U slučaju da se desio klik, moramo da pomerimo početak putanje na druge koordinate pomoću **MoveTo**. U slučaju da se desio desni klik (MouseButton.SECONDARY), brišemo sve sa metodom **clear**.

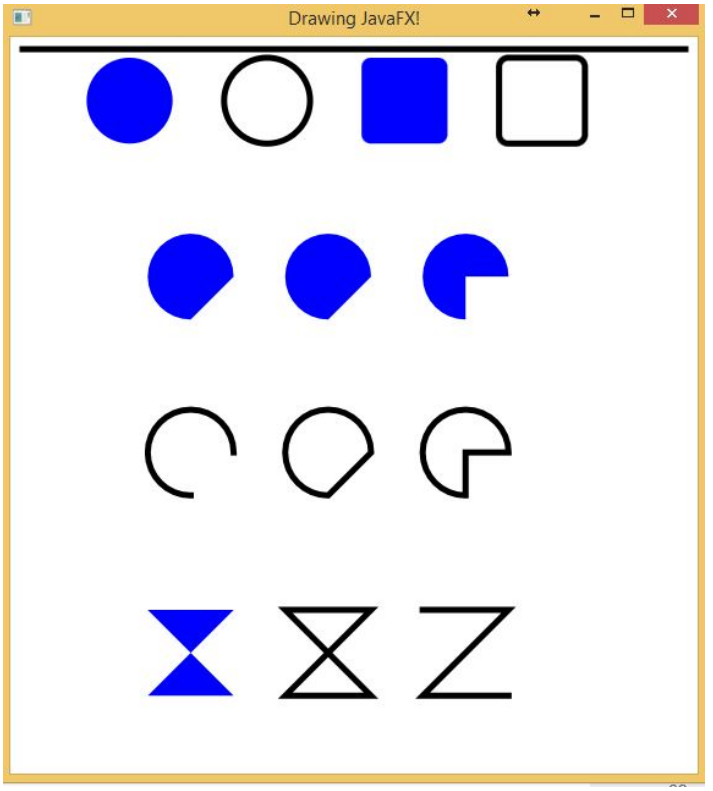
Izmeniti kod tako da se na desni klik umesto brisanja samo promeni boja linije.

PRIMER 7

Cilj primera je da se demonstrira iscrtavanje podobjekata klase Shape pomoću GraphicsContext-a u Javi FX

Napraviti aplikaciju koja crta sve što je prikazano na sledećoj slici. Koristiti GraphicsContext za crtanje preko JaveFX

Potrebno vreme: 10 minuta



Slika 10.5 .5: Crtanje sličica

PRIMER 7 - REŠENJE

Cilj ovog primera je da prikaže korišćenje crtanja pomoću GraphicsContext-a u Javi FX

Klasa MyCanvas:

```
import javafx.scene.canvas.Canvas;
import javafx.scene.canvas.GraphicsContext;
import javafx.scene.paint.Color;
import javafx.scene.shape.ArcType;

/**
 * Klasa MyCanvas nasladjuje klasu Canvas i sluzi za demonstraciju iscrtavanja
objekata
 * koji nasledjuju klasu Shape
 * @author Rados
 */
public class MyCanvas extends Canvas {
    // atribut graphicsContext na osnovu kog znamo da radimo sa 2D grafikom
    GraphicsContext gc2d = this.getGraphicsContext2D();

    public MyCanvas() {
        super();
        setWidth(600);
        setHeight(600);
        draw();
    }

    // metoda koja iscrtava sav sadrzaj unutar canvas-a
    public void draw() {
        gc2d.clearRect(0, 0, getWidth(), getHeight());
        gc2d.setFill(Color.BLUE);
        gc2d.setStroke(Color.BLACK);
        gc2d.setLineWidth(5);
        gc2d.strokeLine(10, 10, getWidth() - 10, 10);
        gc2d.fillOval(getWidth() / 5 - 50, getHeight() / 9 - 50, getWidth() / 8,
getWidth() / 8);
        gc2d.strokeOval(2 * getWidth() / 5 - 50, getHeight() / 9 - 50, getWidth() /
8, getWidth() / 8);
        gc2d.fillRoundRect(3 * getWidth() / 5 - 50, getHeight() / 9 - 50,
getWidth() / 8, getWidth() / 8, 15, 15);
        gc2d.strokeRoundRect(4 * getWidth() / 5 - 50, getHeight() / 9 - 50,
getWidth() / 8, getWidth() / 8, 15, 15);
        gc2d.fillArc(getWidth() / 5, 3 * getHeight() / 9 - 40, getWidth() / 8,
getWidth() / 8, 0, 270, ArcType.OPEN);
        gc2d.fillArc(2 * getWidth() / 5, 3 * getHeight() / 9 - 40, getWidth() / 8,
getWidth() / 8, 0, 270, ArcType.CHORD);
        gc2d.fillArc(3 * getWidth() / 5, 3 * getHeight() / 9 - 40, getWidth() / 8,
```

```
getWidth() / 8, 0, 270, ArcType.ROUND);
        gc2d.strokeArc(getWidth() / 5, 5 * getHeight() / 9 - 30, getWidth() / 8,
getWidth() / 8, 0, 270, ArcType.OPEN);
        gc2d.strokeArc(2 * getWidth() / 5, 5 * getHeight() / 9 - 30, getWidth() /
8, getWidth() / 8, 0, 270, ArcType.CHORD);
        gc2d.strokeArc(3 * getWidth() / 5, 5 * getHeight() / 9 - 30, getWidth() /
8, getWidth() / 8, 0, 270, ArcType.ROUND);
        gc2d.fillPolygon(new double[]{getWidth() / 5, 13 * getWidth() / 40,
getWidth() / 5, 13 * getWidth() / 40},
                    new double[]{7 * getHeight() / 9, 7 * getHeight() / 9, 7 *
getHeight() / 9 + getWidth() / 8, 7 * getHeight() / 9 + getWidth() / 8}, 4);
        gc2d.strokePolygon(new double[]{2 * getWidth() / 5, 21 * getWidth() / 40, 2
* getWidth() / 5, 21 * getWidth() / 40},
                    new double[]{7 * getHeight() / 9, 7 * getHeight() / 9, 7 *
getHeight() / 9 + getWidth() / 8, 7 * getHeight() / 9 + getWidth() / 8}, 4);
        gc2d.strokePolyline(new double[]{3 * getWidth() / 5, 29 * getWidth() / 40,
3 * getWidth() / 5, 29 * getWidth() / 40},
                        new double[]{7 * getHeight() / 9, 7 * getHeight() / 9, 7 *
getHeight() / 9 + getWidth() / 8, 7 * getHeight() / 9 + getWidth() / 8}, 4);
    }

}
```

Klasa DrawingMain:

```
import javafx.application.Application;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.scene.Scene;
import javafx.scene.layout.AnchorPane;
import javafx.stage.Stage;

public class DrawingMain extends Application {

    private MyCanvas canvas = new MyCanvas();

    @Override
    public void start(Stage primaryStage) {
        // kreiranje AnchorPane korenog container-a
        AnchorPane root = new AnchorPane();
        Scene scene = new Scene(root, 600, 600);
        // ubacivanje canvas objekta unutar root container-a
        root.getChildren().add(canvas);

        primaryStage.setTitle("Drawing JavaFX!");
        primaryStage.setScene(scene);
        primaryStage.show();

        // listener-i koji osluskuju promenu velicine scene unutar stage-a
        scene.widthProperty().addListener(new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> ov, Number
oldSceneWidth, Number newSceneWidth) {
```

```
                canvas.setWidth((double) newSceneWidth);
                canvas.draw();
            }
        });
        scene.heightProperty().addListener(new ChangeListener<Number>() {
            @Override
            public void changed(ObservableValue<? extends Number> observableValue,
Number oldSceneHeight, Number newSceneHeight) {
                canvas.setHeight((double) newSceneHeight);
                canvas.draw();
            }
        });

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);
    }

}
```

PRIMER 7 - OBJAŠNJENJE

Cilj sekcije je da se pojasni kod primera 7

Ovaj primer demonstrira iscrtavanje raznih objekata koji pripadaju potklasama klase **Shape** koji su navedeni u predavanjima. Koristimo klasu **GraphicsContext2D** za rad sa 2D grafikom. Klasa nam pruža razne metode sa isrctavanje različitih oblika, kao i za podešavanje boja, debljine linije itd. Treba znati da objekat ove klase čuva trenutno stanje nekih osobina za crtanje, na primer ako postavite boju na crnu, sve što posle toga crtate biće crno, ako želite da naredni objekat bude crvene boje, onda morate da postavite boju GraphicContext-a na crvenu.

Ovde se koristi kontejner **AnchorPane** koji ima mogućnost da prati promene dimenzija prozora i omogućava nam da obrađujemo događaje promene visine ili širine prozora što je korisno u slučaju da želimo da pravimo prilagodljive crteže tj. crteže koji menjaju dimenzije u skladu sa promenom veličine prozora.

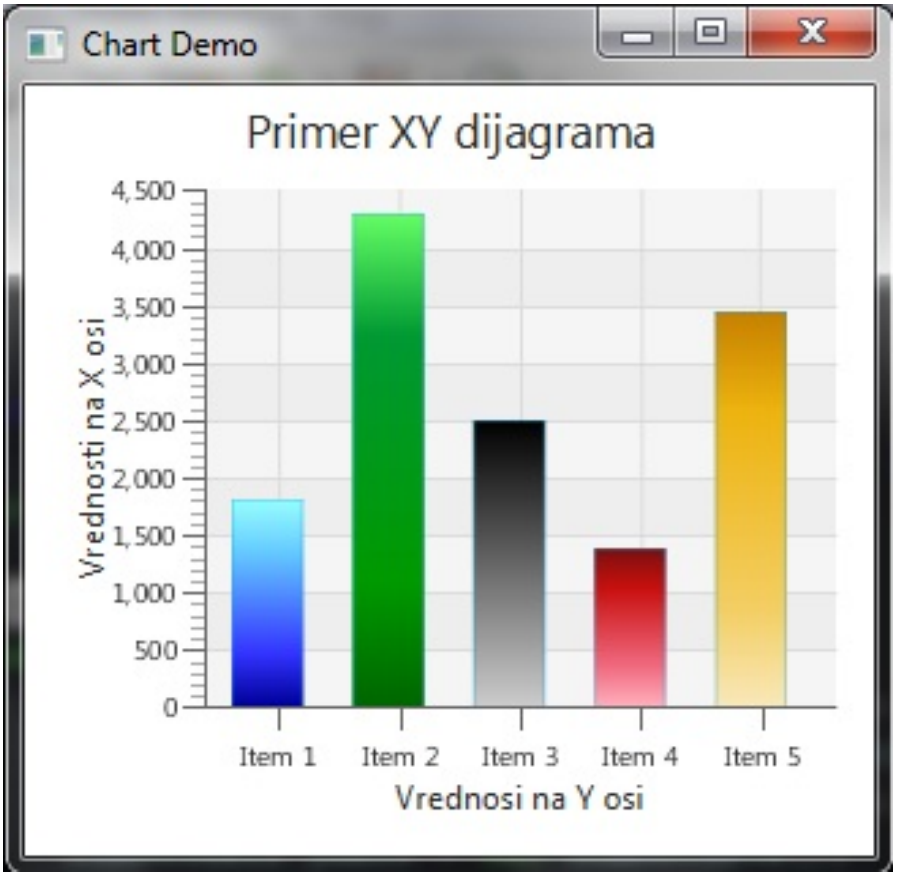
Iscrtajte više linija pomoću strokeLine metoda teko da dobijete prvo slovo Vašeg imena.

PRIMER 8

Cilj zadatka je da se prikaže rad sa dijagramom i korišećenje CSS-a za stilizovanje.

Koristeći JavuFX (BarChart klasu), napraviti dijagram sa X i Y osama na kojem će na X osi biti prikazani tekstualni elementi dok će se na Y osi nalaziti odgovarajuće numeričke vrednosti. Takođe koristeći CSS potrebno je stilizovati pravouganike koji predstavljaju podeoke na dijagramu. Vrednosti na X i Y osama mogu biti proizvoljne.

Potrebno vreme: 10 minuta



Slika 10.6 .6: Primer XY dijagrama

PRIMER 8 - REŠENJE

Prikaz programskog koda primer 8

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.scene.Scene;
import javafx.scene.chart.BarChart;
import javafx.scene.chart.CategoryAxis;
import javafx.scene.chart.NumberAxis;
import javafx.scene.chart.XYChart;
import javafx.stage.Stage;

public class ChartMain extends Application {

    @Override
    public void start(Stage stage) {
```

```
        // definisemo da ce vrednosti na X osi biti tipa string tj. tekstulane
        CategoryAxis xAxis = new CategoryAxis();
        // definisemo da ce vrednosti na Y osi bit brojevi
        NumberAxis yAxis = new NumberAxis();
        // kreiramo objekat klase BarChart koji prikazuje koordinatni sistem sa X i
Y osama i pri kreiranju
        // navodimo kog tipa ce biti X i Y ose
        BarChart<String, Number> bc = new BarChart<>(xAxis, yAxis);
        // postavljamo naslov dijagrama kao i naziv X i Y osa
        bc.setTitle("Primer XY dijagrama");
        xAxis.setLabel("Vrednosi na Y osi");
        yAxis.setLabel("Vrednosti na X osi");
        // kriramo objekat klase XYChart.Series koji cini kolekciju parova podataka
za X i Y osu
        XYChart.Series series = new XYChart.Series();
        // dodajemo vrendnosti za X i Y osu u parovima kao objekat XYChart.Data
klase
        series.getData().add(new XYChart.Data("Item 1", 1800.58));
        series.getData().add(new XYChart.Data("Item 2", 4300.40));
        series.getData().add(new XYChart.Data("Item 3", 2500.50));
        series.getData().add(new XYChart.Data("Item 4", 1390.75));
        series.getData().add(new XYChart.Data("Item 5", 3450.89));

        Scene scene = new Scene(bc, 600, 600);
        // ucitavamo css fajl u kome cemo stilizovati dijagram

//scene.getStylesheets().add(ChartMain.class.getResource("stylechars.css").toString(
));
        bc.getData().add(series);
        bc.setLegendVisible(false);
        stage.setTitle("Chart Demo");
        stage.setScene(scene);
        stage.show();
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        launch(args);

    }
}
```

PRIMER 8 - OBJAŠNJENJE

Pojašnjava kod rešenja primera 8

Definišemo tipove vrednosti na osama. **CategoryAxis** nam označava dad će biti string, a **NumberAxis** da će biti brojevi. Zatim se kreira objekat **BarChart** klase sa te dve ose. Podatke unosimo u objekat tipa **XYChart.Series**, gde se nalazre parovi podataka, string i

broj. Par kreiramo sa **XYChart**.**Data** i ubacujemo u series objekat. Na kraju dodajemo CSS na scenu sa **getStylesheets().add.**

Poglavlje 11

Vežba – Zadaci za samostalni rad

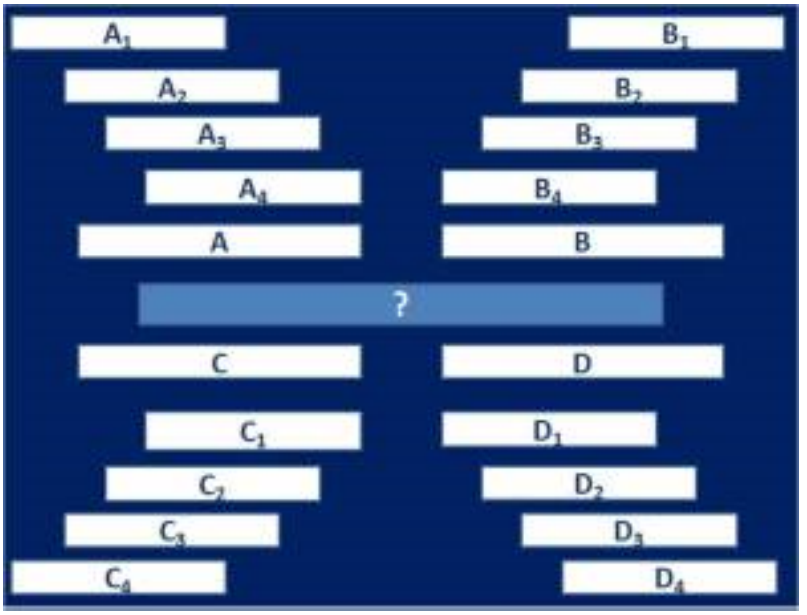
ZADACI ZA SAMOSTALAN RAD

Ove zadatke student treba da radi samostalno za vreme individualnih vežbi.

1. Napraviti login formu koristeći GridPane kontainer i stilizovati je pomoću CSS-a. Login forma treba da ima polja email i password, kao i dugme za potvrdu. (25 minuta)
2. Napisati program koji iscrtava n krugova iste veličine organizovanih u strukturu matrice. Program treba da prima kao parametar broj vrsta i kolona krugova i krugovi treba da budu različitih boja. (25 minuta)
3. Napraviti program koji kao parametar može da primi brojeve od 1 do 6 (kao na kocki za igru) i u zavisnosti od primljenih parametara na ekranu iscrtava izgled kocke za igru. Raspored krugova treba da bude kao na kocki za igru. Potruditi se da krugovi budu prilagodljivi na promenu veličine ekrana. (25 minuta)

4. Isprobati rad sa dijagramima koristeći „Pie“ strukturu dijagrama. Može se koristiti gotova JavaFX klasa PieChart ili se realizovati sopstveno rešenje iscrtavanjem delova krugova. (25 minuta)

5. Koristeći JavaFX napraviti strukturu dugmića u obliku X slova kao u igri asocijacija na slagalici. Dugmiće stilizovati korišćenjem CSS-a. (35 minuta)



Slika 11.1 .1: Slika dugmića

Zaključak

REZIME

Glavne poruke lekcije

1. JavaFX je novi radni okvir za razvoj bogate Internet aplikacije. JavaFX u celosti zamenjuje Swing i AWT.
2. Metod main() u JavaFX mora da proširi javafx.application.Application i primenjuju metod start(). Primarna pozornica se automatski kreira od strane JVM i ubacuje u metod start().
3. Pozornica je prozor za prikaz scene. Možete dodavati čvorove u scenu. Okna (panes), kontrole (controls) i oblici (shapes) su čvorovi. Okna se mogu da koriste kao kontejneri u čvorova.
4. Svojstvo povezivanja se može vezati za izvorni objekat koji se posmatra (observable). Promena u izvornom objektu će se automatski odraziti na povezano svojstvo. Svojstvo povezivanja ima vrednost u geter metodu, vrednost u seter metodu i svojstvo u geter metodu.
5. Klasa Node definiše mnoga svojstva koja su zajednička ya sve švorove. Mogu se primeniti kod okana, kontrola i oblika.
6. Možete kreirati Color objekat sa specificiranom crvenom, zelenom i plavom komponentom, a vrednostu prozirnosti.
7. Možete kreirati objekat Font, i uneti njegov naziv, veličinu, širinu i postavka.
8. Klasa javafx.scene.image.Image se koriste za dovođenje slike i njeno prikazivanje u ImageView objektu.
9. JavaFX obezbeđuje mnogo tipova okana za automatsko raspoređivanje čvorova u željenu lokaciju i veličinu. Klasa Pane je osnovna klasa ya sva okna. Sadrži metod getChilidren() koji vraća listu ObservableList. Možete upotrebiti metode add(node) i addAll(node1, node 2, ...) radi dodavanja čvorova u okno.
10. Okno FlowPane uređuje čvorove horizontalno, s leva u desno, ili vertikalno, od gore ka dole i redosledu njihovog dodavanja. Okno GridPane uređuje čvorove u formatu matrice. Čvorovi se stavljaju u određenu kolonu i red. Okno BorderPane postavlja čvorove u pet regiona, na vrhu, dole, levo, desno i u centar. Hbox stavlja decu u jedan horizonatalni red, a VBox ih postavlja vertikalno i jednu kolonu.
11. JavaFX obezbeđuje mnogo klasa oblika.

REFERENCE

Literatura primenjena u ovoj lekciji

Osnovna literatura :

1. Y. Daniel Liang, INTRODUCTION TO JAVA PROGRAMMING (COMPREHENSIVE VERSION), Tenth Edition, Pearson, ISBN 10: 0-13-376131-2, ISBN 13: 978-0-13-376131-3, 2014 . Ovo je osnovni udžbenik za ovaj predmet i preporučuje se studentima da ga koriste,

Dodatna literatura:

1. Barbara Liskov with John Guttag. Program Development in Java. Addison Wesley, 2001, ISBN 0-201-65768-6.
2. Joshua Bloch. Effective Java. Second Edition. Addison Wesley, 2008, ISBN 0-321-25668-3. Note that the Second edition is newly released (May 2008);
3. Java - How To Program,Paul Deitel, Harvey Deitel, 9th_Edition, Pearson,ISBN 978-0-13-257566-9, 2012

Preporučeni onlajn Java kursevi:

1. https://docs.oracle.com/javase/tutorial/
2. http://www.javatpoint.com/java-tutorial
3. https://www.ibm.com/developerworks/learn/java/intro-to-java-course/