PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET
METROPOLITAN
BEOGRAD

KI105 - JAVA 3: PROGRAMIRANJE KORISNIČKOG INTERFEJSA

JavaFX UI kontrole i multimedija

Lekcija 05

PRIRUČNIK ZA STUDENTE

KI105 - JAVA 3: PROGRAMIRANJE KORISNIČKOG INTERFEJSA

Lekcija 05

JAVAFX UI KONTROLE I MULTIMEDIJA

- JavaFX UI kontrole i multimedija
- Poglavlje 1: Klase Labeled i Label
- Poglavlje 2: Klasa Button
- Poglavlje 3: Klasa CheckBox
- Poglavlje 4: Klasa RadioButton
- Poglavlje 5: Klasa TextField
- Poglavlje 6: Klasa TextArea
- Poglavlje 7: Klasa ComboBox
- Poglavlje 8: Klasa ListView
- Poglavlje 9: Klasa ScrollBar
- Poglavlje 10: Klasa Slider
- Poglavlje 11: Video i Audio
- Poglavlje 12: Domaći zadaci
- Zaključak

UVOD

Ciljevi lekcije. Za ovu lekciju predviđene su dve nedelje nastave.

1. Kreiranje grafičkog korisničkog interfejsa sa različitim kontrolama interfejsa sa korisnikom
2. Kreiranje natpisa sa tekstom i grafikom upotrebom klase ***Label*** i ispitivanje svojstava apstraktne klase Labeled.
3. Kreiranje dugmeta sa tekstom i grafikom upotrebom klase ***Button*** i postavljanje obrađivača upotrebom metoda***setOnAction*** metoda iz klase ***ButtonBase***.
4. Kreiranje polja za potvrđivanje upotrebom klase ***CheckBox***
5. Kreiranje dugmeta opcije (radio-dugmeta) upotrebom klase ***RadioButton*** i grupe dugmedi opcija upotrebom klase***ToggledGroup***
6. Unošenje podataka upotrebom klase ***TextField*** i lozinke upotrebom klase ***PasswordField.***
7. Unos podataka sa više linija upotrebom klase ***TextField***
8. Izbor jedne opcije upotreebom klase ***ComboBox***
9. Izbor jedni ili više opcija upotrebom***ListView***
10. Izbor opsega vrednosti upotrebom klase ***ScrollBar***
11. Izbor opsega vrednosti upotrebom klase ***Slider*** i ispitavnje razlika između klasa***ScrollBar*** i ***Slider***
12. Razvoj tic-toe igre
13. Prikazivanje video i audio sadržaja upotrebom klasa ***Media, MediaPlayer***, i ***MediaView***
14. Razvoj studije slučaja prikaza nacionalne yastave i sviranje himne

Referenca: 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

Ovo je osnovni udžbenik za ovaj predmet i preporučuje se studentima da ga koriste,

Poglavlje 1

Klase Labeled i Label

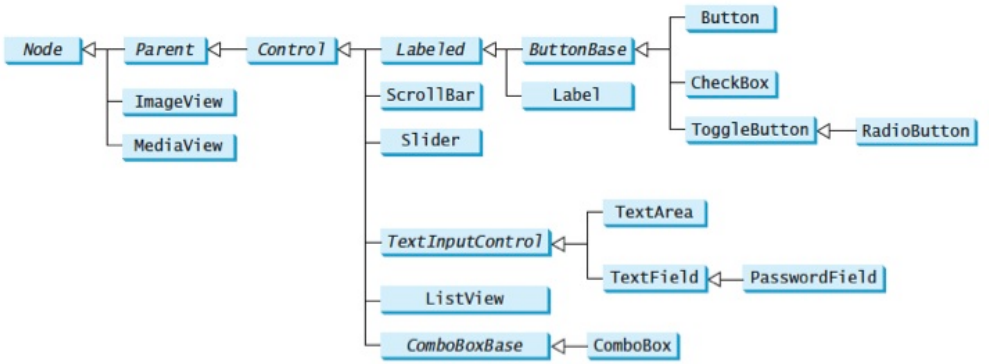
JAVAFX KLASE ZA KONTROLU UI INTERAKCIJE SA KORISNIKOM

JavaFX obezbeđuje puno klasa za kontrolu ulazno-izlaznih (UI) interakcija korisnika sa softverskim sistemom, što omogućava razvoj obimnog korisničkog interfejsa (GUI).

Grafički korisnički interfejs, koji se obično skraženo označava sa GUI (GraphicUser Interface) olakšava rad korisnika sa sistemom. Da bi se kreirao dobar GUI, neophodna je kreativnost i znanje o tome kako se odvijaju UI (ulazno-izlane) operacije kontrole interakcije korisnika sa sistemom.

Postoje alati koji znatno olakšavaju kreiranje GUI (npr. Orarcle), sa minimumom programiranja. Međutim, često morate da iyvršite neke promene u tako kreiranom interfejsu, a ya to vam je neophodno dobro poynavanje JavaFX klasa koje se koriste za kreiranje GUI interfejsa. Zato, pre nego što počnete da koristite alate za kreiranje GUI interfejsa, neophodno je naučite osnovne koncepte JavaFX GUI programiranja.

Slika 1 prikazuje JavaFX klase koje se najčešće koriste pri kreiranju GUI interfejsa.



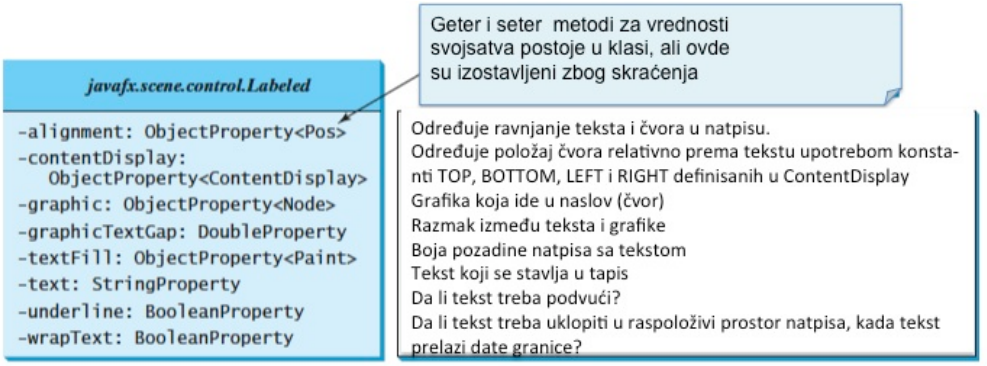
Slika 1.1 JavaFX klase koje se najčešće koriste za kontrolu UI interakcija i u razvoju GUI interfejsa

SVOJSTVA KLASA LABELED I LABEL

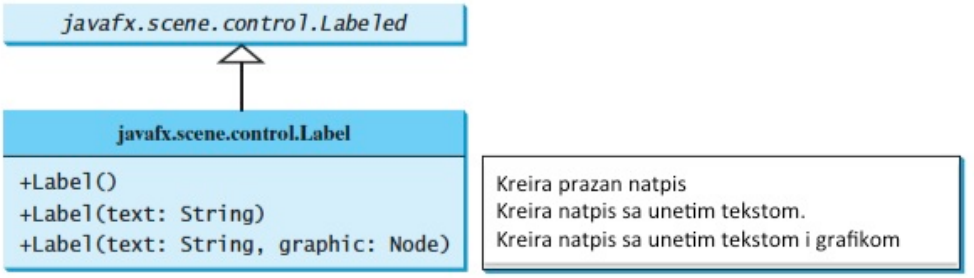
Klasa Label kreira natpis u kome se prikazuje tekst, neki čvor , ili i jedno i drugo. Klasa Labeled sadrži zajednička svojstva koja dele klase Label, Button, CheckBox i RadioButton

Natpis (engl. label) je prostor u kome se prikazuje neki kratki tekst, ili neki čvor, ili i jedno i drugo. Obično se koristi za označavanje drugih UI kontrola (najčešće pola sa tekstom). Kako natpisi i dugmad (klase ***Label*** i ***Button***) dele mnoga zajednička svojstva, ***JavaFX*** obzbeđuje klasu ***Labeled*** od koje klase***Label*** i ***Button*** nasleđuju ta zajednička svojstva, kao što je prikazano na slici 2.

Svojstva klase ***Label*** prikazana su na UML dijagramu na slici 3. Objekat klase ***Label*** se može kreirati primenom jednog od tri data konstruktora.



Slika 1.2 Klasa Labeled definiše zajednička svojstva klase Label, Button, CheckBox, i RadioButton

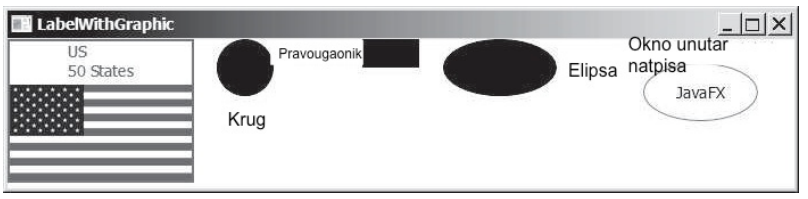


Slika 1.3 Klasa Label kreira natpis sa tekstom, ili čvorom ili sa oba

PRIMER 1 - UPOTREBA NATPISA

Primer pokazuje kreiranje različitih natpisa sa tekstom i grafikom.

Bilo koje grafičko svojstvo (oblik, slika), kao čvor se može ubaciti u natpis. Na slici Prikayani su rayličiti natpisi koji se kreiraju primenom klase LabelWithGraphic čiji je listing ovde prikazan.



Slika 1.4 Program kreira natpise sa tekstom i grafikom

Program kreira natpis sa tekstom i slikom (linija 19). Linija 21 određuje da se slika postavlja ispod teksta. Program kreira natpis sa tekstom i krugom (linija 24). Krug se postavlja iznad

teksta (linija 25). Program kreira natpis sa tekstom i pravougaonikom. (linija 28). Pravougaonik se postavlja desno u odnosu na tekst. (linija 29). Program kreira natpis sa tekstom i elipsom (linija 31). Elipsa se postavlja levo od teksta (linija 32). Program kreira elipsu u stek oknu(linija 38). I kreira natpis sa tekstom i stek oknom kao čvor (linija 39). Program kreira Hboc okno (linija 43).

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.ContentDisplay;
import javafx.scene.control.Label;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.StackPane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.scene.shape.Ellipse;
import javafx.scene.shape.Rectangle;
import javafx.stage.Stage;

public class LabelWithGraphic extends Application {

    @Override // Redefinisanje metoda start() klase
    public void start(Stage primaryStage) {
        //primer sa slikom unutar label-e
        ImageView us = new ImageView(new Image("us.jpg"));
        //postavljanje velicine slike
        us.setFitHeight(50);
        us.setFitWidth(100);
        Label lb1 = new Label("US\n50 States", us);
        lb1.setStyle("-fx-border-color: green; -fx-border-width: 2");
        lb1.setContentDisplay(ContentDisplay.BOTTOM);
        lb1.setTextFill(Color.RED);

        //primer sa grafičkim objektom iznad teksta labele
        Label lb2 = new Label("Krug", new Circle(50, 50, 25));
        lb2.setContentDisplay(ContentDisplay.TOP);
        lb2.setTextFill(Color.ORANGE);

        //primer sa grafičkim objektom desno od teksta labele
        Label lb3 = new Label("Pravougaonik", new Rectangle(10, 10, 50, 25));
        lb3.setContentDisplay(ContentDisplay.RIGHT);

        //primer sa grafičkim objektom levo od teksta labele
        Label lb4 = new Label("Elipsa", new Ellipse(50, 50, 50, 25));
        lb4.setContentDisplay(ContentDisplay.LEFT);

        //primer sa grafičkim objektom koji u sebi sadrzi tekst i nalazi se ispod
teksta labele
        Ellipse ellipse = new Ellipse(50, 50, 50, 25);
        ellipse.setStroke(Color.GREEN);
        ellipse.setFill(Color.WHITE);
        StackPane stackPane = new StackPane();
```

```
        stackPane.getChildren().addAll(ellipse, new Label("JavaFX"));
        Label lb5 = new Label("A pane inside a label", stackPane);
        lb5.setContentDisplay(ContentDisplay.BOTTOM);

        HBox pane = new HBox(20);
        pane.getChildren().addAll(lb1, lb2, lb3, lb4, lb5);

        //kreiranje scene i postavljanje stene na pozornicu
        Scene scene = new Scene(pane, 650, 100);
        primaryStage.setTitle("LabelWithGraphic"); //unos naziva pozornice
        primaryStage.setScene(scene); //postavljanje scene na pozornicu
        primaryStage.show(); //prikaz pozornice
    }

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

ZADATAK 1 - DOPUNA PRIMERA SA NATPISIMA

Dopunite prethodni primer sa natpisima.

Proširite prethodni primer:

- dodajte poligon po želji;
- dodajte polililiniju po želji;
- za oba objekta dodajte labele, koje ih opisuju, na mesto po želji.

Poglavlje 2

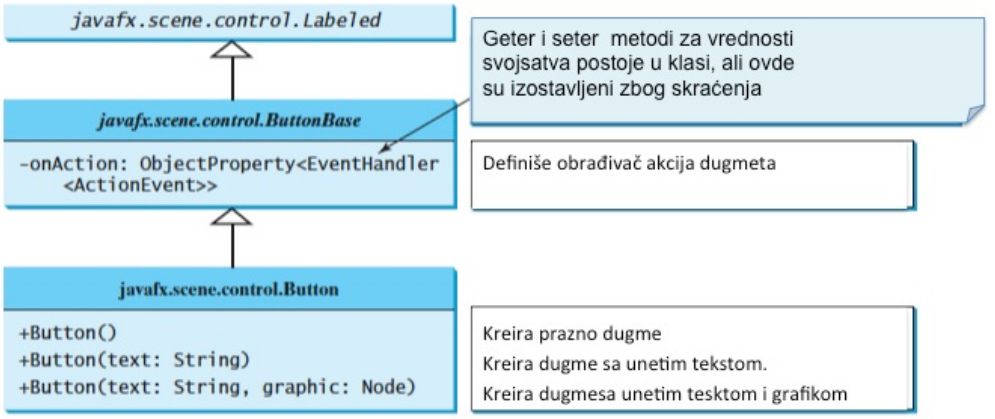
Klasa Button

SVOJSTVA KLASE BUTTON

Dugme (klasa Button) je kontrola koja pokreće jedan događaj akcije kada se klikne

Dugme je kontrola koja pokreće jedan događaj akcije kada se klikne. JavaFX obezbeđuje uobičajenu dugmad, preklopnu dugmad, dugmad za potvrđivanje i dugmad za opcije. Zajednička svojstva ovih različitih digmadi su definisani u klasama ***ButtonBase*** i ***Labeled***, kao što je to pokazano na slici 1

Klasa Labeled definiše zajednička svojstva .svih objekat akoji definišu natpise i dugmad. Dugme je kao i natpis, sem što dugme ima svojstvo ***onAction*** koje je definisano u klasi ***ButtonBase***. Ovaj metod postavlja obrađivača za obradu akcije dugmeta.

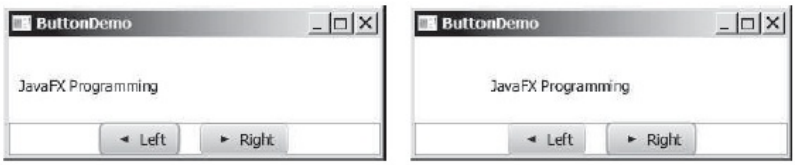


Slika 2.1 Klasa Button i ButtonBase

PRIMER 2 - KORIŠĆENJE DUGMETA

Listing klase ButtonDemo prikazuje program koji koristi dugmad za kontrolu kretanja teksta

Listing klase ButtonDemo prikazuje program koji koristi dugmad za kontrolu kretanja teksta, kao što je prikazano na slici 2.



Slika 2.2 Program demonstrira upotrebu digmadi

Program kreira dva dugmeta: btLeft i btRight. Oba sadrže i tekst i sliku (linije 17-20). Dugmad su postavljeni u okno Hbox (linija 21) a Hbox okno je je postavljeno na donjoj strani okna sa granicama (linija 30). Obraživač akcija za btRight dugme pokreže tekst udesno (linija 33). Program sa svrhom definiše zaštićeni ***getPane()*** metod da bi vratio okno (linija 15). Ovaj metod biće oredefisan u poklasam u drugim primerima a da bi se dodali novi čvorovi u okno. Tekst se deklariše kao zaštićen, tako d amože da mu se prilayzi samo od njegovih podklasa (linija 13).

```
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.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class ButtonDemo extends Application {

    Text text = new Text(50, 50, "JavaFX Programming");

    public BorderPane getPane() {
        HBox paneForButtons = new HBox(20);
        // kreiranje dugmeta za levo
        ImageView img_left = new ImageView("left.jpg");
        img_left.setFitHeight(50);
        img_left.setFitWidth(50);
        Button btLeft = new Button("Left", img_left);

        //kreiranje dugmeta za desno
        ImageView img_right = new ImageView("right.jpg");
        img_right.setFitHeight(50);
        img_right.setFitWidth(50);
        Button btRight = new Button("Right", img_right);

        //dodavanje panela sa dugmucima na border layout
        paneForButtons.getChildren().addAll(btLeft, btRight);
        paneForButtons.setAlignment(Pos.CENTER);
        paneForButtons.setStyle("-fx-border-color: green");

        BorderPane pane = new BorderPane();
```

```
        pane.setBottom(paneForButtons);

        //dodavanje teksta na centar BorderPane-a
        Pane paneForText = new Pane();
        paneForText.getChildren().add(text);
        pane.setCenter(paneForText);

        //dodavanje akcija na dugmice
        btLeft.setOnAction(e -> text.setX(text.getX() - 10));
        btRight.setOnAction(e -> text.setX(text.getX() + 10));

        return pane;
    }

    @Override //redefinisanje metoda start() klase Application
    public void start(Stage primaryStage) {
        //kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(getPane(), 450, 200);
        primaryStage.setTitle("ButtonDemo"); //unos naslova pozornice
        primaryStage.setScene(scene); //postavljanje scene na scenu
        primaryStage.show(); //prikaz pozornice
    }

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

ZADATAK 2 - DOPUNA PRIMERA SA KORIŠĆENJEM DUGMETA

Dopunite primer sa korišćenjem dugmeta

Proširite prethodni primer:

- dodajte dugme "povećaj";
- dodajte dugme "smanji";
- pozicija ova dva dugmeta je između postojećih dugmadi;
- klikom na prvo dugme povećati veličinu fonta prikazanog teksta;
- klikom na drugo dugme smanjiti veličinu fonta prikazanog teksta;

Poglavlje 3

Klasa CheckBox

SVOJSTVA KLASE CHECKBOX

Klasa CheckBox omogućava korisniku da izabere neku opciju.

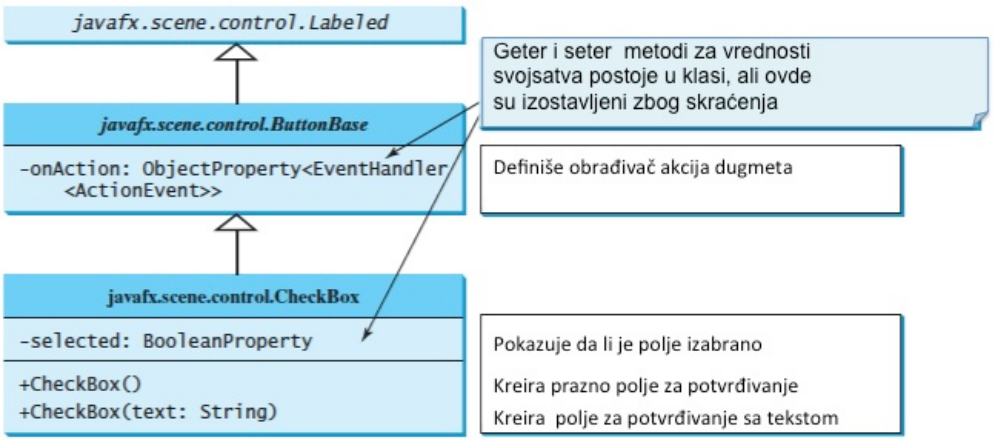
Klasa ***CheckBox*** omogućava korisniku da potvrdi nešto. Kao i klasa ***Button***, i klasa ***CheckBox*** nasleđuje sva svojstva kao što su ***onAction, text, graphic, alignment, grašhicTextCap, textFill. concetentDisplay*** i klasa ***ButtomBase*** i ***Labled***, kao što je prikazano na slici 1.

Pored toga, obezbeđuje svojstvo potvrđivanja, da bi pokazalo da li je izvršena potvrda u nekom polju .

Kada se klikne polje potvrđivanja, bilo da se stavlja ili skida oznaka potvrđivanja, to stvara događaj tipa ***ActionEvent.***

Da bi se odredilo da li je opcija koju nudi polje za potvrđivanje odabrana, upotrebljava se metod ***isSelected().***

.



Slika 3.1 Klasa CheckBox sadrži svojstva koja nasleđuje od klasa ButtonBase i Labeled

PRIMER 3 - KORIŠĆENJE KLASE CHECKBOX

Primer daje listing klase CheckBoxDemo koje kreira i koristi polja za potvrđivanje nekog svojstva (Bold ili Italic), kao podklasa klase ButtonBoxDemo.

Radi prikaza korišćenja klase ***CheckBox,*** napisaćemo program koji dodaje dva polja za odabiranje pod nazivom **Bold** i **Italic**, da bi se omogućilo da korisnik odabere pojavno svojstvo fonta koji koristi. Slika 2 prikazuje GUI interfejsa, a daje se i listing klase **CheckBoxDemo** koja kreira i koristi ova GUI interfejs.

SLika-2: GUI primer sa poljima za izbor opcije

Postoje najmanje dva načina pisanja ovog programa. Prvi se svodi na ponavljanje programa za klasu **ButtonDemo**, s tim što se u njega ubacuje kod za dodavanje polja za izbor opcije i za obradu generisanog događaja. Drugi način je da se definiše podklasa koja proširuje klasu **ButtonDemo.** Ovde se daje listing ovog drugog načina, a predlažemo vam da sami uradite program primenom prvog načina programiranja datog problema.

Klasa **CheckBoxDemo** proširuje klasu **ButtonDemo** i redefiniše metod ***getPane()*** (lina 13). Novi metod ***getPane()*** poziva ***super.getPane()*** metod iz klase **ButtonDemo** da bi se dobilo granično okno koje sadrži dugmad i tekst (linija 14)..

Polja za odabiranje se kreiraju i dodaju u okno ***paneForCheckBoxes*** (linija 30-32), a ovo okno se onda dodaje u granično okno (linij 33). Obrađivač koji obrađuje događja kacije je kreiran u linijama 35-48. On podešava odgovarajući font i njegov pojavni oblik u skladu sa odabranim svojstvima.

Klasa CheckBoxDemo:

```
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.control.CheckBox;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;

public class CheckBoxDemo extends ButtonDemo {

    @Override // Redefinisanje metoda start() klase Application
    public BorderPane getPane() {
        BorderPane pane = super.getPane();

        Font fontBoldItalic = Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.ITALIC, 20);
        Font fontBold = Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.REGULAR, 20);
        Font fontItalic = Font.font("Times New Roman",
                FontWeight.NORMAL, FontPosture.ITALIC, 20);
        Font fontNormal = Font.font("Times New Roman",
                FontWeight.NORMAL, FontPosture.REGULAR, 20);
```

```
        text.setFont(fontNormal);

        VBox paneForCheckBoxes = new VBox(20);
        paneForCheckBoxes.setPadding(new Insets(5, 5, 5, 5));
        paneForCheckBoxes.setStyle("-fx-border-color: green");
        CheckBox chkBold = new CheckBox("Bold");
        CheckBox chkItalic = new CheckBox("Italic");
        paneForCheckBoxes.getChildren().addAll(chkBold, chkItalic);
        pane.setRight(paneForCheckBoxes);

        EventHandler<ActionEvent> handler = e -> {
            if (chkBold.isSelected() &amp;amp;amp;amp; chkItalic.isSelected()) {
                text.setFont(fontBoldItalic); // Oba polja su odabrana
            } else if (chkBold.isSelected()) {
                text.setFont(fontBold); // Odabrano je polje Bold
            } else if (chkItalic.isSelected()) {
                text.setFont(fontItalic); // Odabrano je polje Italic
            } else {
                text.setFont(fontNormal); // Oba polja za potvrđivanje nisu
odabrana
            }
        };

        chkBold.setOnAction(handler);
        chkItalic.setOnAction(handler);

        return pane; // Vraća novi okvir
    }

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

Klasa ButtonDemo:

```
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.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class ButtonDemo extends Application {

    Text text = new Text(50, 50, "JavaFX Programming");
```

```
    public BorderPane getPane() {
        HBox paneForButtons = new HBox(20);
        // kreiranje dugmeta za levo. Potrebno je u projekat dodati sliku sa
nazivom left.jpg
        ImageView img_left = new ImageView("left.jpg");
        img_left.setFitHeight(50);
        img_left.setFitWidth(50);
        Button btLeft = new Button("Left", img_left);

        //kreiranje dugmeta za desno. Potrebno je u projekat dodati sliku sa
nazivom right.jpg
        ImageView img_right = new ImageView("right.jpg");
        img_right.setFitHeight(50);
        img_right.setFitWidth(50);
        Button btRight = new Button("Right", img_right);

        //dodavanje panela sa dugmucima na border layout
        paneForButtons.getChildren().addAll(btLeft, btRight);
        paneForButtons.setAlignment(Pos.CENTER);
        paneForButtons.setStyle("-fx-border-color: green");

        BorderPane pane = new BorderPane();
        pane.setBottom(paneForButtons);

        //dodavanje teksta na centar BorderPane-a
        Pane paneForText = new Pane();
        paneForText.getChildren().add(text);
        pane.setCenter(paneForText);

        //dodavanje akcija na dugmice
        btLeft.setOnAction(e -> text.setX(text.getX() - 10));
        btRight.setOnAction(e -> text.setX(text.getX() + 10));

        return pane;
    }

    @Override //redefinisanje metoda start() klase Application
    public void start(Stage primaryStage) {
        //kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(getPane(), 450, 200);
        primaryStage.setTitle("ButtonDemo"); //unos naslova pozornice
        primaryStage.setScene(scene); //postavljanje scene na scenu
        primaryStage.show(); //prikaz pozornice
    }

}
```

ZADATAK 3- DOPUNA PRIMERA SA KORIŠĆENJEM KLASE CHECKBOX

Dopunite primer sa korišćenjem dugmeta

Proširite prethodni primer:

- dodajte CheckBox "velika slova";
- dodajte CheckBox "mala slova";
- pozicija ova dva CheckBox-a je ispod postojećih CheckBox kontrola;
- izborom prve opcije prikazani tekst će ima ti sva velika slova;
- izborom prve opcije prikazani tekst će ima ti sva mala slova;
- u slučaju izbora obe opcije izgled teksta se neće menjati.

VIDEO - CHECKBOX

JavaFX Java GUI Tutorial - 11 - CheckBox (7,37 minuta)

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

Poglavlje 4

Klasa RadioButton

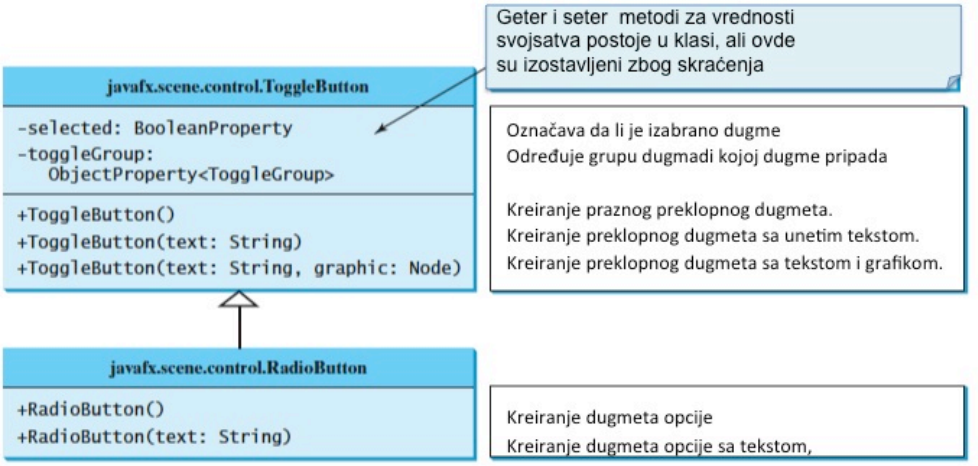
SVOJSTVA KLASE RADIOBUTTON

Radio dugme, tj. dugme opcije, omogućava korisniku da izabere opciju iz grupe izbora.

Radio dugme, tj. dugme opcije , omogućava korisniku da izabere opciju iz grupe izbora. Dugme opcije za razliku od dugmeta potvrđivanja, koji ima kvadrat za unos odabira, koristi krug za unos izbora opcije (tačka u krugu).

Klasa ***RadioButton*** je podklasa klase ***ToggleButton***. Razlika između ovih klasa je u tome što dugme opcije (radio-dugme) pokazuje krug, a dugme prekidač se crta slično kao dugme. UML dijagram klas ***ToggleButton*** i ***RadioButton*** je prikazan na slici 1. Ovde se daje primer dela programa koji kreira i bira dugme opcija, sa slikom tekstom i crnim okvirom. (slika 2)

Slika 4.1 Primer dugme opcije sa slikom i tekstom



Slika 4.2 Klase ToggleButton i RadioBox su dugamd za iizbor opcija

```
RadioButton rbUS = new RadioButton("US");
 rbUS.setGraphic(new ImageView("image/usIcon.gif"));
 rbUS.setTextFill(Color.GREEN);
 rbUS.setContentDisplay(ContentDisplay.LEFT);
 rbUS.setStyle("-fx-border-color: black");
```

```
rbUS.setSelected(true);
rbUS.setPadding(new Insets(5, 5, 5,));
```

GRUPA DUGMADI OPCIJA

Više dugmadi opcija mogu da čine jednu grupu. To omogućava povezivanje izbora opcija koje nude, te se može, na primer, izabrati samo jedna opcija, tj. samo jedno dugme.

Da napravili grupu dugmadi opcija (radio-dugmadi), treba da kreirate primerak klase ***ToggleGroup*** i postaviti svojstvo ***toggleGroup*** u dugmetu opcije da bi dugme pridodali grupi, kao što pokazuje sledeći deo programa:

```
ToggleGroup group = new ToggleGroup();
 rbRed.setToggleGroup(group);
 rbGreen.setToggleGroup(group);
 rbBlue.setToggleGroup(group);
```

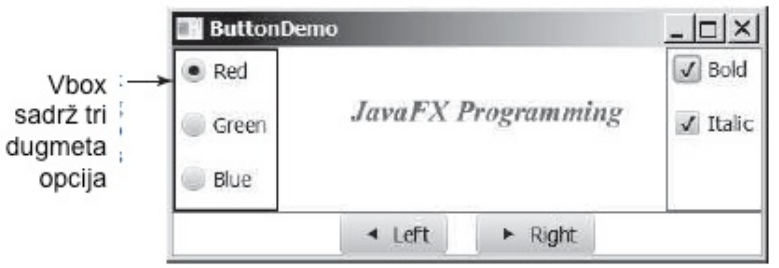
Ovaj program kreira grupu dugmadi opcija **rbRed, rbGreen**, i **rbVlue** tako da se oni mogu da se međusobno ekskluzivno biraju. Bez formiranja grupe, ova dugmad bi bila potpuno nezavisna, što znači da više od jednog mogu biti istovremeno izabrana, što u mnogim slučaja s ene može dozvoliti.

Kada se aktivira dugme opcija (ubacivanje izbora, ili izbacivanje izbora) generiše događaj tipa ***ActionEvent***. Primenom metoda ***isSelected()*** dobija se informacija da li je izabrana opcija dugmeta opcija.

.

.

Sada ćemo preći na dopunu ranijeg primera, u kome ćemo dodatitri dugmeta opcije, pod nazivima: **Red, Green**, i **Blue**. Oni omogućavaju da korisnik izabere boju teksta poruke (slika 2)



Slika 4.3 GUI prethodnih primera, sada se dopunjuju sa tri dugmeta opcija.

Da bi se napravio program koji kreira prikazani GUI, mogu se koristiti najmanje dva načina za izradu tog programa.

Prvi se zasniva na promeni ranije prikazanog programa za klasu **CheckBoxDemo .** Sada je potrebno dodati tri dugmeta opcije i onda treba obraditi ove događaje. Drugo nčin je da se definiše podklasa koja proširuje klasu **CheckBoxDemo**..

PRIMER 4 - KORIŠĆENJE KLASE RADIOBUTTON

RadiButton omogućava grupisanje opcija i za to se koristi klasa ToggleButton.

Klasa **RadioButtonDemo** proširuje **CheckBoxDemo** redefiniše metod ***getPane()*** method (linija10). Novi metod ***getPane()*** poziva metod ***getPane()*** izklase **CheckBoxDemo** da bi kreirao granično okno (border pane) koji sadrži polja potvrđivanja (check boxes), dugamad (buttons) i tekst (linija 11). Granično okno se vraća pozivanjem metoda ***super.getPane().*** Dugmad opcija (radio buttons) se kreiraju i dodaju u **paneForRadioButtons** (linije 18–21). Objekat **paneForRadioButtons** je primerak ***Vbox*** okna koji se dodaje u granično okno (linija 22).

Dugmad opcija se grupišu u liijama 24-27. Obrađivač za obradu događaja akcije dugmadi opcija kreira se u linijama 20-45. On podešava odgovarajuću boju na osnovu stanja dugmadi opcija.

Metod st***art()*** ovog JavaFX programa je definisan u klasi **ButtonDemo** i nasleđen u klasi **CheckBoxDemo** i onda u klasi **RadioButtonDemo**. Kada izvršavate program **RadioButtonDemo,** metod ***start()*** se poziva u klasi **ButtonDemo.** Kako je metod ***getPane()*** redefinisan u klasi **RadioButtonDemo**, metod u klasi RadioButtonDemo se poziva u liniji 41 u listingu klase **ButtonDemo.java**.

Klasa RadioButtonDemo:

```
import javafx.geometry.Insets;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;


public class RadioButtonDemo extends CheckBoxDemo {

    @Override // Redefinisanje metoda  getPane() u super klasi
    public BorderPane getPane() {
        BorderPane pane = super.getPane();

        VBox paneForRadioButtons = new VBox(20);
        paneForRadioButtons.setPadding(new Insets(5, 5, 5, 5));
        paneForRadioButtons.setStyle("-fx-border-color: green");
        paneForRadioButtons.setStyle("-fx-border-width: 2px; -fx-border-color:
```

```
green");
        RadioButton rbRed = new RadioButton("Red");
        RadioButton rbGreen = new RadioButton("Green");
        RadioButton rbBlue = new RadioButton("Blue");
        paneForRadioButtons.getChildren().addAll(rbRed, rbGreen, rbBlue);
        pane.setLeft(paneForRadioButtons);

        ToggleGroup group = new ToggleGroup();
        rbRed.setToggleGroup(group);
        rbBlue.setToggleGroup(group);

        rbRed.setOnAction(e -> {
            if (rbRed.isSelected()) {
                text.setFill(Color.RED);
            }
        });

        rbGreen.setOnAction(e -> {
            if (rbGreen.isSelected()) {
                text.setFill(Color.GREEN);
            }
        });

        rbBlue.setOnAction(e -> {
            if (rbBlue.isSelected()) {
                text.setFill(Color.BLUE);
            }
        });

        return pane;
    }

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

Klasa CheckBoxButtonDemo:

```
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.control.CheckBox;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.VBox;
import javafx.scene.text.Font;
import javafx.scene.text.FontPosture;
import javafx.scene.text.FontWeight;

public class CheckBoxDemo extends ButtonDemo {

    @Override // Redefinisanje metoda start() klase Application
    public BorderPane getPane() {
```

```
        BorderPane pane = super.getPane();

        Font fontBoldItalic = Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.ITALIC, 20);
        Font fontBold = Font.font("Times New Roman",
                FontWeight.BOLD, FontPosture.REGULAR, 20);
        Font fontItalic = Font.font("Times New Roman",
                FontWeight.NORMAL, FontPosture.ITALIC, 20);
        Font fontNormal = Font.font("Times New Roman",
                FontWeight.NORMAL, FontPosture.REGULAR, 20);

        text.setFont(fontNormal);

        VBox paneForCheckBoxes = new VBox(20);
        paneForCheckBoxes.setPadding(new Insets(5, 5, 5, 5));
        paneForCheckBoxes.setStyle("-fx-border-color: green");
        CheckBox chkBold = new CheckBox("Bold");
        CheckBox chkItalic = new CheckBox("Italic");
        paneForCheckBoxes.getChildren().addAll(chkBold, chkItalic);
        pane.setRight(paneForCheckBoxes);

        EventHandler<ActionEvent> handler = e -> {
            if (chkBold.isSelected() &amp;amp;amp;&amp;amp;amp; chkItalic.isSelected()) {
                text.setFont(fontBoldItalic); // Oba polja su odabrana
            } else if (chkBold.isSelected()) {
                text.setFont(fontBold); // Odabrano je polje Bold
            } else if (chkItalic.isSelected()) {
                text.setFont(fontItalic); // Odabrano je polje Italic
            } else {
                text.setFont(fontNormal); // Oba polja za potvrđivanje nisu
odabrana
            }
        };

        chkBold.setOnAction(handler);
        chkItalic.setOnAction(handler);

        return pane; // Vraća novi okvir
    }

}
```

Klasa ButtonDemo:

```
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.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Pane;
```

```
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class ButtonDemo extends Application {

    Text text = new Text(50, 50, "JavaFX Programming");

    public BorderPane getPane() {
        HBox paneForButtons = new HBox(20);
        // kreiranje dugmeta za levo. Potrebno je u projekat dodati sliku sa
nazivom left.jpg
        ImageView img_left = new ImageView("left.jpg");
        img_left.setFitHeight(50);
        img_left.setFitWidth(50);
        Button btLeft = new Button("Left", img_left);

        //kreiranje dugmeta za desno. Potrebno je u projekat dodati sliku sa
nazivom right.jpg
        ImageView img_right = new ImageView("right.jpg");
        img_right.setFitHeight(50);
        img_right.setFitWidth(50);
        Button btRight = new Button("Right", img_right);

        //dodavanje panela sa dugmucima na border layout
        paneForButtons.getChildren().addAll(btLeft, btRight);
        paneForButtons.setAlignment(Pos.CENTER);
        paneForButtons.setStyle("-fx-border-color: green");

        BorderPane pane = new BorderPane();
        pane.setBottom(paneForButtons);

        //dodavanje teksta na centar BorderPane-a
        Pane paneForText = new Pane();
        paneForText.getChildren().add(text);
        pane.setCenter(paneForText);

        //dodavanje akcija na dugmice
        btLeft.setOnAction(e -> text.setX(text.getX() - 10));
        btRight.setOnAction(e -> text.setX(text.getX() + 10));

        return pane;
    }

    @Override //redefinisanje metoda start() klase Application
    public void start(Stage primaryStage) {
        //kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(getPane(), 450, 200);
        primaryStage.setTitle("ButtonDemo"); //unos naslova pozornice
        primaryStage.setScene(scene); //postavljanje scene na scenu
        primaryStage.show(); //prikaz pozornice
    }

}
```

ZADATAK 4

Samostalno vežbanje klase RadioButton

Napravite JavaFX program po sledećim zahtevima:

- Na sceni se nalazi pet grupa po tri pitanja;
- pomoću labele formulišite pitanje za svaku grupu;
- za svaku grupu postoji samo jedan tačan odgovor;
- za tačan odgovor se dobija 3 poena, za netačan - 1;
- nakon izvršenog izbora odgovora, klikom da dugme "Kraj" se vrši obrada rezultata;
- na posebnoj labeli se prikazuje broj ostvarenih poena.

Poglavlje 5

Klasa TextField

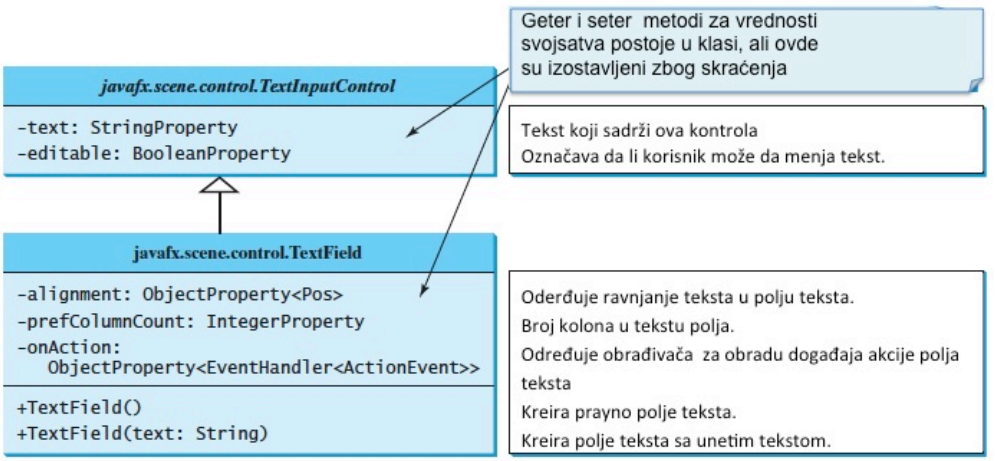
SVOJSTVA KLASE TEXTFIELD

Klasa TextField omogućava korisniku da unese ili da prikaže string, tj. skup oznaka (tekst u jednom redu).

Polje teksta, tj. objekat klase ***TextField***, omogućava unos ili prikaz nekog stringa, tj. skupa oznaka (kratkog teksta). Klasa ***TextFiled*** je podklasa klase ***TextInputControl***. Slika 1 daje prikaz svojstava i konstruktora klase ***TextField***.

Ovde se daje deo programa koji si odnosi na kreiranje jednog polja teksta prikazanog na slici 2. To je tekst koji se ne može menjati, sa crvenom bojom polja, sa određenim fontom, i sa desnim ravnjanjem. Kada pomerite kursor miša u polje teksta i pritisnete taster Enter, "ispaljuje" se događaj tipa ***ActionEvent***.

Slika 5.1 Primer polja teksta



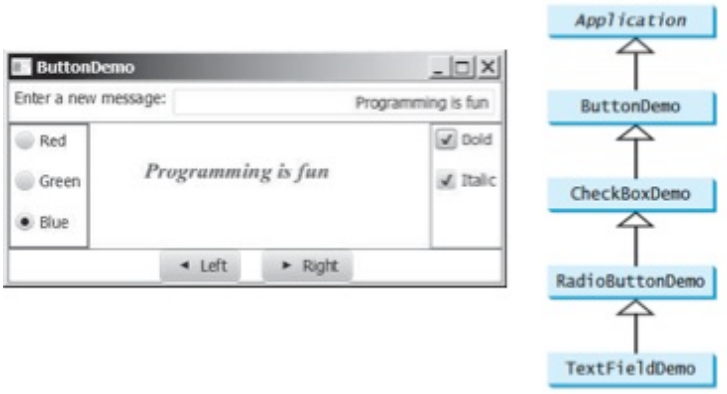
Slika 5.2 Svojstva klase TextField

```
TextField tfMessage = new TextField("T-Strom");
 tfMessage.setEditable(false);
 tfMessage.setStyle("-fx-text-fill: red");
 tfMessage.setFont(Font.font("Times", 20));
 tfMessage.setAlignment(Pos.BASELINE_RIGHT);
```

PRIMER 5 - UPOTREBA KLASE TEXTFIELD

Primer pokazuje kako se može predvideti prostor u GUI interfejsu u kome se može uneti kraći tekst i time izazvati odgovarajući događaj akcije.

Na slici 3 prikazan je GUI kreiran u prethodnim primerima



Slika 5.3 GUI s poljem teksta, kreiran i drugim klasama na putu nasleđivanja

Klasa **TextFieldDemo** proširuje **RadioButtonDemo** (linija 7) i dodaje natpis i polje teksta u kome korisnik može da unese novi tekst (linije 12-19). Posle unosa novog teksta pritiskom tastera **Enter**, prikazuje se nova poruka (linija 22). Pritiskom taster **Enter** na polju teksta, generiše se događaj akcije. Ako se želi da unese lozinka, onda se koristi klasa ***PasswordFiled*** umesto klase ***TextField***. ***PasswordField*** proširuje ***TextField*** i sakriva unetu lozinku sa oznakama **********..

```
import javafx.geometry.Insets;
import javafx.geometry.Pos;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.BorderPane;

public class TextFieldDemo extends RadioButtonDemo {

    @Override // Redefinisanje metoda getPane()super klase
    public BorderPane getPane() {
        BorderPane pane = super.getPane();

        BorderPane paneForTextField = new BorderPane();
        paneForTextField.setPadding(new Insets(5, 5, 5, 5));
        paneForTextField.setStyle("-fx-border-color: green");
        paneForTextField.setLeft(new Label("Enter a new message: "));

        TextField tf = new TextField();
        tf.setAlignment(Pos.BOTTOM_RIGHT);
        paneForTextField.setCenter(tf);
        pane.setTop(paneForTextField);
```

```
        tf.setOnAction(e -> text.setText(tf.getText()));

        return pane;
    }

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

ZADATAK 5 - SAMOSTALNO VEŽBANJE KLASE TEXTFIELD

Dodajte vaše podatke u primer iz Zadatka 4.

Napravite JavaFX program po sledećim zahtevima:

- u razvojnom okruženju kreirati klasu koja nasleđuje klasu kreiranu u Zadatku 4;
- na formi kviza dodati dva polja za unos teksta "ime" i "prezime";
- na labeli za rezultate omogućiti da se rezultat prikazuje u formi "Ime Prezime osvojili se *** poena na kvizu".

Poglavlje 6

Klasa TextArea

STRUKTURA KLASE TEXTAREA

Klasa TextArea omogućava korisniku da unese tekst u više redova.

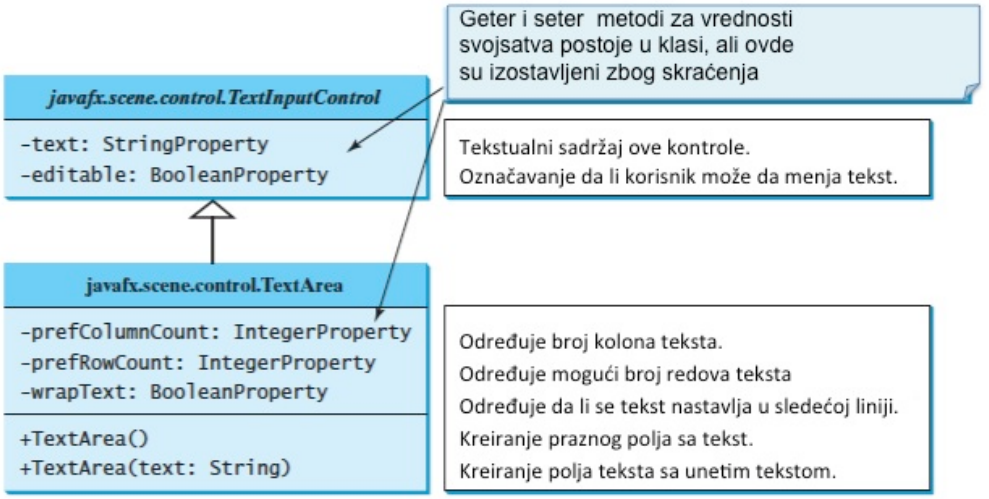
Ako želite da dozvolite da korisnik unese tekst u više redova (linija), onda to možete realizovati sa više primeraka klase ***TextField***. Bolja alternativa je da koristite klasu ***TextArea*** koja omogućava korisniku unos teksta u više redova. Slika 1 prikazuje svojstva klase ***TextArea***.

Ovde se prikazuje deo nekog programa u kome se definiše prostor za unošenje teksta koji ima 5 redova i 20 kolona, ukopljeno i sledećem redu, sa crvenonom bojom teksta i fontom ***Courier*** veličine 20 piksela.

Klasa ***TextArea*** obezbeđuje korišćenje klizača (za pomeranje teksta), ali je obično korisno da se kreira objekat ***ScrollPane*** u koji se smeštaju primerci klasa ***TextArea*** i ***ScrollPane*** rad pomeranja teksta u ***TextArea***, na sledeći način:

.

```
ScrollPane scrollPane = new ScrollPane(taNote);
```



Slika 6.1 Svojstva klase TextArea

```
TextArea taNote = new TextArea("This is a text area");
 taNote.setPrefColumnCount(20);
 taNote.setPrefRowCount(5);
```

```
taNote.setWrapText(true);
taNote.setStyle("-fx-text-fill: red");
taNote.setFont(Font.font("Times", 20));
```

DEFINISANJE KLASE DESCRIPTIONPANE

DescriptionPane klasa proširuje klasu BorderPane i predstavlja okno u koje se smešta polje teksta sa klizačima, ikona (slika) sa naslovom (ispod ikone).

Sada ćemo prikazati program koji prikazuje sliku (kanadsku zastavu), natpis sa kraćim tekstom i duži tekst u prostoru sa klizačima (slika 2).

Slika 6.2 GUI sa prikazom slike, natpisa i prostora za unos teksta sa klizačima

Postupak izrade programa je sledeći: Prvo se definiše klasa **DescriptionPane** koja proširuje klasu ***BorderPane***, a čiji se listnig ovde prikazuje. Klasa sadrži polje teksta unutar okvira sa klizačima (vertikalnim i horizontalnim) i sa natpisom za prikazivanje ikone i naslova. Klasa **DescriptionPane** će biti korišćenja i u drugim primerima.

Prostor teksta je unutar ***ScrollPane*** okna (linija 30) koji ima funkcije pomeranja teksta. Svojstvo ***wrapText*** je postavljeno na ***true*** (linija 26) te tekst ide automatski na sledeći red, posle popune celog reda. Tekst je postavljen kao nepromenljiv (linija 27).

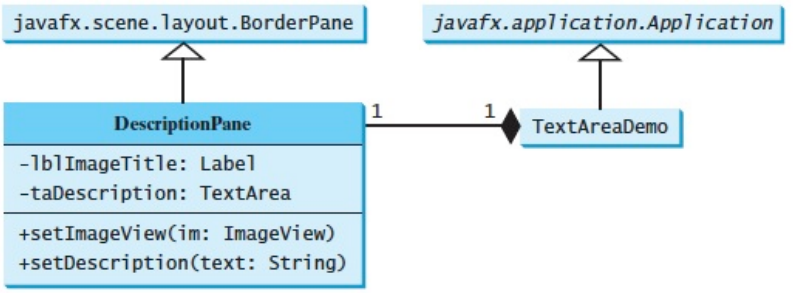
```
import javafx.geometry.Insets;
import javafx.scene.control.ContentDisplay;
import javafx.scene.control.Label;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.TextArea;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.text.Font;

public class DescriptionPane extends BorderPane {

    /**
     * Natpis za prikaz slike i naslova
     */
    private Label lblImageTitle = new Label();

    /**
     * Prostor za prikazivanje teksta
```

```
     */
    private TextArea taDescription = new TextArea();

    public DescriptionPane() {
        // Centriranje ikone i teksta i postavljenje teksta ispod ikone
        lblImageTitle.setContentDisplay(ContentDisplay.TOP);
        lblImageTitle.setPrefSize(200, 100);

        // Unos fonta natpisa i  polja za tekst
        lblImageTitle.setFont(new Font("SansSerif", 16));
        taDescription.setFont(new Font("Serif", 14));

        taDescription.setWrapText(true);
        taDescription.setEditable(false);

        // Kreiranje okna sa klizačima za prikaz polja teksta
        ScrollPane scrollPane = new ScrollPane(taDescription);

        // Postavljanje natpisa i okna sa klizačima u granično okno
        setLeft(lblImageTitle);
        setCenter(scrollPane);
        setPadding(new Insets(5, 5, 5, 5));
    }

    /**
     * Unos naslova
     */
    public void setTitle(String title) {
        lblImageTitle.setText(title);
    }

    /**
     * Unos pogleda na sliku
     */
    public void setImageView(ImageView icon) {
        lblImageTitle.setGraphic(icon);
    }

    /**
     * Unos opisa teksta
     */
    public void setDescription(String text) {
        taDescription.setText(text);
    }
}
```

PRIMER 6 - KORIŠĆENJE PROSTORA ZA TEKST SA KLIZAČIMA

Primer kreira GUI sa slikom (zastava), natpisom (kraki tekst) i sa prostorm za tekst sa klizačima, u koji se unosi duži tekst (opis zastave).

Definiše se klasa **TextAreaDem**o koja proširuje klasu ***Application***, kao što je prikazano na listingu. Tu se kreira primerak klase **DescriptionPane** i dodaje se u scenu. Međusobna veza **DescriptionPane** i **TextAreaDemo** klasa prikazana je na slici 3.



Slika 6.3 upotrebljava DescriptionPane za prikaz ikone (zastave), naslova (naziv zemlje) i tekstualnog opisa zastave

Program kreira primerak klase **DescriptionPane** (linija 10) i stavlja naslov (linija 13), sliku (linija 15) i tekst u opisno okno (linija 16). Klasa **DescriptionPane** je podklasa klase ***Pane***. **DescriptionPane** sadrži natpis za prikaz slike i naslova, kao i prostor teksta u koji se upisuje tekstualni opis zastave.

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.image.ImageView;
import javafx.stage.Stage;


public class TextAreaDemo extends Application {

    @Override // Redefinisanje metoda start() iz klase Application
    public void start(Stage primaryStage) {
        // Objavljivanje i kreiranje okna za opisivanje
        DescriptionPane descriptionPane = new DescriptionPane();

        // Unos naslova, teksta i slike u okvir za opisivanje
        descriptionPane.setTitle("Canada");
        String description = "The Canadian national flag ...";
        ImageView img_view = new ImageView("ca.jpg");
        img_view.setFitHeight(150);
        img_view.setFitWidth(200);
        descriptionPane.setImageView(img_view);
        descriptionPane.setDescription(description);
```

```
        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(descriptionPane, 450, 200);
        primaryStage.setTitle("TextAreaDemo"); // Unos naziva pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

}
```

ZADATAK 6 - SAMOSTALNO VEŽBANJE KLASE TEXTAREA

Dodajte vaše podatke u primer iz Zadatka 5.

Napravite JavaFX program po sledećim zahtevima:

- u razvojnom okruženju kreirati klasu koja nasleđuje klasu kreiranu u Zadatku 5;
- na formi kviza dodati oblast za prikazivanje teksta na mesto po želji;
- prilikom izbora određenog odgovora, iz neke grupe pitanja, u kreiranoj oblasti za prikazivanje teksta prikazuje se string u formi "Odgovor X: dodaje se tekst koji odgovara izabranom odgovoru".

Poglavlje 7

Klasa ComboBox

SVOJSTVA KLASE COMBOBOX

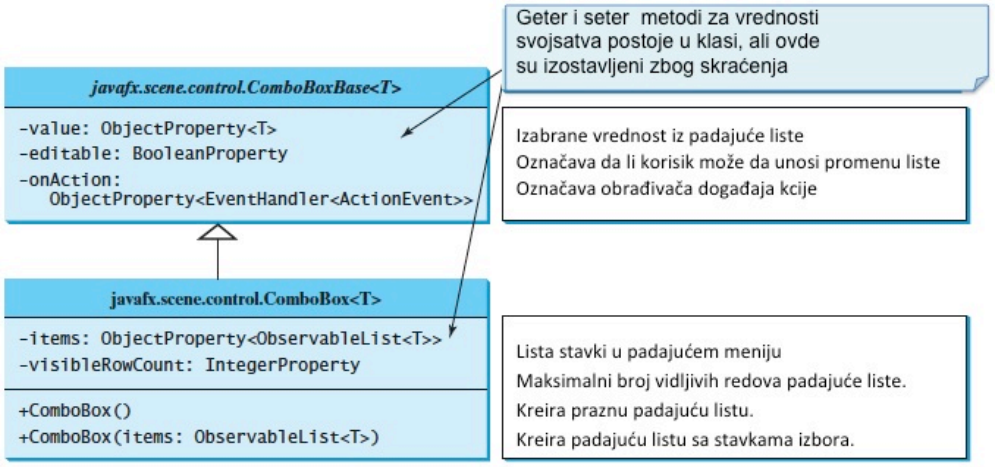
Klasa ComboBox obezbeđuje tzv. padajuću listu sa opcijama koje korisnik može da izabere. .

Padajuća lista je korisna jer korisniku ograničava izbor opcija i eliminiše napornu proveru tačnosti unosa podataka. Slika 1 prikazuje najčešće korišćena svojstva klase ***ComboBox*** . Ona je definisana kao generička klasa. Tip T određuje tip elemenata koje sadrži padajuća lista.

Daje se deo nekog programa koji kreira padajuću listu sa četiri stavke, sa crvenom bojom, i postavljenom vrednošsti za prvu stavku (slika 2).

Slika 7.1 Primer padajuće liste

Klasa ***ComboBox*** nasleđuje klasu ***ComboBoxBase***. ***ComboBox*** ispaljuje akcioni događj tipa ***ActionEvent***..



Slika 7.2 Najvažnija svojstva klase ComboBox

```
ComboBox<String> cbo = new ComboBox<>();
 cbo.getItems().addAll("Item 1", "Item 2",
    "Item 3", "Item 4");
```

```
cbo.setStyle("-fx-color: red");
cbo.setValue("Item 1");
```

ObservableList je podinterfejs interfejsa ***java.util.List***. Možete koristiti metode tog interfejsa akođe, možete koristiti *FXCollections.observableArrayList(arrayOfElements)* za kreiranje ***ObesrvableList*** sa nizom elemenata. .

PRIMER 7 PRIMENA KLASE COMBOBOX

Program koji dozvoljava korisniku da vidi sliku zastave neke zemlje i njen tekstualni opis, po izboru zemlje iz padajućeg menija od strane korisnika

Listing prikazuje program koji dozvoljava korisniku da vidi sliku zastave neke zemlje i njen tekstualni opis, po izboru zemlje iz padajućeg menija od strane korisnika (slika 3).

Slika 7.3 GUI sa prikazom slike zastave, nazivom zemlje i opisom zastave, a izabrane iz padajućeg menija

Kreiranje GUI: Najpre se kreira padajuća lista sa nazivima zemalja. Kreira se DescriptionPane objekat, a padajuća lista se stavlja na vrh graničnog okna a opisno okno se stavlja u sredinu graničnog okna.

Obrada događaja: Kreiranje obrađivača koji obrađuja ispaljen akcioni događaj po izboru stavke sa padajućeg menija.

Program koristi tri niza: flagTitles, flagImage, i flagDescription (linije 13–28).za smeštanje naslova, slika zastava i opisa zastava.

Program kreira primerak klase **DescriptionPane** (linja 31) i kreira padajući meni sa listom zemalja **(flagTittles**) 62–63). Metod ***getItems()*** vraća listu stavki sa padajuće liste (linija 64) a metod ***addAll()*** dodaje više stavka u padajuću listu. Kada korisnik izabere stavku sa padajuće liste, ispaljuje se akcioni događaj koji aktivira odgovarajući obraživač događaja koji nalazi odgovarajući indeks (linija 68) i poziva metod ***setDisplay(int index)*** radi postavljanja izabrane zastave, naslova i opisa u okvir (linije 78-82)

.

```
import javafx.application.Application;
import javafx.collections.FXCollections;
```

```
import javafx.collections.ObservableList;
import javafx.scene.Scene;
import javafx.scene.control.ComboBox;
import javafx.scene.control.Label;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.stage.Stage;
import jdk.nashorn.internal.ir.Flags;

public class ComboBoxDemo extends Application {

    // Deklaracija niza sa String objektima za naslove zastava
    private String[] flagTitles = {"Canada", "China", "Denmark",
        "France", "Germany", "India", "Norway", "United Kingdom",
        "United States of America"};

    // Deklaracija niza objekata ImageView sa zastavama 9 zemalja.
    private ImageView[] flagImage = {new ImageView("image/ca.jpg"),
        new ImageView("image/china.jpg"),
        new ImageView("image/denmark.jpg"),
        new ImageView("image/fr.jpg"),
        new ImageView("image/germany.jpg"),
        new ImageView("image/india.jpg"),
        new ImageView("image/norway.jpg"),
        new ImageView("image/uk.jpg"),
        new ImageView("image/us.jpg")};

    // Deklaracija niza sa String objektima sa tekstom opisa zastava
    private String[] flagDescription = new String[9];

    // Deklaracija i kreiranja opisnog okna
    private DescriptionPane descriptionPane = new DescriptionPane();

    // Kreiranje padajužeg menija za izabrane zemlje
    private ComboBox<String> cbo = new ComboBox<>(); // nazivi zastava;

    @Override // Redefinisanje metoda start() klase Application class
    public void start(Stage primaryStage) {
        // Unos tekstova sa opisom
        flagDescription[0] = "The Canadian national flag ...";
        flagDescription[1] = "Description for China ... ";
        flagDescription[2] = "Description for Denmark ... ";
        flagDescription[3] = "Description for France ... ";
        flagDescription[4] = "Description for Germany ... ";
        flagDescription[5] = "Description for India ... ";
        flagDescription[6] = "Description for Norway ... ";
        flagDescription[7] = "Description for UK ... ";
        flagDescription[8] = "Description for US ... ";

        for (int i = 0; i < flagImage.length; i++) {
            flagImage[i].setFitHeight(150);
            flagImage[i].setFitWidth(200);
        }
```

```
        //  Unos prve zemlje (Canada) za prikaz
        setDisplay(0);

        // Dodaj padajući meni i opisno okno u granično okno
        BorderPane pane = new BorderPane();

        BorderPane paneForComboBox = new BorderPane();
        paneForComboBox.setLeft(new Label("Select a country: "));
        paneForComboBox.setCenter(cbo);
        pane.setTop(paneForComboBox);
        cbo.setPrefWidth(400);
        cbo.setValue("Canada");

        ObservableList<String> items
                = FXCollections.observableArrayList(flagTitles);
        cbo.getItems().addAll(items);
        pane.setCenter(descriptionPane);

        // Prikaz izabrane zemlje
        cbo.setOnAction(e -> setDisplay(items.indexOf(cbo.getValue())));

        // Kreiranje scene i njeno  postavljanje na pozornicu
        Scene scene = new Scene(pane, 450, 250);
        primaryStage.setTitle("ComboBoxDemo"); // Unos naslova pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

    /**
     * Postavljanje prikaza informacija na opisno okno
     */
    public void setDisplay(int index) {
        descriptionPane.setTitle(flagTitles[index]);
        descriptionPane.setImageView(flagImage[index]);
        descriptionPane.setDescription(flagDescription[index]);
    }

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

}
```

ZADATAK 7- SAMOSTALNO VEŽBANJE KLASE COMBOBOX

Dodajte vaše podatke u primer iz Zadatka 5.

Napravite JavaFX program po sledećim zahtevima:

- u razvojnom okruženju kreirati klasu koja nasleđuje klasu kreiranu u Zadatku 6;
- dodati ComboBox kontrolu koja sadrži listu sa učesnicima kviza;
- podatke o izabranom učesniku, iz ComboBox kontrole, dodati u dva postojeća polja za unos teksta "ime" i "prezime";
- na labeli za rezultate omogućiti da se rezultat prikazuje u formi "Ime Prezime osvojili se *** poena na kvizu".

VIDEO - CHOICEBOX

JavaFX Java GUI Tutorial - 12 - ChoiceBox (Drop Down Menu) (8,10 minuta)

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

VIDEO - IZBOR PROMENE

JavaFX Java GUI Tutorial - 13 - Listening for Selection Changes (6,12 minuta)

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

VIDEO - COMBOBOX

JavaFX Java GUI Tutorial - 14 - ComboBox (8,05 minuta)

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

Poglavlje 8

Klasa ListView

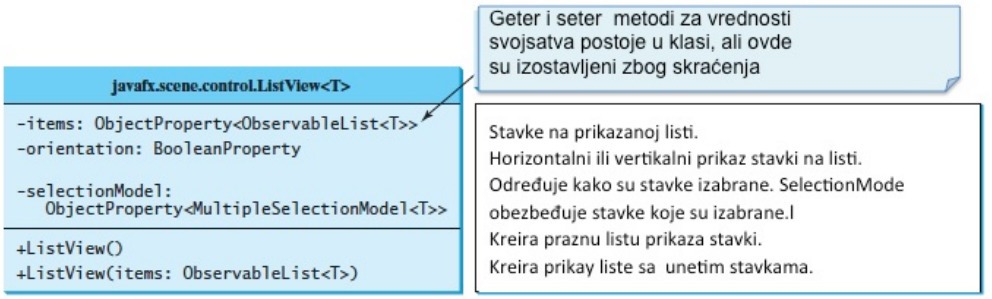
SVOJSTVA KLASE LISTVIEW

Klasa ListView daje prikaz liste, ko što daje i ComboBox klasa, s tom razlikom što korisnik, pored jedne ponuđene stavke, može da izabere i više ponuđenih stavki.

Prikaz liste, tj. objekat klase ***ListView***, daje listu ponuđenih opcija, od koji korisnik može d aizabere jednu, ali i više njih. Slika 2 prikazuje najšešće korišćena svojstva klase ***ListView*** i njene konstruktore.

Klasa ListView je definisana kao generička klasa. Generički (opšti) tip T označava tip elemenata prikazanih u listi prikaza.

.

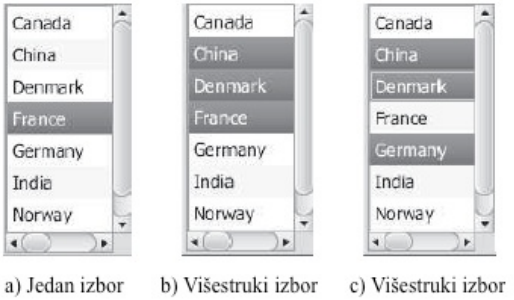


Slika 8.1 Najvažnija svojstva klase ListView

IZBOR JEDNE ILI VIŠE STAVKI SA LISTE

Način izbora se definiše preko jedne od dve moguće konstante: SelectionMode.SINGLE ili SelectionMode.MULTIPLE koje ukazuje da li se bira jedna stavka ili se biraju više stavki

Metod ***getSelectionModel()*** vraća primerak klase ***SelectionModel*** , koji sadrži metode za postavljanje načina izbora i dobijanje izabranih indeksa i stavka. Način izbora se definiše preko jedne od dve moguće konstante: ***SelectionMode.SINGLE*** ili ***SelectionMode.MULTIPLE*** koje ukazuje da li se bita jedna stavka ili se biraju više stavki. Početna vrednost je ***SelectionMode.SINGLE***. Slika 2.a prikazuje jedan izbor, a slike 2.b i 2.c sa višestrukim izborom. Dole je dat kod kreiranja prilaza liste sa stavka,a Item 1 – Item 6.



Slika 8.2 Biranje jedne ili više stavki

```
ObservableList<String> items =
   FXCollections.observableArrayList("Item 1", "Item 2",
   "Item 3", "Item 4", "Item 5", "Item 6");
 ListView<String> lv = new ListView<>(items);
 lv.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);
```

Model izbora stavki sa prikazane liste sadrži svojstvo ***selectedItemProperty***, koja je primerak od ***Obeservable*** klase. Ovde se mogu pridorati osluškivač radi praćenja promene svojstava, na sledeći način:

```
lv.getSelectionModel().selectedItemProperty().addListener(
   new InvalidationListener() {
    public void invalidated(Observable ov) {
      System.out.println("Selected indices: "
        + lv.getSelectionModel().getSelectedIndices());
      System.out.println("Selected items: "
        + lv.getSelectionModel().getSelectedItems());
    }
 });
```

Anonimna unutrašnja klasa se može uprostiti primenom lambda izraza na sledeći način:

```
lv.getSelectionModel().selectedItemProperty().addListener(ov -> {
   System.out.println("Selected indices: "
    + lv.getSelectionModel().getSelectedIndices());
   System.out.println("Selected items: "
    + lv.getSelectionModel().getSelectedItems());
 });
```

PRIMER 8.1. - NAČIN PRIPREME PROGRAMA PRIMENOM LISTVIEW KLASE

Prvo se kreira GUI interefejs, primenom svih njegovih elemenata, a onda se kreiraju osluškivači, tj. obrađivači događaja koji nastaju pri interakciji korisnika sa GUI interfejsom.

Listing klase ***ListViewDemo*** daje program koji dozvoljava izbor država u prikazanoj listi i njen prikaz zastava izabranih zemalja u prikazu na slici 3.

Slika 8.3 GUI sa prikazom liste naziva zastava i prikaz odabrane zastave zajedno sa opisom

Program se razvija u sledećim koracima:

1. **Kreiranje GUI interfejsa:** Kreira se prikaz liste sa nazivima devet zemalja kao vrednosti za izbor i postavljanje prikaza liste unutar prostora sa klizačima. Postavljanje okvira sa klizačima na levoj strani graničnog okvira , Kreiraju se pogledi sa devet slika zastava . Kreira se tekuće okno (flow pane) za smeštaj prikaza slika i postavljanje u centar ograničenog okna.

.

1. **Obrada događaja:** Kreiranje osluškivača .za primenu metoda poništenj u iterfejsu ***invalidtionLIstener*** da bi se postavila slika zastaea izabrane zemlje u oknu.

PRIMER 8.1. - LISTING KLASE LISTVIEWDEMO

Program kreira niz stringova naziva zemalja i niz sa devet slika zastava, devet zemalja u istom redosledu kao i u nizu sa nazivima zemalja

Program kreira niz stringova naziva zemalja (linije 14-16) i niz sa devet slika zastava, devet zemalja. (linije 19-29) u istom redosledu kao i u nizu sa nazivima zemalja. Stavke u prizanoj listi su preuzete iz niza sa nazivim azemalja (linija 34).

Prikazana lista je postavljena u okno sa klizačima (linija 41) tako da se može lista pomerati uz pomoć klizača, kada je broj stavki veća od prossora za prikaz liste.

Početno je podešeno da je način izbora stavki sa liste pojedinačno. Međutim, u način izbora ovde u programu je podešeno da bude višestruki (linija 36), što dozvoljava korisniku da izabere više stavki sa liste. Kada korisnik izabere zemlje sa liste, obrađivač događaja se izvršava (linije 44-50), čime dobija indekse izabranih stavki (zemalja(i onda dodaje odgovarajuće slike u okno.

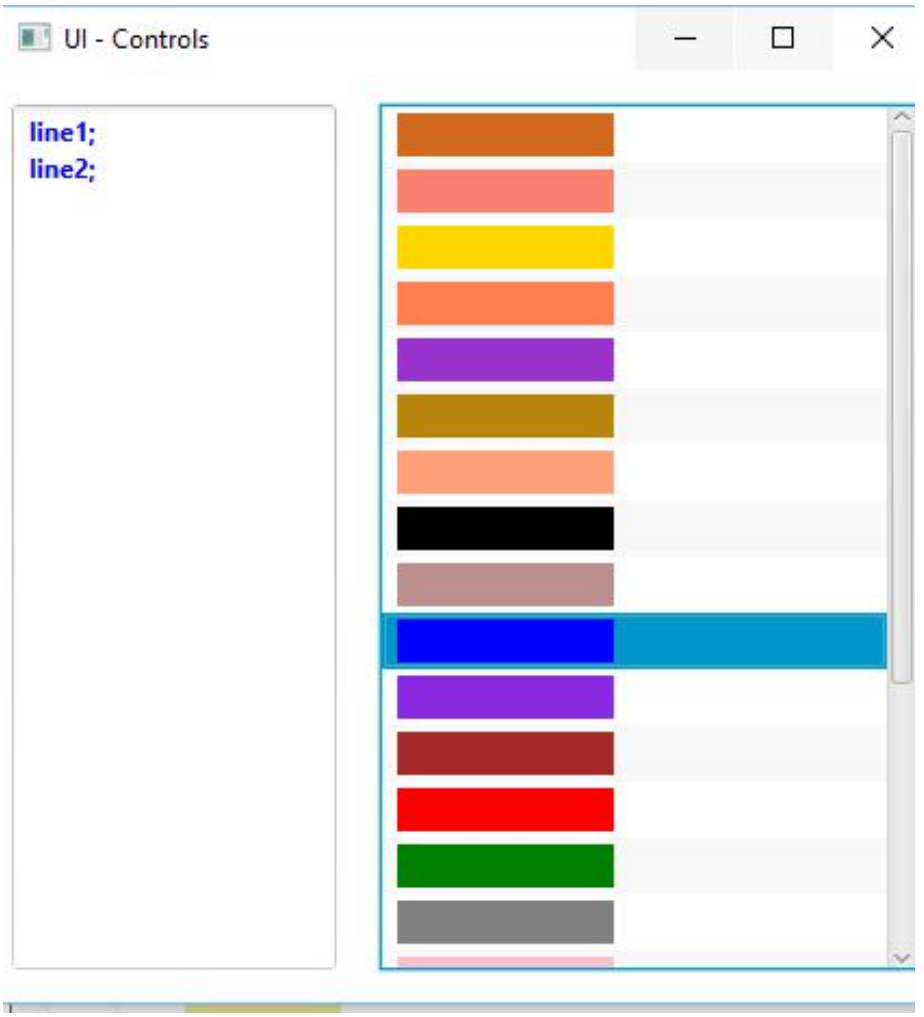
```
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.scene.Scene;
import javafx.scene.control.ListView;
import javafx.scene.control.ScrollPane;
import javafx.scene.control.SelectionMode;
import javafx.scene.image.ImageView;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class ListViewDemo extends Application {

    // Deklaracija niza objekata String sa nazivima zastava s
    private String[] flagTitles = {"Canada", "China", "Denmark",
        "France", "Germany", "India", "Norway", "United Kingdom",
        "United States of America"};

    // Deklaracija niza sa ImageView zastava 9 zemalja.
    private ImageView[] imageViews = {
        new ImageView("image/ca.jpg"),
        new ImageView("image/china.jpg"),
        new ImageView("image/denmark.jpg"),
        new ImageView("image/fr.jpg"),
        new ImageView("image/germany.jpg"),
        new ImageView("image/india.jpg"),
        new ImageView("image/norway.jpg"),
        new ImageView("image/uk.jpg"),
        new ImageView("image/us.jpg")
    };

    @Override // Redefinisanje metoda start() klase Application
    public void start(Stage primaryStage) {
        ListView<String> lv = new
ListView(FXCollections.observableArrayList(flagTitles));
        lv.setPrefSize(400, 400);
        lv.getSelectionModel().setSelectionMode(SelectionMode.MULTIPLE);

        for (int i = 0; i < imageViews.length; i++) {
            imageViews[i].setFitHeight(400);
            imageViews[i].setFitWidth(500);
        }

        // Kreiranje okvir za držanje slika zastava
        FlowPane imagePane = new FlowPane(10, 10);
        BorderPane pane = new BorderPane();
        pane.setLeft(new ScrollPane(lv));
```

```
        pane.setCenter(imagePane);

        lv.getSelectionModel().selectedItemProperty().addListener(
                ov -> {
                    imagePane.getChildren().clear();
                    for (Integer i : lv.getSelectionModel().getSelectedIndices()) {
                        imagePane.getChildren().add(imageViews[i]);
                    }
                });

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane, 900, 400);
        primaryStage.setTitle("ListViewDemo"); // Unos naslova pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozornice
    }

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

}
```

PRIMER 8.2

Cilj zadatka je da se demonstrira rad sa klasama ListView i TextArea

Napraviti program u kom imamo tekst editor koji je objekat klase TextArea i pored njega objekat ListView koji će se koristiti da se unutar njega odabere boja font-a za editor.



Slika 8.4 Zadatak 2

PRIMER 8.2- OBJAŠNJENJE I UPUTSTVO ZA REŠAVANJE

U nastavku je dato pojašnjenje rešenja zadatka 2

Objašnjenje i uputstva:

1. Kreirati JavaFX aplikaciju. Pokretačku klasu nazvati TextAreaDemo.java. U okviru ove klase kreirati GridPane sa komponentama TextArea i komponentom, objektom klase ColorChooser.java koja nasleđuje VBox.

2. Klasa ColorChooser sadrzi atribut TextArea i ListView kome su dodeljeni elementi liste boja:
ObservableList<String> data = FXCollections.observableArrayList(
"chocolate", "salmon", "gold", "coral", "darkorchid"...}

3. Kako ćelije ListView-a nisu tekst već pravougaonici ispunjeni bojom potrebno je kreirati unutrašnju klasu koja će predstavljati jednu ćeliju (ColorRectCell) komponente ListView. Ova klasa nasleđuje klasu ListCell<String> i redefiniše metodu updateItem:

```
@Override
public void updateItem(String item, boolean empty) {
super.updateItem(item, empty);
Rectangle rect = new Rectangle(100, 20);
if (item != null) {
rect.setFill(Color.web(item));
setGraphic(rect);
}
}
```

4. Nakon definisanja ćelije, potrebno je postaviti da svaka ćelija bude objekat unutrašnje klase ColorRectCell:

```
list.setCellFactory(new Callback<ListView<String>, ListCell<String>>() {
@Override
public ListCell<String> call(ListView<String> list) {
return new ColorRectCell();
}
}
);
```

5. Dodati listener koji će detektovati promenu selekcije ćelije u okviru komponente ListView i koji ima zadatak da tekstu u okviru TextArea komponente promeni boju na onu koja je selektovana:

```
list.getSelectionModel().selectedItemProperty().addListener(new ChangeListener<String>()
{
@Override
public void changed(ObservableValue<? extends String> ov, String old_val, String new_val)
{
editor.setStyle("-fx-text-fill: " + new_val + "; "
+ "-fx-font-weight: bold;");
}
});
```

PRIMER 8.2. - REŠENJE

Programiski kod koji predstvalja rešenje zadatka 2

ColorChooser Klasa:

```
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.control.ListCell;
import javafx.scene.control.ListView;
import javafx.scene.control.TextArea;
import javafx.scene.layout.VBox;
import javafx.scene.paint.Color;
```

```
import javafx.scene.shape.Rectangle;
import javafx.util.Callback;

public class ColorChooser extends VBox {

    TextArea editor;
    ListView<String> list = new ListView<String>();
    ObservableList<String> data = FXCollections.observableArrayList(
            "chocolate", "salmon", "gold", "coral", "darkorchid",
            "darkgoldenrod", "lightsalmon", "black", "rosybrown", "blue",
            "blueviolet", "brown", "red", "green", "gray", "pink", "orange",
            "skyblue", "darkgray", "plum", "indigo", "lime");

    public ColorChooser(final TextArea editor) {
        this.editor = editor;
        list.setItems(data);

        // postavlja se da svaka celija bude objekat unutrasnje klase ColorRectCell
        list.setCellFactory(new Callback<ListView<String>, ListCell<String>>() {
            @Override
            public ListCell<String> call(ListView<String> list) {
                return new ColorRectCell();
            }
        }
        );

        list.getSelectionModel().selectedItemProperty().addListener(new
ChangeListener<String>() {
            @Override
            public void changed(ObservableValue<? extends String> ov, String
old_val, String new_val) {
                editor.setStyle("-fx-text-fill: " + new_val + "; "
                        + "-fx-font-weight: bold;");
            }
        });

        getChildren().add(list);
    }

    // kreiranje unutrasnje klase koja predstavlja jednu celiju unutar ListView
objekta
    private class ColorRectCell extends ListCell<String> {

        @Override
        public void updateItem(String item, boolean empty) {
            super.updateItem(item, empty);
            Rectangle rect = new Rectangle(100, 20);
            if (item != null) {
                rect.setFill(Color.web(item));
                setGraphic(rect);
            }
        }
    }
```

```
}
```

TextAreaDemo Klasa:

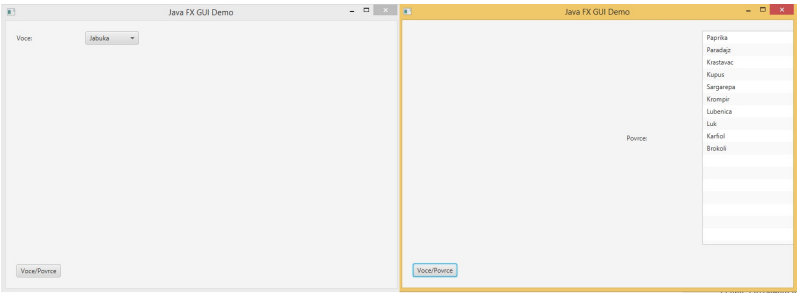
```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.TextArea;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


public class TextAreaDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        Group root = new Group();
        Scene scene = new Scene(root, 430, 430, Color.WHITE);

        GridPane gridpane = new GridPane();
        gridpane.setPadding(new Insets(5));
        gridpane.setHgap(10);
        gridpane.setVgap(10);

        final TextArea editorFld = new TextArea();
        ColorChooser colorChr = new ColorChooser(editorFld);
        editorFld.setPrefRowCount(10);
        editorFld.setPrefColumnCount(100);
        editorFld.setWrapText(true);
        editorFld.setPrefWidth(150);
        GridPane.setHalignment(editorFld, HPos.CENTER);
        gridpane.add(editorFld, 0, 1);
        gridpane.add(colorChr, 2, 1);

        editorFld.setStyle("-fx-text-fill: black;"
                + "-fx-font-weight: bold;");
        String cssDefault = "line1;\nline2;\n";
        editorFld.setText(cssDefault);

        root.getChildren().add(gridpane);
        primaryStage.setTitle("UI - Controls");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

```
import javafx.application.Application;
import javafx.geometry.HPos;
import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.TextArea;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.stage.Stage;


public class TextAreaDemo extends Application {

    @Override
    public void start(Stage primaryStage) {
        Group root = new Group();
        Scene scene = new Scene(root, 430, 430, Color.WHITE);

        GridPane gridpane = new GridPane();
        gridpane.setPadding(new Insets(5));
        gridpane.setHgap(10);
        gridpane.setVgap(10);

        final TextArea editorFld = new TextArea();
        ColorChooser colorChr = new ColorChooser(editorFld);
        editorFld.setPrefRowCount(10);
        editorFld.setPrefColumnCount(100);
        editorFld.setWrapText(true);
        editorFld.setPrefWidth(150);
        GridPane.setHalignment(editorFld, HPos.CENTER);
        gridpane.add(editorFld, 0, 1);
        gridpane.add(colorChr, 2, 1);

        editorFld.setStyle("-fx-text-fill: black;"
                + "-fx-font-weight: bold;");
        String cssDefault = "line1;\nline2;\n";
        editorFld.setText(cssDefault);

        root.getChildren().add(gridpane);
        primaryStage.setTitle("UI - Controls");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

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

PRIMER 8.3

Cilj ovog zadatka je pokaz primene ChoiceBox-a i ListView-a u JaviFX kao i sakrivanja istih na događaje

Napraviti ChoiceBox za voće kao i ListView za povrće. Napuniti ListView i ChoiceBox sa imenima voća i povrća. Napraviti dugme koje će menjati prikaz (jednom će prikazati ChoiceBox a drugi put ListView)



Slika-5: Zadatak 5

PRIMER 8.3- OBJAŠNJENJE I REŠENJE

U nastavku je dato pojašnjenje rešenja zadatka 5

Ovaj zadatak demonstrira primenu komponenti ChoiceBox i ListView u JaviFX kao i sakrivanja istih na događaje. Klikom na dugme Voce/Povrce vrši se promena vidljivostii trenutno prikazanog/skrivenog objekta:

```
toggle.setOnAction(new EventHandler<ActionEvent>() {
@Override
public void handle(ActionEvent t) {
choicePane.setVisible(!choicePane.isVisible());
listPane.setVisible(!listPane.isVisible());
}
});
```

Klasa JFXTest:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.collections.FXCollections;
import javafx.event.ActionEvent;
import javafx.event.EventHandler;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.Button;
```

```
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.control.Tooltip;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.FlowPane;
import javafx.stage.Stage;

public class JFXTest extends Application {

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

    @Override
    public void start(Stage primaryStage) throws Exception {
        BorderPane root = new BorderPane();
        root.setPadding(new Insets(20, 50, 20, 20));

        final FlowPane choicePane = new FlowPane();
        choicePane.setHgap(100);

        Label choiceLbl = new Label("Voce:");

        final ChoiceBox fruits = new
ChoiceBox(FXCollections.observableArrayList("Jabuka", "Kruska", "Banana",
"Pomorandza", "Mandarina", "Limun", "Kivi", "Grejpfrut", "Ananas", "Grozdje",
"Breskva", "Visnja"));
        fruits.setValue("Jabuka");
        fruits.setTooltip(new Tooltip("Izaberi voce"));
        choicePane.getChildren().add(choiceLbl);
        choicePane.getChildren().add(fruits);

        root.setLeft(choicePane);

        final FlowPane listPane = new FlowPane();
        listPane.setHgap(100);

        Label listLbl = new Label("Povrce: ");
        ListView vegetables = new
ListView(FXCollections.observableArrayList("Paprika", "Paradajz", "Krastavac",
"Kupus", "Sargarepa", "Krompir", "Lubenica", "Luk", "Karfiol", "Brokoli"));

        listPane.getChildren().addAll(listLbl, vegetables);
        root.setRight(listPane);

        choicePane.setVisible(true);
        listPane.setVisible(false);

        Button toggle = new Button("Voce/Povrce");
```

```
        //dodajemo listener na dugem
        toggle.setOnAction(new EventHandler<ActionEvent>() {
            @Override
            public void handle(ActionEvent t) {
                choicePane.setVisible(!choicePane.isVisible());
                listPane.setVisible(!listPane.isVisible());
            }
        });
        // dodajemo listener na checkbox koji reaguje na promenu item-a checkbox-u
        fruits.getSelectionModel().selectedIndexProperty().addListener(new
ChangeListener<Number>() {

            @Override
            public void changed(ObservableValue<? extends Number> ov, Number t,
Number t1) {
                System.out.println(fruits.getItems().get(t1.intValue()));
            }

        });

        vegetables.getSelectionModel().selectedItemProperty().addListener(new
ChangeListener() {

            @Override
            public void changed(ObservableValue ov, Object t, Object t1) {
                System.out.println(t1);
            }
        });

        root.setBottom(toggle);

        Scene scene = new Scene(root, 700, 500);

        primaryStage.setTitle("Java FX GUI Demo");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

}
```

ZADATAK 8 - SAMOSTALNO VEŽBANJE KLASE LISTVIEW

Dodajte vaše podatke u primer iz Zadatka 5 listom umesto ComboBox kontrolom.

Napravite JavaFX program po sledećim zahtevima:

- u razvojnom okruženju kreirati klasu koja nasleđuje klasu kreiranu u Zadatku 6;
- dodati ListView kontrolu koja sadrži listu sa učesnicima kviza;

- moguće je izabrati samo jednu stavku iz liste;
- podatke o izabranom učesniku, iz ListView kontrole, dodati u dva postojeća polja za unos teksta "ime" i "prezime";
- na labeli za rezultate omogućiti da se rezultat prikazuje u formi "Ime Prezime osvojili se *** poena na kvizu".

VIDEO - LISTVIEW

JavaFX Java GUI Tutorial - 15 - ListView (7,37 minuta)

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

Poglavlje 9

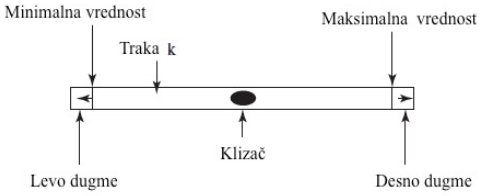
Klasa ScrollBar

SVOJSTVA KLASE SCROLLBAR

ScrollBar je kontrola koja omogućava korisniku da izabere neku vrednost u okviru navedenog opsega mogućih vrednosti.

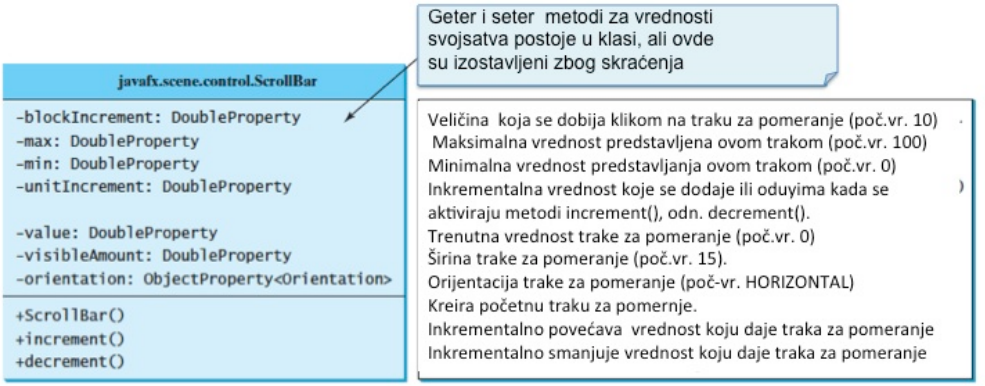
Slika 1 pokazuje traku za pomeranje (vrednosti), tj. GUI komponentu ***ScrollBar***. Korisnik menja vrednost na traci za pomeranje koristeći miš. On to radi ne jedan od sledećih načina:

- Postavi miš preko klizača trake, pritsne dugme miša, i pomer amiš duž trake.
- Klikne dugemtom miša koji je postavljen preko trake.
- Klikne da levo ili desno dugme trake.



Slika 9.1 Traka za pomeranje predstavlja grafički prikazan opseg vrednosti

Slika 2 prikazuje svojstva klase ScrollBar



Slika 9.2 Svojstva klase ScrollBar

Kada korisnik promeni vrednost na traci za pomeranje (scroll bar), automatski se obaveštava osluškivač događaja o ovoj promeni. Osluškivač treba da registrujete kod ScrollBar objekta primenom koda:

```
ScrollBar sb = new ScrollBar();
 sb.valueProperty().addListener(ov -> {
   System.out.println("old value: " + oldVal);
   System.out.println("new value: " + newVal);
 });
```

PRIMER 9 - PRIMENA KLASE SCROLLBAR

Primena traka za pomeranje (objekta ScrollBar) se vrši tako što se prvo kreira korisnčki interfejs sa njima, a onda se kreiraju osluškivači koji prate i generišu događaje pri promeni vre

Slika 3 prikazuje GUI interfejs sa vertikalnom i horizontalnom trakom za pomeranje. Pomeranjem klizača horizontalne trake za pomeranje, tekst “JavaFX Programming” se pomera u smeru pomeranja klizača. Vertikalnim pomeranjem klizača vertikalne trake za pomeranje, vrši se pomeranje prikazanog teksta u smeru pomeranja klizaču u vertikalnom pravcu. Listing klase ScrollBarrDemo predstavlja program koji ovo omogućava.

Slika 9.3 Trake za pomeranje vrše horizontalno i vertikalno pomeranje teksta

Problem se rešava u dve faze.

1. **Kreiranje korisnikog interfejsa**: Kreira se objekat Text i postavlja se u centar graničnog okna. Kreira se vertikalna traka za pomeranje i postavlja se na desnu stranu graničnog okna. Kreira se horizontalna trak aza pomeranje i postavlja se na donji deo graničnog okna.
2. **Obrada događaja**: Kreiranje osluškivača događaja pomeranja teksta pomoću traka za pomeranje. Svaka promena vrednosti na njima, vrš generisanje odgovarajućih događaja akcije i, u skladu s njima, pomeranje teksta.

.

PRIMER 9 - LISTING KLASE SCROLLBOXDEMO

Pomeranjem klizača na trakama, menja se x i y vrednost početka teksta, te se tekst pomera.

Program kreira tekst (linija 13) i dve trake za pomeranje (scroll bars), tj. sbHorizontal i sbVertical (linije 15-16). Tekst se postavlja u okno (linija 21), koje se onda postavlja u sredinu graničnog okna (linija 25). Horizontalna i vertikalna traka sbHorizontal i sbVertical se postavljaju dole i desno u odnosu na centar graničnog okna (linije 26-27).

Osluškivač se registruje sa svojstvo vrednosti trake sbHorizontal (linije 30-32). Kada se promeni vrednost na traci, osluškivač je obavešten pozivanje odrađivača (handler) da postavi novu x vrednost početka teksta u skladu sa novo postavljenom vrednošću na horizontalnoj tracia za pomeranje (linije 31-32).

Isto tako se registruje osluškivač za svojstvo vrednosti klizača sbVertical (linije 35-37). Kada se promeni vrednost na traci, osluškivač je obavešten pozivanjem metoda ***handler()*** i on postavlja novu vrednost za y koja odgovara trenutnoj vrednosti na traci sbVertical (linije 30-37). Umesto linija 30-37 može se koristiti i sledeći kod:

```
text.xProperty().bind(sbHorizontal.valueProperty().
   multiply(paneForText.widthProperty()).
   divide(sbHorizontal.maxProperty()));
 text.yProperty().bind(sbVertical.valueProperty().multiply(
   paneForText.heightProperty().divide(
   sbVertical.maxProperty())));
```

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.geometry.Orientation;
import javafx.scene.Scene;
import javafx.scene.control.ScrollBar;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class ScrollBarDemo extends Application {

    @Override // Redefinisanje metoda start() klase Annimation
    public void start(Stage primaryStage) {
        Text text = new Text(20, 20, "JavaFX Programming");

        ScrollBar sbHorizontal = new ScrollBar();
        ScrollBar sbVertical = new ScrollBar();
        sbVertical.setOrientation(Orientation.VERTICAL);

        // Kreiranje teksta u okbu
        Pane paneForText = new Pane();
        paneForText.getChildren().add(text);

        // Kreiranje graničnog okna za držanje teksta i traka za pomeranje
        BorderPane pane = new BorderPane();
        pane.setCenter(paneForText);
        pane.setBottom(sbHorizontal);
        pane.setRight(sbVertical);
```

```
        // Osluškivač promene vrednosti za horizontalnu traku za pomeranje
        sbHorizontal.valueProperty().addListener(ov
                -> text.setX(sbHorizontal.getValue() * paneForText.getWidth()
                        / sbHorizontal.getMax()));

        // Osluškivač promene vrednosti vertikalne trake za pomeranje
        sbVertical.valueProperty().addListener(ov
                -> text.setY(sbVertical.getValue() * paneForText.getHeight()
                        / sbVertical.getMax()));

        // Kreiranje scene i njeno postavljanjena pozornicu
        Scene scene = new Scene(pane, 450, 170);
        primaryStage.setTitle("ScrollBarDemo"); // Set the stage title
        primaryStage.setScene(scene); // Place the scene in the stage
        primaryStage.show(); // Display the stage
    }

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

}
```

ZADATAK 9 - SAMOSTALNO VEŽBANJE KLASE SCROLLBAR

Dodajte ScrollBar na dnu scene.

Napravite JavaFX program po sledećim zahtevima:

- u razvojnom okruženju kreirati klasu koja nasleđuje klasu kreiranu u prethodnom primeru;
- implementirajte kontolu ScrollBar na dnu scene.

Poglavlje 10

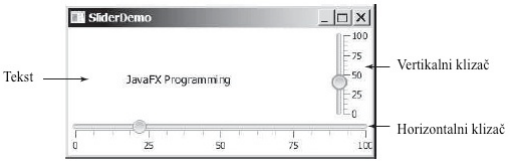
Klasa Slider

SVOJSTVA KLASE SLIDER

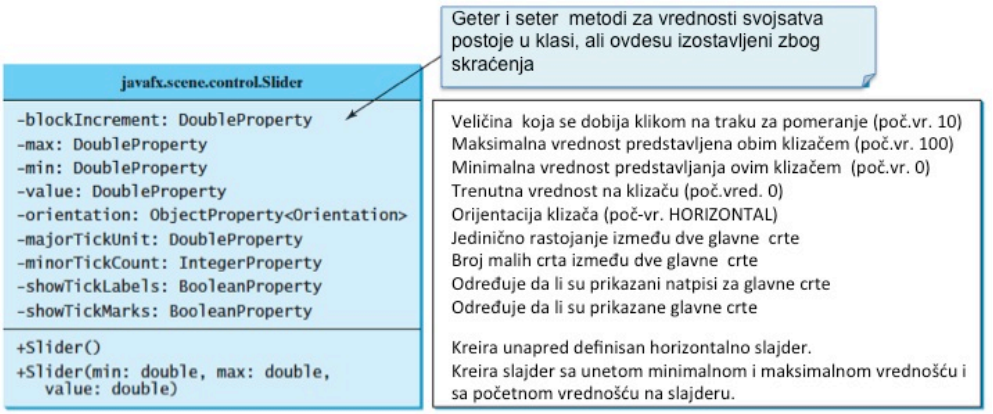
Klasa Slider omogućava korisniku unos vrednosti primenom klizača, tj. trake sa klizačem. Klasa Slider je slična klase ScrollBar, samo pruža više svojstava i omogućava više oblika prikazivanja.

Slika 1 pokazuje primer sa dva klizača. Klizač omogućava korisniku da grafički izabere vrednost pomeranjem klizača duž trake, a između dve granične vrednosti. Klizač može da koristi i glavne (veće) i manje crte između glavnih crta. Broj piksela između glavnih (debelih) crta se određuje vrednošću atributa ***majorTickUnit*** i ***minorTickUnit.***

Klizači se mogu predstaviti u horizontalnom i vertikalnom pravcu , sa ili bez crta, sa ili bez natpisa za crte.



Slika 10.1 Klizačem se tekst pomera levo-desno i gore-dole



Slika 10.2 Svojstva klase Slider

PRIMER 10.1 - KORIŠĆENJE KLASE SLIDER – KLASA SLIDERDEMO

Klasa SliderDemo koristi vertikalni i horizontalni klizač za pomeranje teksta u horizontalnom i vertikalnom pravcu.

Listing klase **SliderDemo** omogućava kreiranje i korišćenje korisničkog interfejsa (GUI) prikazanog na slici 1. Klizač (***Slider***) je sličan trakama za pomeranje (***ScrollBar***), ali ima više svojstava. Na primer, možete prikazati glavne i sporedne oznake sa njihovim natpisima (linije 16-17). Jedan osluškivač se registruje za prijem promenjene vrednosti horizontalnog klizača slHorizontal (linije 35-37), a drugi se registruje da prati promene vrednosti na vertikalnom klizaču sbVertical (linije 39-41). Kda se promeni vrednost na klizaču, osluškivač se obaveštava pozivom metoda ***handle()*** koji postavlja tekst u novi položaj (linije 36-37, 40-41). Vrednost na vertikalnom klizaču opada odozgo nadole.

Linije 35-41 se mogu zamenitii sledećim kodom:.

```
text.xProperty().bind(slHorizontal.valueProperty().
   multiply(paneForText.widthProperty()).
   divide(slHorizontal.maxProperty()));
 text.yProperty().bind((slVertical.maxProperty().subtract(
   slVertical.valueProperty()).multiply(
   paneForText.heightProperty().divide(
   slVertical.maxProperty()))));
```

```
import javafx.application.Application;
import javafx.geometry.Orientation;
import javafx.scene.Scene;
import javafx.scene.control.Slider;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.Pane;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class SliderDemo extends Application {

    @Override // Redefinisanje metoda start() klase Application
    public void start(Stage primaryStage) {
        Text text = new Text(20, 20, "JavaFX Programming");

        Slider slHorizontal = new Slider();
        slHorizontal.setShowTickLabels(true);
        slHorizontal.setShowTickMarks(true);

        Slider slVertical = new Slider();
        slVertical.setOrientation(Orientation.VERTICAL);
        slVertical.setShowTickLabels(true);
        slVertical.setShowTickMarks(true);
        slVertical.setValue(100);
```

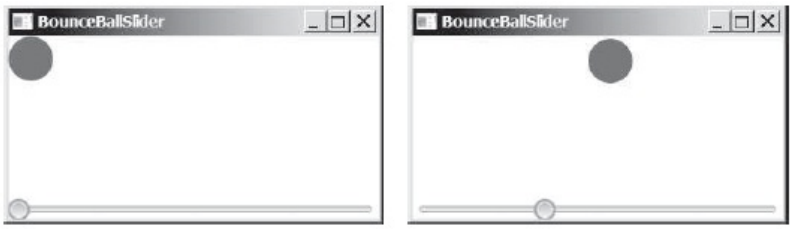
```
        // Kreiranje teksta u okviru
        Pane paneForText = new Pane();
        paneForText.getChildren().add(text);

        // Kreiranje graničnog okvira koji sadrži tekst i klizače
        BorderPane pane = new BorderPane();
        pane.setCenter(paneForText);
        pane.setBottom(slHorizontal);
        pane.setRight(slVertical);

        slHorizontal.valueProperty().addListener(ov
                -> text.setX(slHorizontal.getValue() * paneForText.getWidth()
                        / slHorizontal.getMax()));

        slVertical.valueProperty().addListener(ov
                -> text.setY((slVertical.getMax() - slVertical.getValue())
                        * paneForText.getHeight() / slVertical.getMax()));

        // Kreiranje scene i njeno postavljanje na pozornicu
        Scene scene = new Scene(pane, 450, 170);
        primaryStage.setTitle("SliderDemo"); // Postavljanje naslova pozornice
        primaryStage.setScene(scene); // Postavljanje scene na pozornicu
        primaryStage.show(); // Prikaz pozprnice
    }

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

PRIMER 10.2 - KLASA BOUNCEBALLSLIDER

Listing klase BounceBallSlider prikazuje program koji vrši animaciju kretanja lopte koja se kreće u pravougnom okviru

Listing klase **BounceBallSlider** prikazuje program koji vrši animaciju kretanja lopte koja se kreće u pravougnom okviru (slika 3).

.



Slika 10.3 Može se preko klizača podešavati brzina kretanja lopte

Metod ***rateProperty()*** klase **BallPane** vraća vrednost brzine animacije kretanja lopte. Animacija se yaustavlja kada brzina postane 0. Ako je brzina veća od 20, animacija postaje isuviše brza. Zato su granične vrednosti brzina postavljene od 0 do 20. Ove se vrednosti definišu u liniji 13 . Zato, maksimalno moguća brzima kretanja lopte je 20 (linija 12).

Klasa BounceBallSlider:

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.control.Slider;
import javafx.scene.layout.BorderPane;

public class BounceBallSlider extends Application {
  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    BallPane ballPane = new BallPane();
    Slider slSpeed = new Slider();
    slSpeed.setMax(20);
    ballPane.rateProperty().bind(slSpeed.valueProperty());

    BorderPane pane = new BorderPane();
    pane.setCenter(ballPane);
    pane.setBottom(slSpeed);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 250, 250);
    primaryStage.setTitle("BounceBallSlider"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

  /**
   * The main method is only needed for the IDE with limited
   * JavaFX support. Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}
```

Klasa BallPane:

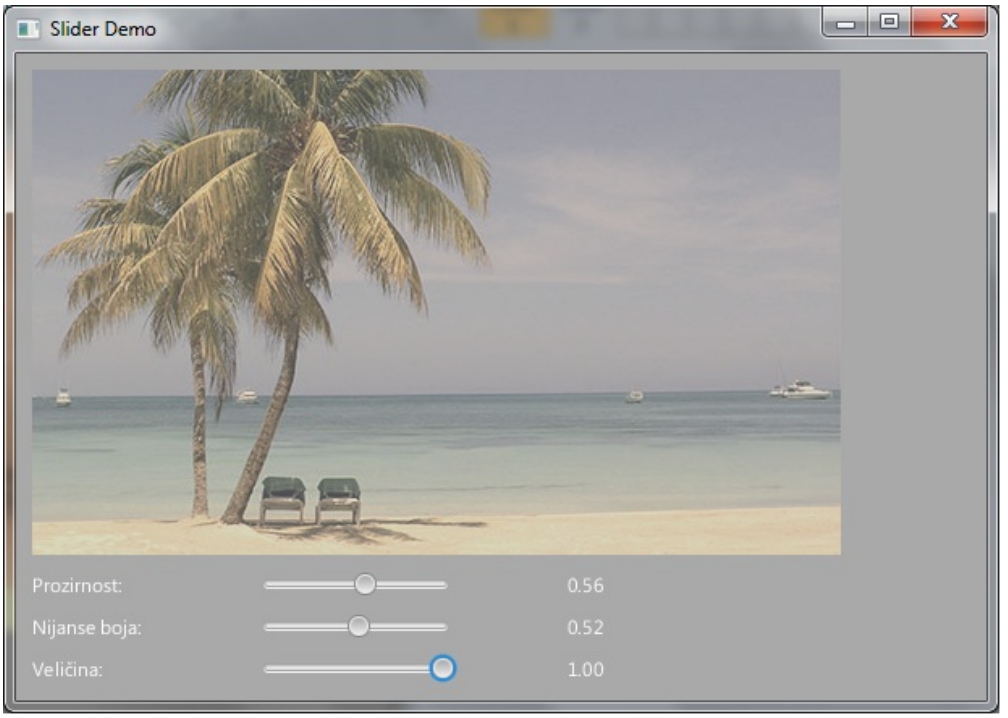
```
import javafx.animation.KeyFrame;
import javafx.animation.Timeline;
import javafx.beans.property.DoubleProperty;
import javafx.scene.layout.Pane;
import javafx.scene.paint.Color;
import javafx.scene.shape.Circle;
import javafx.util.Duration;

public class BallPane extends Pane {
  public final double radius = 20;
```

```
  private double x = radius, y = radius;
  private double dx = 1, dy = 1;
  private Circle circle = new Circle(x, y, radius);
  private Timeline animation;

  public BallPane() {
    circle.setFill(Color.GREEN); // Set ball color
    getChildren().add(circle); // Place a ball into this pane

    // Create an animation for moving the ball
    animation = new Timeline(
      new KeyFrame(Duration.millis(50), e -> moveBall()));
    animation.setCycleCount(Timeline.INDEFINITE);
    animation.play(); // Start animation
  }

  public void play() {
    animation.play();
  }

  public void pause() {
    animation.pause();
  }

  public void increaseSpeed() {
    animation.setRate(animation.getRate() + 0.1);
  }

  public void decreaseSpeed() {
    animation.setRate(
      animation.getRate() > 0 ? animation.getRate() - 0.1 : 0);
  }

  public DoubleProperty rateProperty() {
    return animation.rateProperty();
  }

  protected void moveBall() {
    // Check boundaries
    if (x < radius || x > getWidth() - radius) {
      dx *= -1; // Change ball move direction
    }
    if (y < radius || y > getHeight() - radius) {
      dy *= -1; // Change ball move direction
    }

    // Adjust ball position
    x += dx;
    y += dy;
    circle.setCenterX(x);
    circle.setCenterY(y);
  }
}
```

PRIMER 10.3

Cilj zadatka je da se demonstrira rad sa klasom Slider

Napraviti program koji na ekranu prikazuje sliku i omogućava da se pomoću objekata klase Slider menjaju transparentost (opacity), jačina boja (sepia tone) i veličina slike (scale).



Slika 10.4 Zadatak 3

PRIMER 10.3 - OBJAŠNJENJE I UPUTSTVO ZA REŠAVANJE

U nastavku je dato pojašnjenje rešenja zadatka 3

Objašnjenje i uputstva:

1. Kreirati JavaFX aplikaciju. Pokretačku klasu nazvati SliderDemo.java

2. Nakon dodavanje svih neophodnih komponenti na pozornicu, potrebno je kontrolisati rad slider-a, tako što će prvi slider menjati prozirnost (opacity), drugi nijansu boje (sepia) i treći veličinu slike (scale). Osobina prozirnosti i veličine su osobine koje mogu postavljati direktno metodi klase komponente ImageView, dok ćemo za nijansu boje koristiti objekat klase SepiaTone.java

3. Promenu prozirnosti vršimo na jednostavan način metodom setOpacity u okviru listener-a koji osluškuje promenu vrednosti na slideru:

```
opacityLevel.valueProperty().addListener(new ChangeListener<Number>() {
public void changed(ObservableValue<? extends Number> ov,
Number old_val, Number new_val) {
```

```
imgView.setOpacity(new_val.doubleValue());
opacityValue.setText(String.format("%.2f", new_val));
}
});
```

4. Promenu nijanse boje detektujemo dodavanjem listener-a odgovarajućem slider-u u okviru koga menjamo nivo jačine boja. Bitno je da pre samog listener-a imamo postavljen efekat imgView.setEffect(sepiaEffect); jer u suprotnom promena neće imati uticaja na samu sliku

```
sepiaTone.valueProperty().addListener(new ChangeListener<Number>() {
public void changed(ObservableValue<? extends Number> ov,
Number old_val, Number new_val) {
sepiaEffect.setLevel(new_val.doubleValue());
sepiaValue.setText(String.format("%.2f", new_val));
}
});
```

3. Promenu veličine slike vršimo na jednostavan način metodom setScale X i setScaleY u okviru listener-a koji osluškuje promenu vrednosti na poslednjem slideru:

```
scaling.valueProperty().addListener(new ChangeListener<Number>() {
public void changed(ObservableValue<? extends Number> ov,
Number old_val, Number new_val) {
imgView.setScaleX(new_val.doubleValue());
imgView.setScaleY(new_val.doubleValue());
scalingValue.setText(String.format("%.2f", new_val));
}
});
```

PRIMER 10.3 - REŠENJE

Programiski kod koji predstvalja rešenje zadatka 3

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.beans.value.ChangeListener;
import javafx.beans.value.ObservableValue;
import javafx.geometry.Insets;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Slider;
import javafx.scene.effect.SepiaTone;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.layout.GridPane;
import javafx.scene.paint.Color;
import javafx.scene.text.Text;
import javafx.stage.Stage;

public class SliderDemo extends Application {
```

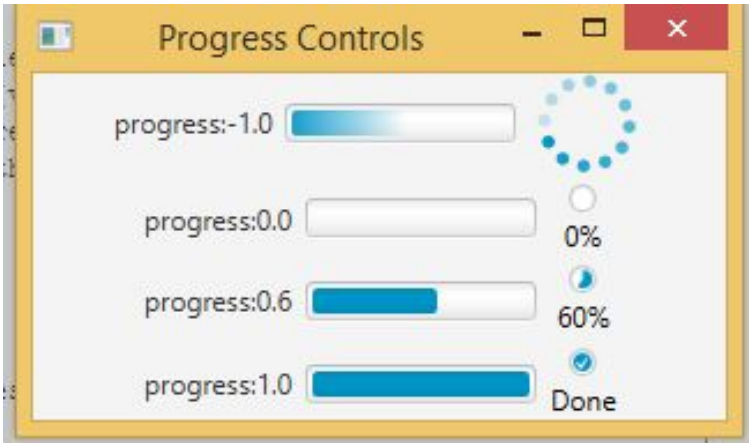
```
    //kreiranje objekata Slider
    Slider opacityLevel = new Slider(0, 1, 1);
    Slider sepiaTone = new Slider(0, 1, 1);
    Slider scaling = new Slider(0.5, 1, 1);
    //U okviru samog projekta mora da se nalazi slika jamaica.jpg
    Image image = new Image(getClass().getResourceAsStream("jamaica.jpg"));

    //kreiranje polja za slajdere
    Text opacityCaption = new Text("Prozirnost: ");
    Text sepiaCaption = new Text("Nijanse boja:");
    Text scalingCaption = new Text("Veličina:");

    //Kreiranje teksta za vrednost slajdera
    Text opacityValue = new Text(Double.toString(opacityLevel.getValue()));
    Text sepiaValue = new Text(Double.toString(sepiaTone.getValue()));
    Text scalingValue = new Text(Double.toString(scaling.getValue()));

    //objekat klase koji služi za postavljanje sepia efekta.
    SepiaTone sepiaEffect = new SepiaTone();

    @Override
    public void start(Stage stage) {
        Group root = new Group();
        Scene scene = new Scene(root, 600, 400);
        stage.setScene(scene);
        stage.setTitle("Slider Demo");
        scene.setFill(Color.DARKGRAY);

        GridPane grid = new GridPane();
        grid.setPadding(new Insets(10, 10, 10, 10));
        grid.setVgap(10);
        grid.setHgap(70);

        ImageView imgView = new ImageView(image);
        imgView.setFitHeight(300);
        imgView.setFitWidth(600);
        imgView.setEffect(sepiaEffect);
        GridPane.setConstraints(imgView, 0, 0);
        GridPane.setColumnSpan(imgView, 3);
        grid.getChildren().add(imgView);
        scene.setRoot(grid);

        GridPane.setConstraints(opacityCaption, 0, 1);
        grid.getChildren().add(opacityCaption);

        //Listener za promenu vrednosti slidera opacity
        opacityLevel.valueProperty().addListener(new ChangeListener<Number>() {
            public void changed(ObservableValue<? extends Number> ov,
                    Number old_val, Number new_val) {
                imgView.setOpacity(new_val.doubleValue());
                opacityValue.setText(String.format("%.2f", new_val));
            }
        });
```

```
        GridPane.setConstraints(opacityLevel, 1, 1);
        grid.getChildren().add(opacityLevel);

        GridPane.setConstraints(opacityValue, 2, 1);
        grid.getChildren().add(opacityValue);

        GridPane.setConstraints(sepiaCaption, 0, 2);
        grid.getChildren().add(sepiaCaption);

        //Listener za promenu vrednosti slidera sepia
        sepiaTone.valueProperty().addListener(new ChangeListener<Number>() {
            public void changed(ObservableValue<? extends Number> ov,
                    Number old_val, Number new_val) {
                sepiaEffect.setLevel(new_val.doubleValue());
                sepiaValue.setText(String.format("%.2f", new_val));
            }
        });
        GridPane.setConstraints(sepiaTone, 1, 2);
        grid.getChildren().add(sepiaTone);

        GridPane.setConstraints(sepiaValue, 2, 2);
        grid.getChildren().add(sepiaValue);

        GridPane.setConstraints(scalingCaption, 0, 3);
        grid.getChildren().add(scalingCaption);

        //Listener za promenu vrednosti slidera scaling-a
        scaling.valueProperty().addListener(new ChangeListener<Number>() {
            public void changed(ObservableValue<? extends Number> ov,
                    Number old_val, Number new_val) {
                imgView.setScaleX(new_val.doubleValue());
                imgView.setScaleY(new_val.doubleValue());
                scalingValue.setText(String.format("%.2f", new_val));
            }
        });
        GridPane.setConstraints(scaling, 1, 3);
        grid.getChildren().add(scaling);

        GridPane.setConstraints(scalingValue, 2, 3);
        grid.getChildren().add(scalingValue);

        stage.show();
    }

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

PRIMER 10.4

Cilj zadatka je da se demonstrira rad sa klasom ProgressBar

Napraviti program koji prikazuje sve vrste Progress kontrola. Ukoliko je progres u minusu pokazaće se Loader. Ukoliko je 0 treba da se pokaže kružić sa 0% a potom prikazati vrednosti i za 60% kao i 100%



Slika 10.5 Zadatak 4

PRIMER 10.4 - OBJAŠNJENJE I REŠENJE

U nastavku je dato pojašnjenje rešenja zadatka 4

Ovaj primer demonstrira rad sa komponentom ProgressBar. Komponenta ima mogućnost prikaza nekog vida animiranog progress bar-a setovanjem vrednosti na -1.0f što je dato na prvom bar-u. U svim ostalim primerima komponenta prikazuje procentualno dodeljene vrednosti iz opsega 0-1. Isti način koristi i ProgressIndicator da ukaže na trenutnu vrednost samog ProgressBar-a.

Klasa ProgressExample:

```
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.geometry.Pos;
import javafx.scene.Group;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.ProgressBar;
import javafx.scene.control.ProgressIndicator;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class ProgressExample extends Application {

    final Float[] values = new Float[]{-1.0f, 0f, 0.6f, 1.0f};
```

```
    final Label[] labels = new Label[values.length];
    final ProgressBar[] pbs = new ProgressBar[values.length];
    final ProgressIndicator[] pins = new ProgressIndicator[values.length];
    final HBox hbs[] = new HBox[values.length];

    @Override
    public void start(Stage stage) {
        Group root = new Group();
        Scene scene = new Scene(root, 300, 150);
        stage.setScene(scene);
        stage.setTitle("Progress Controls");

        for (int i = 0; i < values.length; i++) {
            final Label label = labels[i] = new Label();
            label.setText("progress:" + values[i]);

            final ProgressBar pb = pbs[i] = new ProgressBar();
            pb.setProgress(values[i]);

            final ProgressIndicator pin = pins[i] = new ProgressIndicator();
            pin.setProgress(values[i]);
            final HBox hb = hbs[i] = new HBox();
            hb.setSpacing(5);
            hb.setAlignment(Pos.CENTER);
            hb.getChildren().addAll(label, pb, pin);
        }

        final VBox vb = new VBox();
        vb.setSpacing(5);
        vb.getChildren().addAll(hbs);
        scene.setRoot(vb);
        stage.show();
    }

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

ZADATAK 10 - SAMOSTALNO VEŽBANJE KLASE SLIDER

Implementirajte kontolu Slider na desnoj ivici scene

Napravite JavaFX program po sledećim zahtevima:

- u razvojnom okruženju kreirati klasu koja nasleđuje klasu kreiranu u prethodnom primeru;
- implementirajte kontolu Slider na desnoj ivici scene.

VIDEO - RAD SA TABELAMA

avaFX Java GUI Tutorial - 17 - Introduction to TableView (5,30)

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

VIDEO - TABLEVIEW

avaFX Java GUI Tutorial - 18 - Simple TableView (12,30 minuta)

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

VIDEO - PROMENILJIVE TABELE

JavaFX Java GUI Tutorial - 19 - Editable Tables (7,52 minuta)

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

VIDEO - DODAVANJE I BRSIANJE REDOVA U TABLEVIEW

JavaFX Java GUI Tutorial - 20 - Adding and Deleting TableView Rows (10.04)

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

Poglavlje 11

Video i Audio

SVOJSTVA KLASA MEDIA I MEDIAPLAYER

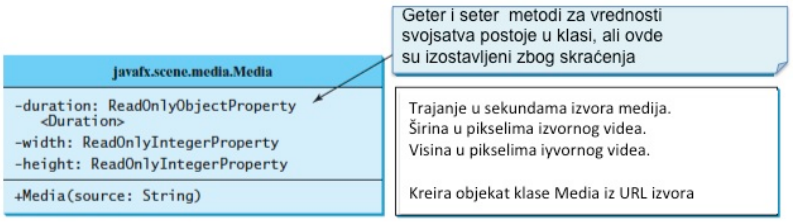
Klasa Media određuje izvor medije, klasa MediaPlayer izvršava i kontroliše mediju (video ili audio prikaz),

Klasa ***Media*** određuje izvor medije, klasa ***MediaPayer*** izvršava i kontroliše mediju (video ili audio prikaz), a klasa MediaView prikazuje video.

Media (video i audio prikaz) je bitan u razvoju bogatih Internet aplikacija. ***JavaFX*** obezbeđuje klase ***Media***. ***MediaPlayer*** i ***MediaView*** za rad sa medijima. Trenutno, ***JavaFX*** podržava ***MP3, AIFF, WAV*** i ***MPEG-4*** autio formate i ***FLV*** i ***MPEG-4*** video formate.

Klasa ***Media*** predstavlja izvor medija sa svojstvoma kao što su trajanje, širina i visina, kao što je prikazano na slici 1. Možete kreirati ***Media*** objekat iz ***URL*** strnga.

Klasa ***MediaPlayer*** emituje i kontroliše medij sa svojstvima kao što su: ***autoPlay*** ,***currentCount*** , ***cycleCount , mute*** , ***volume*** , i ***totalDuration*** , kao što je prikazano na slici 2 Možete da konstruišete***MediaPlayer*** objekat iz klase ***Media*** a možete upotrebiti metode ***pause()*** i ***play()*** da bi privremeno zaustavili izvršenje prikaza medija, odn. da bi obnovili izvršenje prikaza medija.



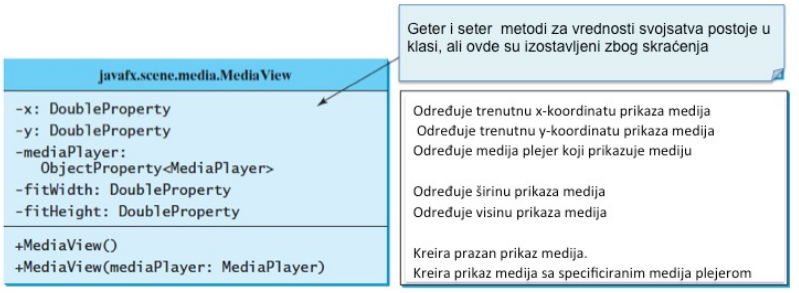
Slika 11.1 Klasa Media predstavlja izvor medija, kao što je video i audio zapis

Slika 11.2 MediaPlayer izvršava medij, tj. video i audio zapis

SVOJSTVA KLASE MEDIAVIEW

Klasa MediaView obezbeđuje svojstva koja omogućavaju prikaz (gledanje i slušanje) medija, tj. video i audio prikaza

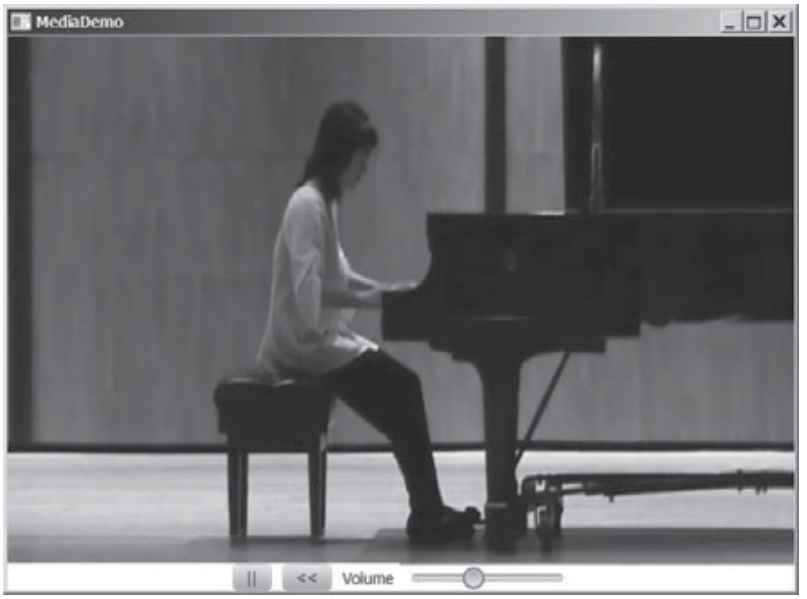
Klasa ***MediaView*** je podklasa klase ***Node*** koja obezbeđuje prikaz objekta ***Media*** koji je u izvršenju od strane objekta ***MediaPlayer***. Klasa ***MediaView*** obezbeđuje svojsva za gledanje i slušanje media, kao što je to prikazano na slici 3.



Slika 11.3 Klasa MediaView obezbeđuje svojstva neophodna za prikaz i kontrolu medija u izvršenju

Klasa ***Media*** određuje izvor medije, klasa ***MediaPayer*** izvršava i kontroliše mediju (video ili audio prikaz), a klasa ***MediaView*** prikazuje video. Slika 5 prikazuje njihov odnos.

Listing klase **MediaDemo** je primer prikazivanja video prikaza, kao što je prikazan na slici 4. Možete koristi **play/pause** dugme kao i **rewind** dugme za ponovno startovnje video prikaza. **Slajder** koristite za podešavanje jačine zvuka.



Slika 11.4 Program MediaDemo kontroliše i prikazuje medija zapis

PRIMER 11 - PROGRAM MEDIADEMO

Program MediaDemo kontroliše prikaz medija, tj. video i audio prikaza

Izvor medija je URL string definisan u niniji 17 i 18. Program kreira Media objekat sa ove URL adrese (linija 22), a objekat MediaPlayer objekat i objekta Media (linij 24). Media objekat podržava striming. Sada možede prebaciti veliki media fajl i izvršavati ga istovremeno. Media objekat se može deliti (istovremeno koristiti) od strane veše objekata MediaPlayer, a isti MediaPlayer objekat može istovremeno prikazivati različite prikaze medija.

Play/pause dugme se kreira u liniji 26 da bi izvršavalo prikaz ili zaustavljala prikaz (linija 29). Teks dugma je promenjen u || (linija 30) ako je trenutni tekst dugmeta > (linija 28). Ako je trenutni tekst ||, onda se on menja u > (linija 33) i plejer pauzira (linija 32). Rewind dugme je kreiran (linija 37) koji resetuje plejbek vreme na početak medija strima, pozivanjem metoda seek(Duration.ZERO) (linija 38). Objekat Slider je kreiran u liniji 40 radi podešavanja jačine zvuka. Linije 45 i 46 povezuju svojstvo jačine zvuka sa slajderom. Dugme i slajder se postavljaju u Hbox okno (linij 48-51) i prikaz medije je postavljen u centar graničnog okna (linija 54) a Hbox okno se postavlja u donji deo graničnog okna (linija 55).

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.geometry.Pos;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.Slider;
import javafx.scene.layout.BorderPane;
import javafx.scene.layout.HBox;
import javafx.scene.layout.Region;
import javafx.scene.media.Media;
import javafx.scene.media.MediaPlayer;
import javafx.scene.media.MediaView;
import javafx.util.Duration;

public class MediaDemo extends Application {
  private static final String MEDIA_URL =
    "http://cs.armstrong.edu/liang/common/sample.mp4";

  @Override // Override the start method in the Application class
  public void start(Stage primaryStage) {
    Media media = new Media(MEDIA_URL);
    MediaPlayer mediaPlayer = new MediaPlayer(media);
    MediaView mediaView = new MediaView(mediaPlayer);

    Button playButton = new Button(">");
    playButton.setOnAction(e -> {
      if (playButton.getText().equals(">")) {
        mediaPlayer.play();
        playButton.setText("||");
      } else {
```

```
        mediaPlayer.pause();
        playButton.setText(">");
      }
    });

    Button rewindButton = new Button("<<");
    rewindButton.setOnAction(e -> mediaPlayer.seek(Duration.ZERO));

    Slider slVolume = new Slider();
    slVolume.setPrefWidth(150);
    slVolume.setMaxWidth(Region.USE_PREF_SIZE);
    slVolume.setMinWidth(30);
    slVolume.setValue(50);
    mediaPlayer.volumeProperty().bind(
      slVolume.valueProperty().divide(100));

    HBox hBox = new HBox(10);
    hBox.setAlignment(Pos.CENTER);
    hBox.getChildren().addAll(playButton, rewindButton,
      new Label("Volume"), slVolume);

    BorderPane pane = new BorderPane();
    pane.setCenter(mediaView);
    pane.setBottom(hBox);

    // Create a scene and place it in the stage
    Scene scene = new Scene(pane, 650, 500);
    primaryStage.setTitle("MediaDemo"); // Set the stage title
    primaryStage.setScene(scene); // Place the scene in the stage
    primaryStage.show(); // Display the stage
  }

  /**
   * The main method is only needed for the IDE with limited
   * JavaFX support. Not needed for running from the command line.
   */
  public static void main(String[] args) {
    launch(args);
  }
}
```

ZADATAK 11 - KREIRANJE VLASTITOG MEDIA PLAYER-A

Kreirajte vlastiti media player-a

- Napraviti interfejs za muzički player, koji treba imati dugmiće za pokreni, zaustavi, sledeća i prethodna pesma.
- Pesme koje će biti ponuđene moraju se nalaziti u objektu klase ListView ili ComboBox kako bi korisnik mogao da ih odabere.

- Menjanje pesmi se takođe može vršiti pomoću dugmića za puštanje sledeće i prethodne pesme
- Povežite kreirani interfejs sa odgovarajućim funkcionalnostima media player-a.

Poglavlje 12

Domaći zadaci

ZADACI 1-5

Cilj zadataka je da student samostalno primeni korišćenje UI kontrola

U dogovoru sa Vašim asistentom - tutorom biće Vam dodeljenadva zadatka, iz liste ponuđenih,za domaći rad.

1. Napraviti formu koja sadrži polja ime, prezime, pol, godina, mesec, dan rođenja, email, šifra i oznaku da li je zaposlen. Polja ime, prezime i email treba da budu objekti klase TextField, šifra PasswordField, godina, mesec i dan rođenja ComboBox-ovi, a polje zaposlen treba biti CheckBox. Potrebno je dodati i dugme Potvrdi nakon čega će se izvršiti provera da li su sva polja forme popunjena i ispisati poruka na ekranu.
2. Napraviti program u kom korisnik može da bira sliku po nazivu iz ListView objekta, a zatim da pomoću slider-a menja transparentnost i jačinu boja na slici. Svaki put kada korisnik odabere neku od slika u pozadini treba da se čuje pesma. Za svaku sliku treba da se pusti druga pesma.
3. Napraviti video player sa sliderom za kretanje kroz video koristeći Media JavaFX komponente.
4. Napraviti audio player sa playlistom. Playlista treba da bude implementirana pomoću ListView objekta.
5. Napraviti program po uzoru na testove za vožnju. Program treba da ima pitanja i odgovore koji su Radio dugmići ili CheckBoxovi.

Zaključak

REZIME

Pouke ove lekcije

1. Apstraktna klasa ***Labeled*** je osnovna klasa za klase ***Label, Button, CheckBox i RadioButton***. Ona definiše svojstva: ***alignment, contentDisplay, text, graphic, graphicTextGap, textFill, underline,*** i***wrapText.***
2. Apstraktna klasa ***ButtonBase*** je osnovna klasa za klase ***Button, CheckBox i RadioButton***. Ona definiše svojstvo ***onAction*** za specifikaciju obrađivača ***(handler***) događaja akcija.
3. Apstraktna klasa ***TextInputControl*** je osnovna klasa za ***TextField*** i ***TextArea*** klase. Ona definiše svojstva: text i ***ediatble.***
4. Objekt klase TextGield ispaljuje događaj akcije kada se klikne Enter taster sa fokusom na polje teksta. Objekat TextArea se često koristi za promenu teksta koji ima više redova.
5. ***ComboBox<T>*** and ***ListView<T>*** su generičke klase za smeštaj elemenata tipa ***T***. Elemnti u padajućem meiju (ComboBox) su generičke klase ya memorisanje elemenata tipa T. Elementi u padajućem meniju ili lista prikaza (slika) se memoriše u ***observable*** listi.
6. Objekat ***ComboBox*** ispaljuje događaj akcije kada se bira nova stavka u ponuđenoj listi stavki.
7. Možete postaviti da se bira samo jedna stavka, ili više stavki padajužeg menija koji daje objekat ***ListView*** i onda dodajete osluškivač za obradu odabranih stavki.
8. Možete koristiti***ScrollBar*** ili ***Slider*** da bi izabrali opseg vrednosti i dodali osluškivač ya vrednost nekog svojstva koji reaguje na promenu te vrednosti.
9. ***JavaFX*** obezbeđuje klasu***Media*** za unos nekog media (video ili audio fajl), klasu***MediaPlayer*** za kontrolu medija i klasu ***MediaView*** za prikaz medija.

.