7. Sedma laboratorijska vježba

7.1. DIZAJNIRANJE JAVAFX GRAFIČKOG SUČELJA

Svrha laboratorijske vježbe je upoznavanje načina dizajniranja grafičkog sučelja korištenjem JavaFX komponenti. Za dizajniranje je potrebno koristiti Scene Builder 1.1., a za razvoj aplikacije je potrebno koristiti Eclipse koji ima sve potrebne *pluginove* i koji je moguće preuzeti kao cjelinu s mrežnih stranica koje će biti navedene u nastavku vježbe. YouTube video koji prikazuje punu funkcionalnost aplikacije koju je potrebno implementirati moguće je vidjeti na sljedećoj poveznici: https://www.youtube.com/watch?v=aIiGl3sJwrQ.

7.2. ZADATAK

Za implementaciju vježbe potrebno je obaviti sljedeće korake i uvažiti primjedbe navedene na kraju dokumenta:

1. S mrežnih stranica „http://www.oracle.com/technetwork/java/javase/downloads/javafxscenebuilder-download-2157683.html" potrebno je preuzeti i instalirati razvojno okruženje Scene Builder.

2. S mrežnih stranica „http://downloads.efxclipse.bestsolution.at/downloads/0.9.0/" potrebno je preuzeti i instalirati Eclipse koji sadrži sve potrebne *pluginove* za razvoj JavaFX aplikacija.

3. Pomoću novog Eclipsea potrebno je kreirati novi „workspace" i unutar njega kreirati novi „JavaFX Project" korištenjem opcije „File->New->Other->JavaFX->JavaFX Project". Projekt je potrebno nazvati po prezimenu i rednom broju vježbe, npr. „Horvat-7". Glavnu klasu koja se generira potrebno je promijeniti tako da izgleda kao sljedeći primjer:

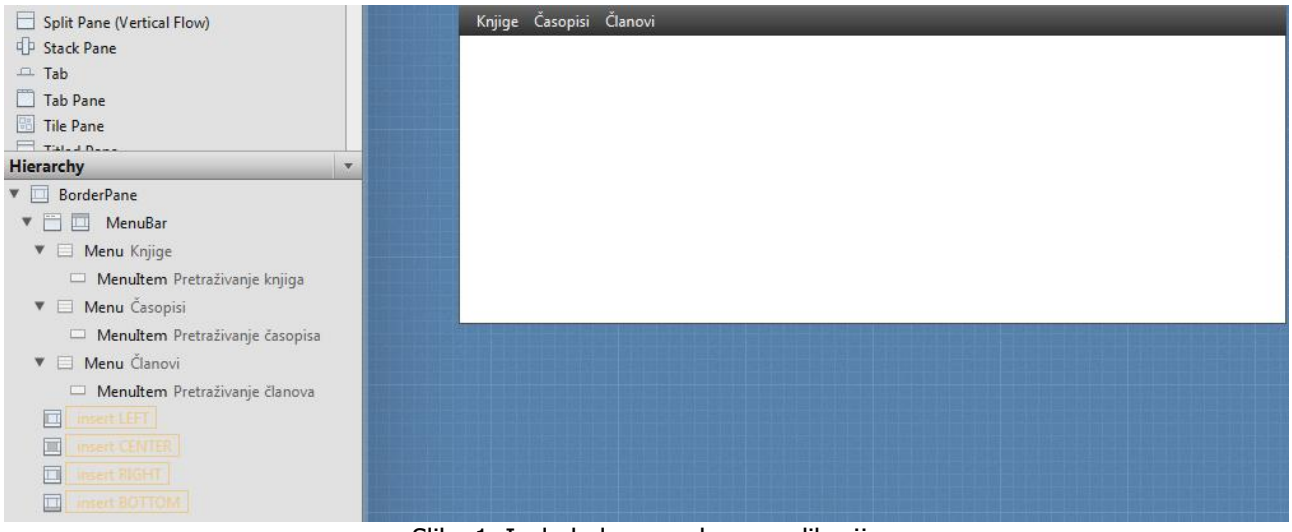
```
public class JavaFXGlavna extends Application {

	private static BorderPane root;
	private Stage primaryStage;

	@Override
	public void start(Stage stage) {
		primaryStage = stage;
		try {
			BorderPane rootPane = (BorderPane)
				FXMLLoader.load(JavaFXGlavna.class
					.getResource("../javafx/glavna.fxml"));
			root = rootPane;
			Scene scene = new Scene(root, 650, 250);
			scene.getStylesheets().add(getClass().getResource(
				"../glavna/application.css").toExternalForm());
			primaryStage.setScene(scene);
			primaryStage.show();
```

```
            } catch (Exception e) {
                  e.printStackTrace();
            }
      }

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

      public static void setCenterPane(BorderPane centerPane) {
            root.setCenter(centerPane);
      }
```

Preporučljivo je da se glavna klasa nalazi u paketu poput naziva „hr.tvz.java.vjezbe.glavna", kako bi se relativno od njega moglo pristupati ostalim resursima aplikacije. Na primjer, „../javafx/glavna.fxml" označava lokaciju datoteke „glavna.fxml" unutar paketa „hr.tvz.java.vjezbe.javafx".

4. Iz šeste vježbe je u sedmi potrebno kopirati paket s domenskim klasama, paket s enumeracijama, paket s iznimkama i tekstualne datoteke koje sadrže podatke o članovima, knjigama i časopisima.

5. Kreirati novi paket u kojem će se nalaziti FXML datoteke s kojima su definirani izgledi ekrana. Unutar tog paketa potrebno je kreirati prvu FXML datoteku pod nazivom „glavna.fxml" koja će sadržavati „kostur" ekrana aplikacije s izbornikom u kojem će se prikazivati ostale funkcionalnosti aplikacije. Datoteku je moguće kreirati korištenjem opcije „File->New->Other->JavaFX->New FXML Document". Nakon toga je potrebno označiti novu datoteku, kliknuti na desnu tipku miša i odabrati opciju „Open With SceneBuilder". Ekran treba dizajnirati tako da izgleda slično onome koji je prikazan na slici 1, korištenjem elemenata koji su prikazani u „Hierarchy" modulu. Prilikom dizajniranja grafičkog sučelja poželjno je koristiti „manual" naveden u četvrtoj napomeni na kraju dokumenta. Svaki od izbornika „Knjige", „Časopisi" i „Članovi" mora imati jedan „podizbornik" koji će pokretati pretraživanje entiteta (npr. „Pretraživanje knjiga").

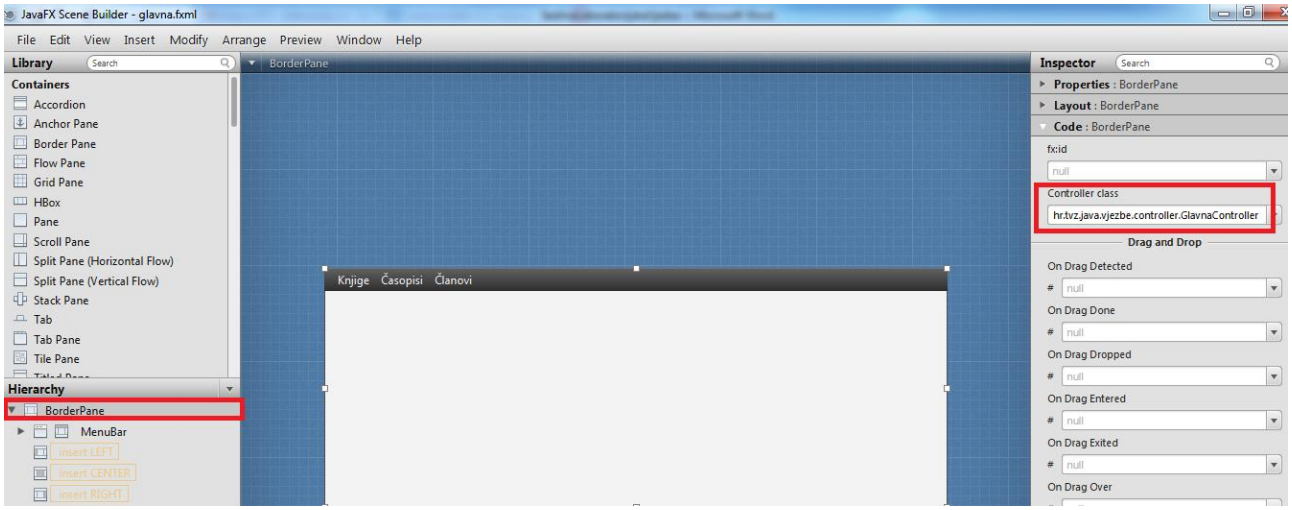


Slika 1. Izgled glavnog ekrana aplikacije

6. Kreirati novi paket pod nazivom npr. „hr.tvz.java.vjezbe.controller" u kojem će se nalaziti sve „Controller" klase koje će se koristiti unutar projekta. Na početku je unutar tog paketa potrebno kreirati klasu pod nazivom „GlavnaController" koja će sadržavati metode koje će se pozivati iz izbornika s ekrana prikazanog na slici 1. Primjer izgleda tog „Controllera" s jednom metodom za prikaz ekrana koji će sadržavati tablicu s knjigama je prikazan u nastavku:

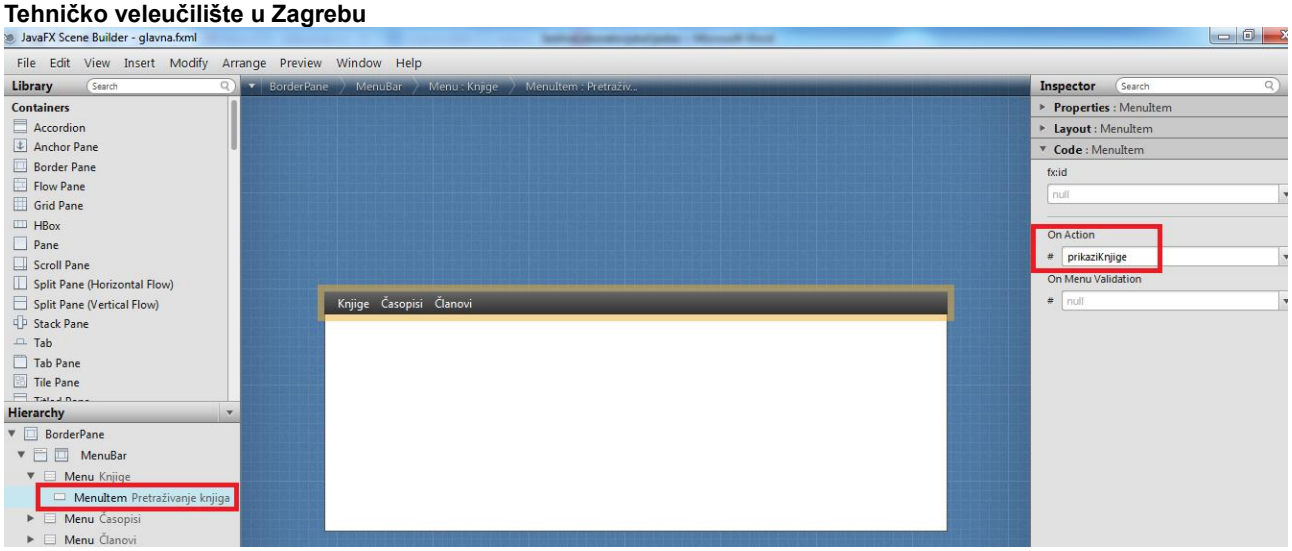
```
public class GlavnaController {

	public void prikaziKnjige() {
		try {
			BorderPane knjigePane = (BorderPane)
				FXMLLoader.load(JavaFXGlavna.class
					.getResource("../javafx/knjige.fxml"));
			JavaFXGlavna.setCenterPane(knjigePane);
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

Nakon što je napisana ta metoda, korištenjem Scene Buildera je potrebno povezati komponentu na vrhu hijerarhije i upisati naziv klase koja označava klasu „GlavnaController" kao na sljedećoj slici (unutar „Code" modula):



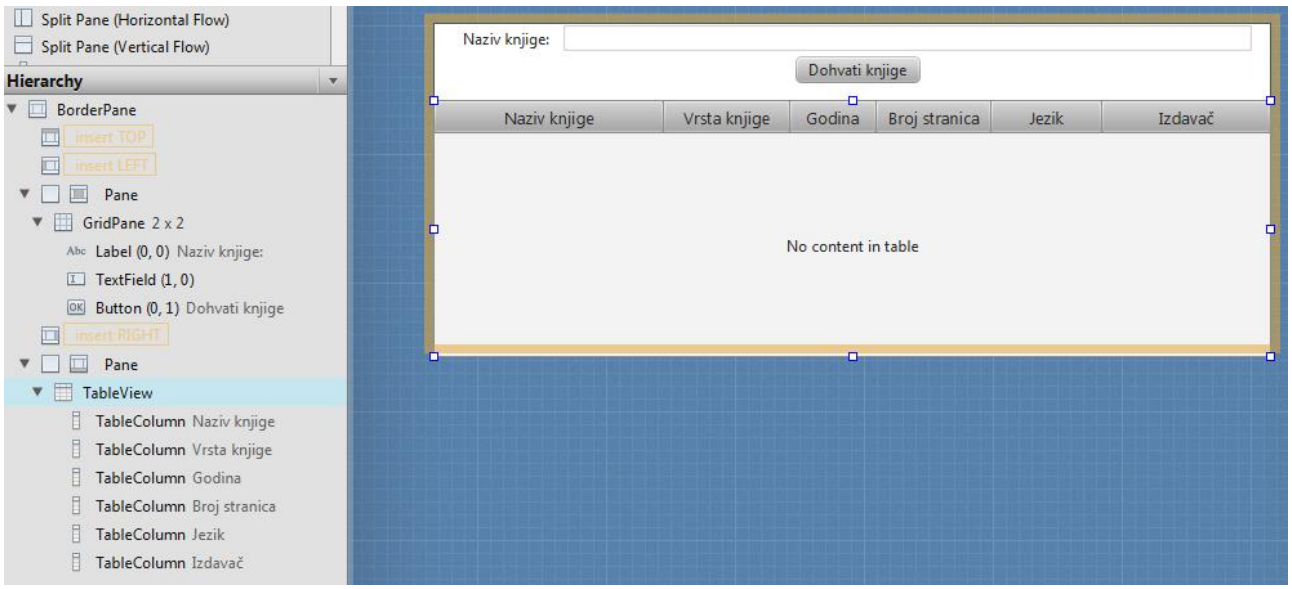
Slika 2. Konfiguriranje „Controller" klase za ekran

Nakon povezivanja ekrana s „Controller" klasom potrebno je povezati izbornik „Pretraživanje knjiga" unutar glavnog izbornika „Knjige". To je moguće napraviti tako da se odabere komponenta s izbornikom i unutar modula „Code" u polje „On Action" upiše naziv metode „prikaziKnjige" iz „GlavnaController" klase koju je potrebno pozvati prilikom odabira te opcije u izborniku (što je prikazano na slici 3).



Slika 3. Konfiguriranje metode „Controllera" koja će se pozivati na odabir te opcije

7. Unutar paketa s FXML datotekama potrebno je kreirati novu datoteku pod nazivom „knjige.fxml" koja će sadržavati komponente za filtriranje zapisa o knjigama i samu tablicu s knjigama.



Slika 4. Izgled ekrana za prikaz i filtriranje podataka o knjigama

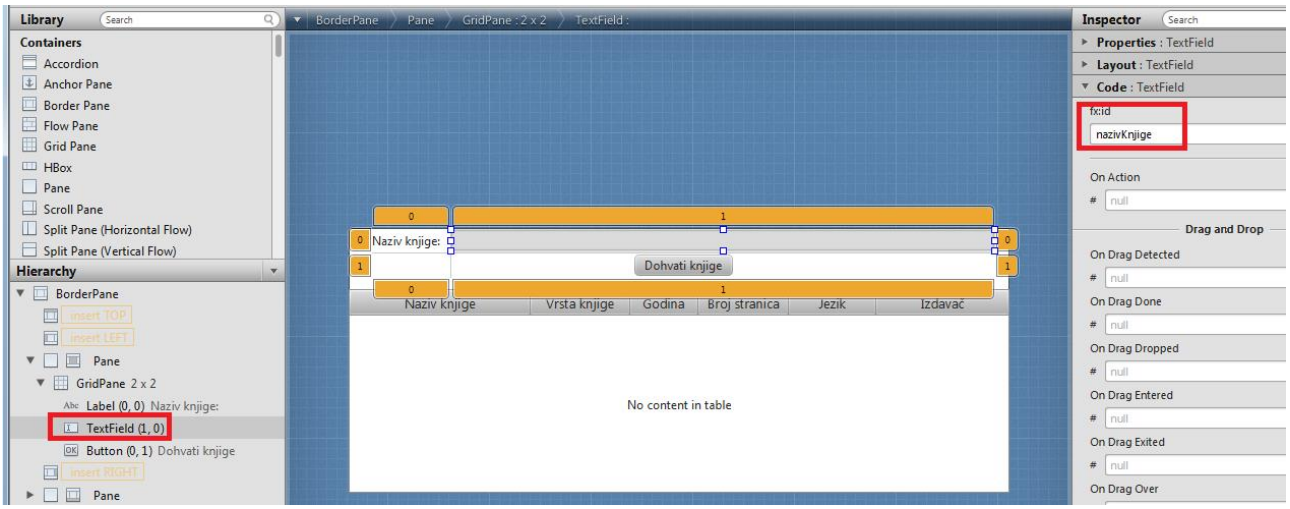
Preporuča se korištenje panela za svaki od segmenata ekrana radi lakšeg manipuliranja rasporedom komponenti na ekranu.

8. U paketu s „Controller" klasama potrebno je kreirati novu klasu „KnjigeController" koja će implementirati funkcionalnosti dohvaćanja podataka o knjigama iz tekstualnih datoteka i njihovo „filtriranje" po nazivu. Osim toga je u tu klasu potrebno „preseliti" cjelokupni kod koji čita tekstualnu datoteku „knjige.txt", kreira objekte klase „Knjiga" i dodaje ih u listu, koju je kasnije potrebno konvertirati u objekt klase „ObservableList". Varijable i metode iz „KnjigeController" je potrebno povezati s grafičkim sučeljem definiranim u sučelju „knjige.fxml".

Prvo je potrebno „knjige.fxml" povezati s nazivom „KnjigeController" klase (kao u šestom koraku), a nakon toga gumb „Dohvati knjige" povezati s metodom „prikaziSveKnjige" (što

je također prikazano u šestom koraku na primjeru izbornika za pretraživanje knjiga).

Sljedeće što je potrebno napraviti vezano je uz „mapiranje" naziva varijabli označenih s „@FXML" anotacijom s identifikatorima na grafičkom sučelju „knjige.fxml" (slika 5):



Naziv varijable „nazivKnjige" unutar klase „KnjigeController" mora se podudarati s poljem „fx:id" u „Code" modulu unutar „Scene Buildera". Na sličan način je potrebno povezati naziv tablice i kolona u tablici.

Unutar metode „initialize" obavljena je inicijalizacija vrijednosti koje će se prikazivati u tablici u određenom stupcu iz određene varijable unutar klase „Knjiga". Na primjer, vrijednost varijable „naziv" iz klase „Knjiga" prikazivati će se unutar stupca pod nazivom „nazivKnjigeColumn". Na kraju je potrebno povezati gumb „Dohvati knjige" s pozivom metode „prikaziSveKnjige".

Prikaz moguće implementacije klase „KnjigeController" prikazan je u nastavku:

```
public class KnjigeController {

      public KnjigeController() {
      }

      @FXML
      private TextField nazivKnjige;

      @FXML
      private TableView<Knjiga> knjigaTable;

      @FXML
    private TableColumn<Knjiga, String> nazivKnjigeColumn;

      @FXML
    private TableColumn<Knjiga, VrstaPublikacije> vrstaKnjigeColumn;

      @FXML
    private TableColumn<Knjiga, Integer> godinaIzdanjaKnjigeColumn;
```

```
    @FXML
    private TableColumn<Knjiga, Integer> brojStranicaKnjigeColumn;

    @FXML
    private TableColumn<Knjiga, Jezik> jezikKnjigeColumn;

    @FXML
    private TableColumn<Knjiga, String> nazivIzdavacaKnjigeColumn;

    @FXML
    public void initialize() {
        nazivKnjigeColumn.setCellValueFactory(
                new PropertyValueFactory<Knjiga, String>("naziv"));
        vrstaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, VrstaPublikacije>("vrstaPublikacije"));
        godinaIzdanjaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, Integer>("godinaIzdanja"));
        brojStranicaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, Integer>("brojStranica"));
        jezikKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, Jezik>("jezik"));
        nazivIzdavacaKnjigeColumn.setCellValueFactory(
new PropertyValueFactory<Knjiga, String>("izdavac"));
    }

    public void prikaziSveKnjige() {
        List<Knjiga> knjige = dohvatiKnjige();

        List<Knjiga> filtriraneKnjige = new ArrayList<Knjiga>();

        if(nazivKnjige.getText().isEmpty() == false) {
            for(Knjiga knjiga : knjige) {
                if(knjiga.getNaziv().indexOf(nazivKnjige.getText())
!= -1) {
                    filtriraneKnjige.add(knjiga);
                }
            }
        }
        else {
            filtriraneKnjige = knjige;
        }

        ObservableList<Knjiga> listaKnjiga =
FXCollections.observableArrayList(filtriraneKnjige);
        knjigaTable.setItems(listaKnjiga);
    }

    public List<Knjiga> dohvatiKnjige() {
        List<Knjiga> listaKnjiga = new ArrayList<Knjiga>();
        BufferedReader reader = null;
```

```
            String fileName = "knjige.txt";
            try {
                  reader = new BufferedReader(new FileReader(new Fi-
le(fileName)));
            } catch (FileNotFoundException e) {
                  e.printStackTrace();
            }

            Knjiga knjiga = null;
            while (true) {
                  try {
                        knjiga = ucitajKnjigu(reader);
                  } catch (IOException e) {
                        e.printStackTrace();
                  }

                  if(knjiga == null) {
                        break;
                  }
                  listaKnjiga.add(knjiga);
            }
            return listaKnjiga;
      }

      private final Knjiga ucitajKnjigu(BufferedReader reader) throws
IOException {
            String nazivKnjige = reader.readLine();

            if(nazivKnjige == null) {
                  return null;
            }

            Jezik odabraniJezik = null;
            Integer uneseniJezik = Integer.parseInt(reader.readLine());

            for(Jezik jezik : Jezik.values()) {
                  if(uneseniJezik == jezik.getKod()) {
                        odabraniJezik = jezik;
                        break;
                  }
            }

            Izdavac noviIzdavac = unesiIzdavaca(reader);
            String godinaIzdavanjaString = reader.readLine();
            Integer godinaIzdavanja = Inte-
ger.parseInt(godinaIzdavanjaString);

            VrstaPublikacije vrstaPublikacije = null;

            String vrstaPublikacijeUnos = reader.readLine();
```

```
            if(VrstaPublikacije.ELEKTRONICKA.getKod() == Inte-
ger.parseInt(vrstaPublikacijeUnos)) {
                  vrstaPublikacije = VrstaPublikacije.ELEKTRONICKA;
            }
            else if (VrstaPublikacije.PAPIRNATA.getKod() == Inte-
ger.parseInt(vrstaPublikacijeUnos)){
                  vrstaPublikacije = VrstaPublikacije.PAPIRNATA;
            }

            String brojStranicaString = reader.readLine();
            Integer brojStranica = Integer.parseInt(brojStranicaString);
            Knjiga novaKnjiga = new Knjiga(nazivKnjige, odabraniJezik, no-
viIzdavac, godinaIzdavanja, vrstaPublikacije, brojStranica);

            return novaKnjiga;
      }

      private final Izdavac unesiIzdavaca(BufferedReader reader) throws
IOException {
            String nazivIzdavaca = reader.readLine();
            String drzavaIzdavaca = reader.readLine();
            Izdavac noviIzdavac = new Izdavac(nazivIzdavaca, drzavaIzdava-
ca);
            return noviIzdavac;
      }
}
```

9. Na sličan način kao što je prikazano za knjige, potrebno je implementirati dohvat i filtriranje podataka o časopisima.

10. Na sličan način kao što je prikazano za knjige, potrebno je implementirati dohvat i filtriranje podataka o članovima.

NAPOMENA:

1. Sve detalje u uputama koji nisu definirani moguće je proizvoljno definirati.

2. Nakon promjene parametara unutar Scene Buildera poželjno je spremiti njegove promjene, te izvršiti naredbu „Refresh" nad projektom unutar Eclipsea kako bi se promjene ažurirale.

3. Tijekom razvoja poželjno je koristiti upute „manuala" s mrežnih stranica: http://code.makery.ch/java/javafx-2-tutorial-intro/.

4. Za korištenje Scene Buildera poželjno je koristiti „manual" sa sljedećih mrežnih stranica: http://docs.oracle.com/javafx/scenebuilder/1/user_guide/jsbpub-user_guide.htm.

5. Aplikaciju je po želji moguće ukrasiti korištenjem CSS atributa kako je prikazano na predavanjima.