JAVA 2

JavaFX 01

Teme

- Java GUI
- JavaFX arhitektura
- Prva JavaFX aplikacija
- FXML
- *Layouts*

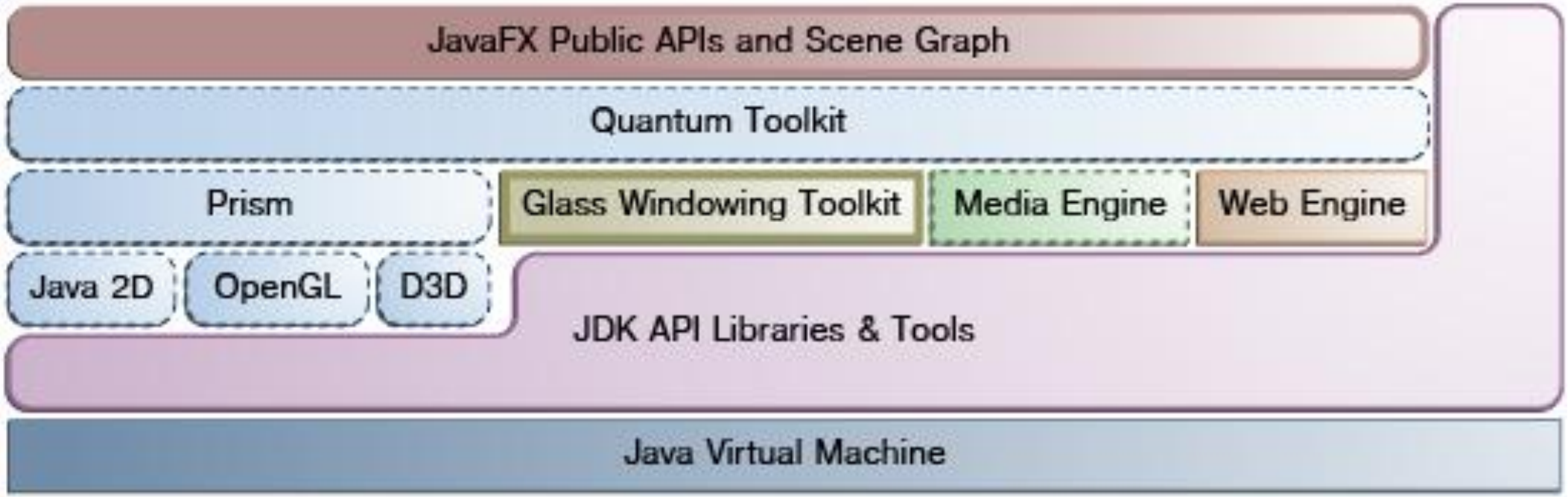
Java GUI

- GUI aplikacija – funkcionalnosti omogućene kroz grafičke elemente
  - kontrole (*controls*) ili widgeti (*widgets–Window Gadgets*)
- GUI komponenta - objekt koji se može koristiti pomoću miša i tipkovnice – korak prema *accessibility*
- AWT (*Abstract Window Toolkit*) – Java SE 1.0
- Swing - Jave SE 1.2 -> Jave SE 7 - primarni skup alata za GUI razvoj
  - trenutno u fazi održavanja
  - prisutan radi kompatibilnosti sa starim verzijama (*backward compatibility)*

Java GUI

- JavaFX - grafički i multimedijski API - 2007. godine kao konkurentna tehnologija *Adobe Flashu* i *Microsoft Silverlightu*
  - 1.0 - 2008. - *JavaFXScript* koji je bio sličan *JavaScriptu* – prevođenje izvornog koda u *bytecode*
  - 2.0 - 2011. - JavaFX je implementiran kao skupina *librarya*
  - JavaFX8 – 2014, sa Java8
  - JavaFX21 – trenutna verzija – zahtijeva JDK >=17
  - *SceneBuilder – standalone visual layout editor*
  - *CSS support*
  - *Transformations*
  - napredno korištenje višedretvenosti na arhitekturama sa više procesora

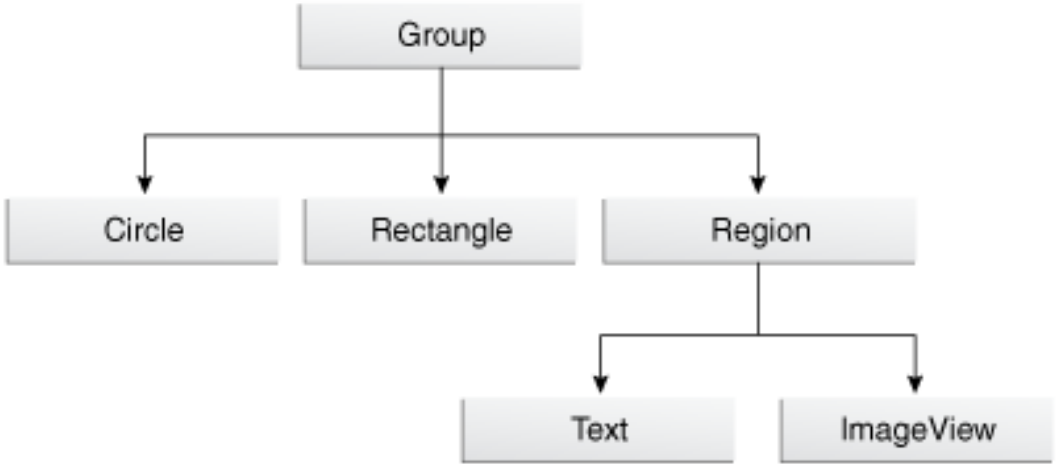
JavaFX arhitektura



Izvor: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-architecture.htm

JavaFX arhitektura

- *Scene Graph* – hijerarhija GUI čvorova (*node*)
  - *node* – osnovni gradbeni elementi (ID, style class, parent (osim *roota*))
    - efekti, prozirnost, transformacije, specifično stanje, obrađivatelji događaja
  - *Rectangle* i *Text* kao elementi sučelja



Izvor: https://docs.oracle.com/javase/8/javafx/scene-graph-tutorial/scenegraph.htm#JFXSG107

JavaFX arhitektura

- Java Public APIs for JavaFX Features
  - bogatstvo mogućnosti u izgradnji GUI klijenata
  - podrška za *Generics, Annotations, Lambda*...
  - mogućnost povezivanja (*binding*) koda sa sučeljem (*lazy, expressions*...)
  - nadogradnja standardnih kolekcija – *observable pattern*
- Graphics System
  - podrška za 2D, 3D scene
  - *gui* cjevovodi ubrzavanja *(graphics accelerated pipelines)*
    - *prism* – renderiranje – DirectX, OpenGL, software kada hardware ubrzanje nije moguće
    - *Glass Window Toolkit* – nativni servisi za *windows, timers, surfaces*
    - *quantum toolkit* – povezivanje *prism* i *Glass Window Toolkit* sa sučeljem prema *JavaFX*
  - *media & images – media - object (mp3, AIFF, WAV, FLV), player, view*
  - *web component – web viewer & engine*

JavaFX arhitektura

- CSS – omogućuje prilagođavanje izgleda sučelja, totalno odvojeno od koda (*loose coupling*) – moraju imati ***-fx:*** prefiks
- UI kontrole
  - povezivanjem nodeova

Izvor: https://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-architecture.htm

JavaFX arhitektura

- Layout – kontejneri omogućuju fleksibilna i dinamička prilagođavanje
  - *BorderPane - top, bottom, right, left i center*
  - *HBox, VBox* – horizontalno u jednom redu, vertikalno u jednoj koloni
  - *StackPane* – stupnjevito
  - *GridPane* – fleksibilni grid
  - *FlowPane* – horizontalni ili vertikalni *flow*
  - *TilePane* – postavljanje u uniformne čelije
  - *AnchorPane* – usidrenje *top, bottom, left i center*

JavaFX arhitektura

- *2D, 3D Transformations* – transformacije x, y, z
  - translate – pomicanje nodova
  - scale – promjena veličine
  - shear – rotacije osi
  - rotate – rotacije *nodeova*
  - affine – linearna mapiranja koordinata (nije *standalone*, koristi se uz ostale transformacije)
- Visual Effects – vizualni efekti za poboljšanje dojma
  - drop shadow – osjenčavanje
  - reflection – reflektivna verzija
  - lighting – osvjetljavanje

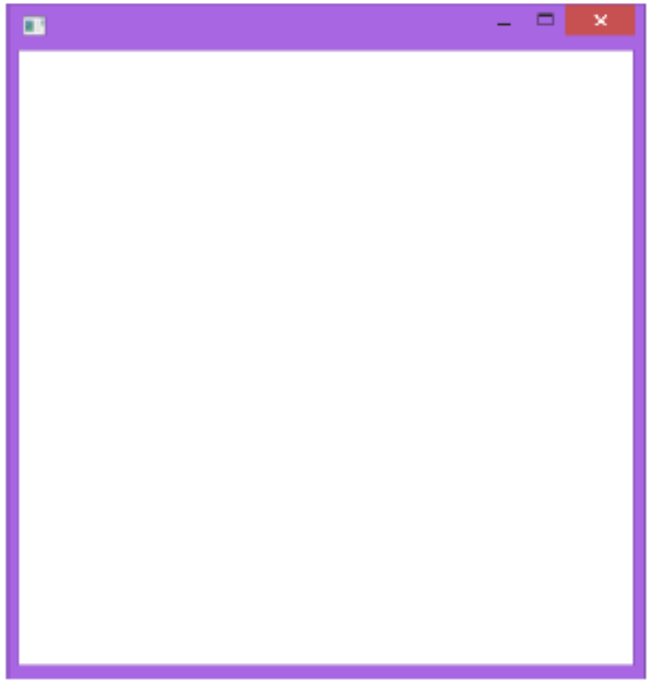
JavaFX arhitektura

- *Threads*
  - *JavaFX application thread*
    - glavni *thread* aplikacije (*event-dispatcher thread*)– komplicirana scena može se izgraditi u pozadinskom *threadu*, ali da bi se povezala sa sučeljem, mora prijeći glavni *thread*
  - *Prism render thread*
    - renderiranje *frameova* u *threadu* odvojenom od glavnog *threada*
  - *Media thread*
    - sinhronizacija *frameova* sa glavnim *threadom*
- *Pulse*
  - puls koji okida sinhronizaciju (*60 fps*) stanje na sučelju sa *prismom*
- *Preloader*
  - opcionalna komponenta za pospješivanje učitavanja aplikacije, starta se prije same aplikacije u svrhu prilagodbe (*applet, webstart...*)

Prva JavaFX aplikacija

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
                BorderPane root = new BorderPane();
                Scene scene = new Scene(root,400,400);
                scene.getStylesheets().add(getClass().getResource(
                    "application.css").toExternalForm());
                primaryStage.setScene(scene);
                primaryStage.show();
        } catch(Exception e) {
            e.printStackTrace();
        }
    }

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



Prva JavaFX aplikacija

- *javafx.application.Application* – apstraktna, dizajnirana za nasljeđivanje
  - upravljanje životnim ciklusom aplikacije, odvojeno od glavnog *threada*
- *pokretanje aplikacije:*
  1. ***launch(String[] args)*** – iz *stacktracea* pronalazi konkretnu *Application* implementaciju, refleksijom ju inicijalizira prosljeđuje:
  2. ***launchApplication(Class appClass, String[] args)*** – pronalazi *preloaderClass* koji se koristi za postavljanje okruženja učitavanja i poziva:
  3. ***launchApplication(Class appClass, Class preloaderClass, String[] args)*** – kreira *launcher thread i* pomoću *CountDown latch* čeka da *launcher* završi i prosljeđuje poziv:
  4. **launchApplication1(Class ppClass preloaderClass, String[] args)** – poziva *startToolkit()* za pokretanje platforme i potom *init* na *preloaderClass*. Napokon priprema *Stage* koji prosljedjuje u *application threadu,* našoj premošćenoj apstraktnoj metodi
  5. **start(Stage primaryStage)** – kreiramo sučelje (ručno ili fxml) sa root kontejnerom, kojim izgrađujemo *Scene* objekt i postavljamo ga na *Stage* – jedan *Stage* može sadržavati više *Scene* objekata

Prva JavaFX aplikacija

- prilagodba *Scene* objekta
  - konstruktorom prima root kontejner, *width* i *height* scene
  - korištenje CSS
    - direktno

    root.setStyle("-fx-background: rgb(225, 228, 203)");
    - zasebna datoteka – potrebno staviti u *package*
      - iz koda

      scene.getStylesheets()
      .add(getClass().getResource("css/application.css").toExternalForm());
      - iz fxml

      stylesheets="@css/application.css"

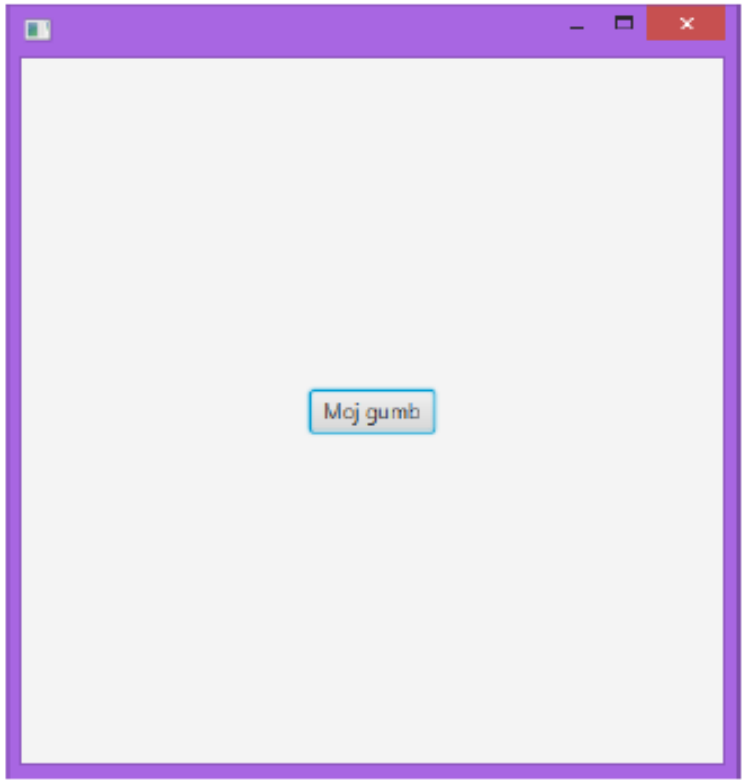
FXML

- promovira **S**eparation **O**f **C**oncerns – bitno odvajanje prezentacijske i poslovne logike aplikacije
- *Scene Builder* - omogućava dizajniranje grafičkog sučelja aplikacije korištenjem posebno oblikovane XML datoteke koja ima ekstenziju „.fxml” –*FX Markup Language*
- mora se uključiti na grafičkom sučelju aplikacije

```
@Override
public void start(Stage stage) throws Exception {
    Parent root = FXMLLoader.load(getClass().getResource("FXMLDocument.fxml"));
    Scene scene = new Scene(root);
    stage.setScene(scene);
    stage.show();
}
```

FXML

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.scene.layout.BorderPane?>
<BorderPane prefHeight="287.0" prefWidth="286.0"
xmlns:fx="http://javafx.com/fxml/1"
xmlns="http://javafx.com/javafx/8"
fx:controller="application.SampleController">
   <center>
      <Button mnemonicParsing="false" text="Moj gumb"
       BorderPane.alignment="CENTER" />
   </center>
</BorderPane>
```



Layouts

- organiziranje rasporeda elemenata korisničkog sučelja
- dinamičko povećanje prostora za prikazivanje komponenti u slučaju povećanja prozora na kojem se prikazuju
- najčešći

  javafx.scene.layout.HBox

  javafx.scene.layout.VBox

  javafx.scene.layout.FlowPane

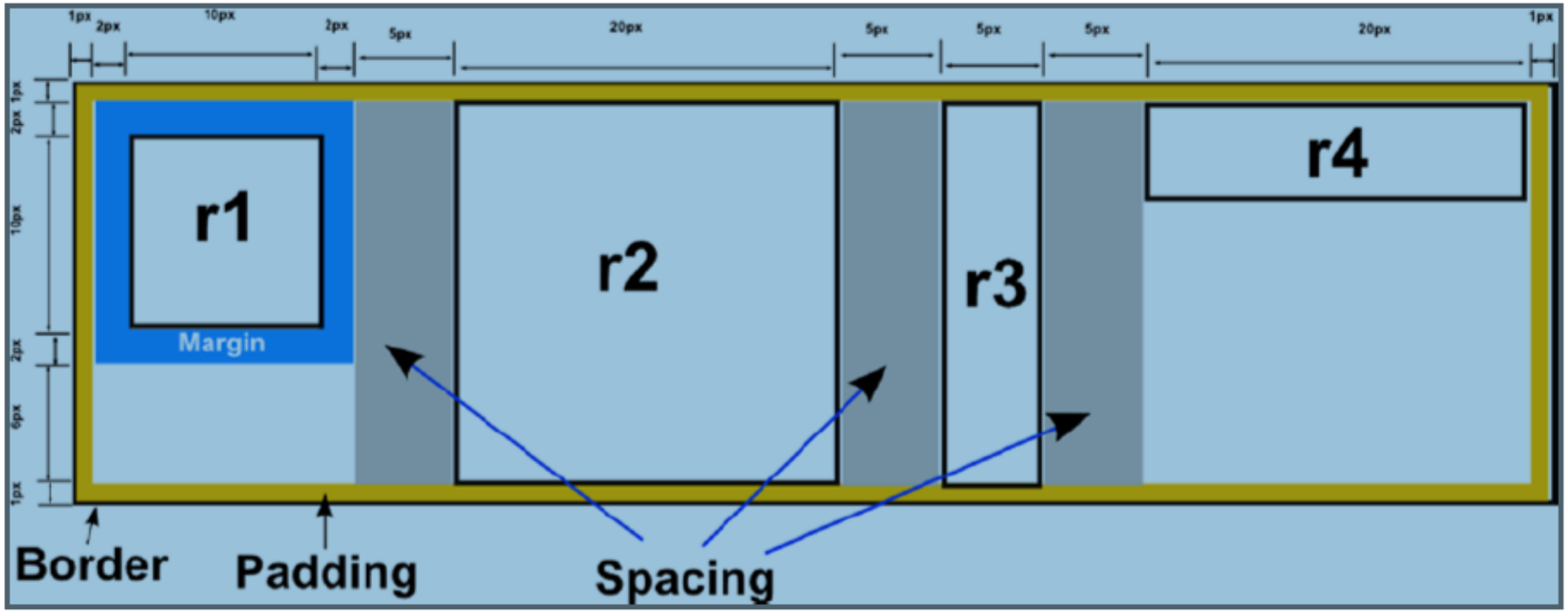
  javafx.scene.layout.BorderPane

Layouts - HBox

- postavlja elemente u horizontalni redak, jedan iza drugog
- pmogućava definiranje razmaka između komponenata koje se mogu mijenjati u ovisnosti o veličini komponente (prozora u kojem se nalaze)

```
HBox hbox = new HBox(5);
hbox.setPadding(new Insets(1));
List<Rectangle> rectangles = Arrays.asList(
        new Rectangle(100, 100),
        new Rectangle(200, 200),
        new Rectangle(50, 200),
        new Rectangle(200, 50));
rectangles.forEach(r ->
    HBox.setMargin(r, new Insets(10, 10, 10, 10)));
hbox.getChildren().addAll(rectangles);
```

Layouts - HBox

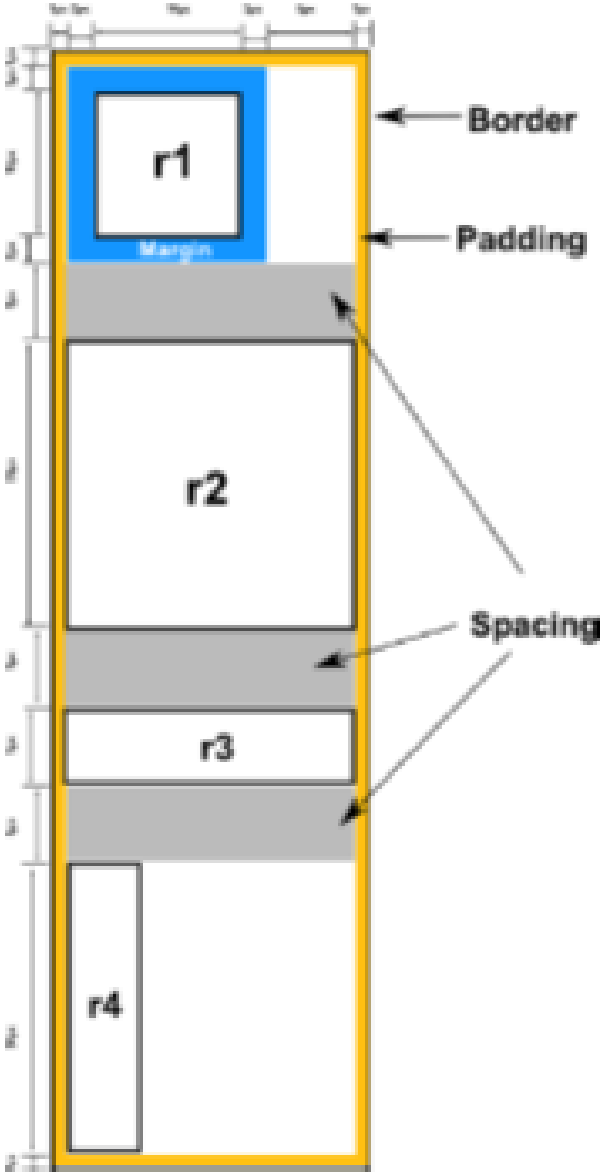


Layouts - VBox

- slično kao i HBox, ali postavlja elemente u stupac, jednog ispod drugog

```
VBox vbox = new VBox(5);
vbox.setPadding(new Insets(1));
 List<Rectangle> rectangles = Arrays.asList(
         new Rectangle(100, 100),
         new Rectangle(200, 200),
         new Rectangle(50, 200),
         new Rectangle(200, 50));
rectangles.forEach(r ->
    VBox.setMargin(r, new Insets(10, 10, 10, 10)));
vbox.getChildren().addAll(rectangles);
```

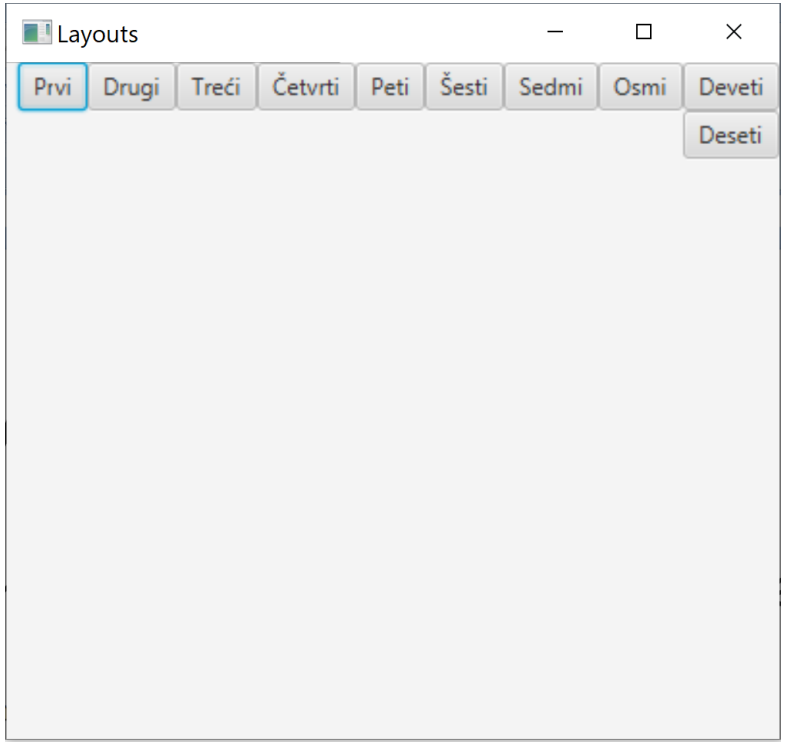
Layouts - VBox



Layouts - FlowPane

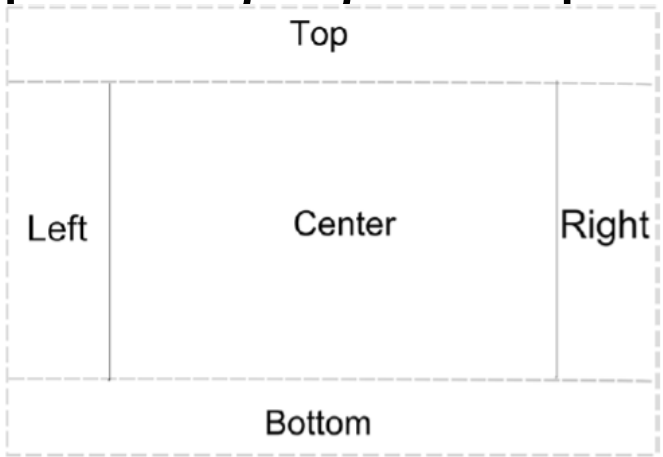
- omogućava postavljanje komponenti jedne iza druge do kraja raspoloživog retka, nakon čega nastavlja u sljedećem retku

```
FlowPane flowPane = new FlowPane();
List<Button> buttons = Arrays.asList(
    new Button("Prvi"),
    new Button("Drugi"),
    new Button("Treći"),
    new Button("Četvrti"),
    new Button("Peti"),
    new Button("Šesti"),
    new Button("Sedmi"),
    new Button("Osmi"),
    new Button("Deveti"),
    new Button("Deseti"));
flowPane.setAlignment(Pos.TOP_RIGHT);
flowPane.getChildren().addAll(buttons);
```



Layouts - BorderPane

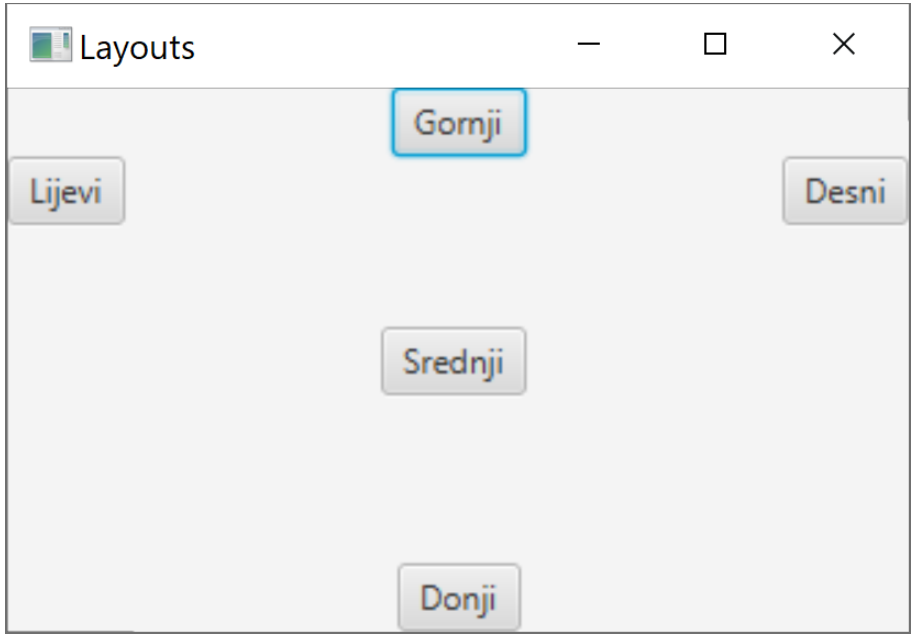
- omogućava postavljanje komponenti u pet različitih područja:
  - TOP
  - LEFT
  - CENTER
  - RIGHT
  - BOTTOM



- iako svako područje može imati samo jednu komponentu, moguće je u nju ubaciti drugi ugniježđeni organizator rasporeda komponenti
- mogu se koristiti koncepti kao i kod dizajniranja web stranica

Layouts - BorderPane

```
BorderPane borderPane = new BorderPane();
List<Button> buttons = Arrays.asList(
        new Button("Gornji"),
        new Button("Lijevi"),
        new Button("Srednji"),
        new Button("Desni"),
        new Button("Donji"));
BorderPane.setAlignment(buttons.get(0), Pos.TOP_CENTER);
borderPane.setTop(buttons.get(0));
borderPane.setLeft(buttons.get(1));
borderPane.setCenter(buttons.get(2));
borderPane.setRight(buttons.get(3));
BorderPane.setAlignment(buttons.get(4), Pos.BOTTOM_CENTER);
borderPane.setBottom(buttons.get(4));
```

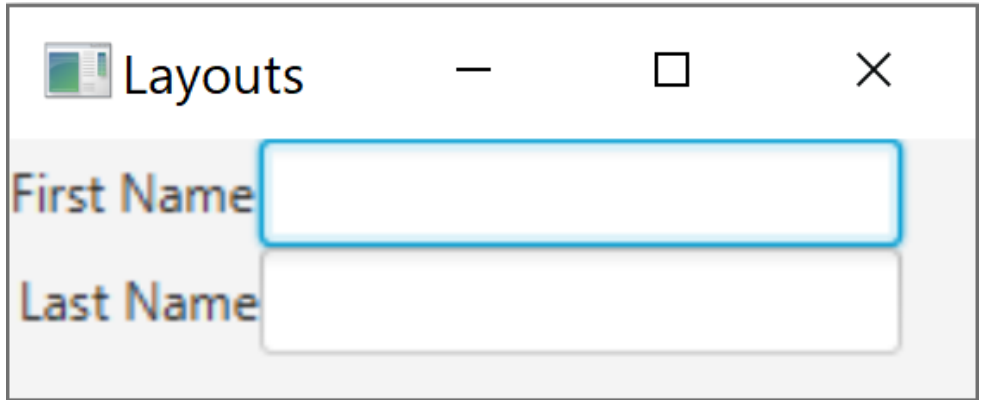


Layouts - GridPane

- omogućava umetanje elemenata na grafičko sučelje korištenjem „tablice” koja ima određen broj redaka i stupaca
- kod dodavanja elementa na grafičko sučelje je potrebno definirati „koordinate” ćelije tablice u koju se dodaje
- svakom stupcu moguće je definirati minimalnu, maksimalnu širinu u slučaju da se mijenja veličina samog ekrana u kojem se komponente nalaze
- prikladan za kreiranje ekrana koji sadrže „forme” za unos podataka

Layouts - GridPane

```
GridPane gridpane= new GridPane();
List<Label> labels = Arrays.asList(
        new Label("First Name"),
        new Label("Last Name"));
List<TextField> textFields = Arrays.asList(
        new TextField(),
        new TextField());
GridPane.setHalignment(labels.get(0), HPos.RIGHT);
gridpane.add(labels.get(0), 0, 0);
GridPane.setHalignment(labels.get(1), HPos.RIGHT);
gridpane.add(labels.get(1), 0, 1);
GridPane.setHalignment(textFields.get(0), HPos.LEFT);
gridpane.add(textFields.get(0), 1, 0);
GridPane.setHalignment(textFields.get(1), HPos.LEFT);
gridpane.add(textFields.get(1), 1, 1);
```



Demo

- Project

Izvor:http://www.jnhsolutions.com/contact-us/request-a-demo/

Hvala na pažnji!