1 Grafički korisnički interfejs. Paketi java.awt i javax.swing

Grafički korisnički interfejs (GUI, Graphical User Interface) je interfejs između korisnika i aplikacije baziran na grafičkim komponentama.
Ranije su se za pravljenje osnovnih elemenata grafičkog korisničkog interfejsa koristili paketi **java.awt** i **javax.swing.**

Paket java.awt

Paket `java.awt` (`awt` je skraćenica od Abstract Window Toolkit) je bio osnovni paket za pravljenje GUI-ja do pojave paketa `javax.swing` sa verzijom Java 2.
Većina klasa iz `awt` paketa zamenjena je klasama iz swing paketa.

Paket javax.swing

Najveći broj klasa iz paketa `javax.swing` definiše GUI elemente, poznate kao *Swing komponente*, koje su unapređena alternativa komponentama definisanim u `java.awt` paketu.
`Swing` komponente su fleksibilne jer su u celosti implementirane u Javi, dok `awt` komponente zavise i od izvornog koda. Iz tog razloga, `Swing` komponente nisu ograničene karakteristikama platforme na kojoj se izvršavaju.
Dodatno, komponente imaju svojstvo *pluggable-look-and-feel*, koje podrazumeva mogućnost promene izgleda komponente.

2 JavaFX

JavaFX, čist Java API, je naredna generacija GUI alata za razvoj RIA (*Ritch Internet Applications*) aplikacija.
JavaFX koristi paradigmu označenu kao ***scene graph,*** koja podrazumeva drvoliku strukturu podataka, gde su čvorovi grafova predstavljeni vektorima.
JavaFX scene graph je ***retained mode API***, što znači da održava interni model svih grafičkih objekata aplikacije. U svakom trenutku, zna se koji objekti treba da budu prikazani, koji delovi ekrana treba ponovo da se iscrtaju i sve se to prikazuje na najefikasniji način. Dakle, umesto da se direktno pozivaju metodi za crtanje, scene graph API prepušta sistemu da se stara o detaljima prikaza. Time se smanjuje i obim samog koda.

JavaFX je Java biblioteka predviđena za upotrebu na različitim uređajima, kao što su mobilni telefoni, pametni telefoni, računari, tablet računari itd. i uključuje rad sa audio i video sadržajem, animacijom i grafikom.

Ako Eclipse koristi Java 8 standardnu biblioteku, u okviru Java projekta je implicitno omogućena upotreba klasa iz paketa javafx, jer standardna biblioteka uključuje datoteku jfxrt.jar.
U praksi se takođe primenjuje i pravljenje JavaFX projekta umesto Java projekta
(New – Other... – JavaFX – JavaFX Project).

3 javafx.application.Application

JavaFX aplikacije nasleđuju apstraktnu klasu **`javafx.application.Application`** koja omogućuje pokretanje i zaustavljanje aplikacija, kao i bezbedno pokretanje u višenitnom okruženju.

U metodu **main()** poziva se metod **Application.launch()** za pokretanje JavaFX aplikacije i prosleđuju mu se eventualni argumenti komandne linije.
Interno se, od strane JavaFX okruženja, izvršavaju sledeće akcije nakon pokretanja aplikacije:

- pravi se instanca klase koja predstavlja aplikaciju
- poziva se metod `init()`
- poziva se metod `start(javafx.stage.Stage)`
- čeka se da se aplikacija završi
  - aplikacija poziva metod `Platform.exit()`
  - poslednji prozor je zatvoren i atribut `implicitExit` za `Platform` je postavljen na true
- poziva se metod `stop()`

Metod `start()` je apstraktan i mora da bude predefinisan.
Metodi `init()` i `stop()` nisu apstraktni, ali su definisani tako da ne rade ništa i ne treba da se predefinišu.

JavaFX pravi posebnu nit (**JavaFX Application Thread**) u okviru koje se pokreće `start` metod, procesiraju događaji ili pokreću animacije. Pravljenje JavaFX `Scene` ili `Stage` objekata, kao i izmena operacija nad objektima koji su deo scene, MORA da se vrši u okviru ove niti.

Poziv konstruktora za aplikaciju (pravljenje instance klase) i metoda `init` vrši se u drugoj niti (**launcher thread**). Iz tog razloga aplikacija NE SME da pravi objekte klasa `Scene` ili `Stage` u okviru konstruktora ili metoda `init`. Dozvoljeno je da se u metodu `init` prave drugi JavaFX objekti.

Detalji se mogu pogledati na lokaciji:
https://docs.oracle.com/javafx/2/api/javafx/application/Application.html

4 javafx.stage.Stage

Nakon poziva metoda **start(),** na raspolaganju je JavaFX objekat **Stage (javafx.stage.Stage)** za dalje rukovanje.
Kreatori JavaFX API-ja su organizovali prikaz grafičkog korisničkog interfejsa slično pozorištu, gde se predstava igra na bini ispred publike.

Zbog toga se na jednoj bini (**Stage**), može postaviti veći broj scena **Scene (javafx.scene.Scene)**, na kojima se odigravaju „predstave“.
Objekat *`Stage`* iz JavaFX je ekvivalentan prozorima aplikacije (`JFrame` ili `JDialog`) koji su se koristili u Java Swing API.
Osnovna pozornica (primaryStage) pravi se od strane platforme. Sama aplikacija može da pravi i dodatne pozornice (u okviru aplikativne niti).

Stilovi pozornice

- `StageStyle.DECORATED` – bela pozadina, sa dekoracijom (okvir, ikone u gornjem desnom uglu za zatvaranje prozora, minimizaciju i maksimizaciju, ikona u gornjem levom uglu) koja zavisi od platforme
- `StageStyle.UNDECORATED` – bela pozadina, bez dekoracije
- `StageStyle.TRANSPARENT` – transparentna pozadina, bez dekoracije

- `StageStyle.UTILITY` – bela pozadina i minimalna dekoracija (okvir i ikona za zatvaranje prozora)

Stil mora biti postavljen pre prikazivanja pozornice.

Detalji se mogu pogledati na lokaciji:
https://docs.oracle.com/javafx/2/api/javafx/stage/Stage.html

5 javafx.scene.Scene

Pozornica (bina) JavaFX aplikacije sadrži jednu ili više scena.
Objekat klase **Scene** može da se posmatra kao okvir za sadržaj (content pane).
Sadržaj scene je predstavljen kao graf sa proizvoljnim brojem čvorova – objekata tipa **Node (javafx.scene.Node)**.

`Node` je osnovna klasa čvorova grafa scene. Čvorovi mogu da budu: **UI kontrole** (dugmad, labele, tekstualna polja), **`Shape` objekti**, **slike** i drugo, koji zajedno čine korisnički interfejs.

Graf scene je drvolika struktura. Aplikacija mora da zada koreni čvor (rootNode) grafa scene postavljanjem parametra **root**.
Ako se kao koreni čvor koristi objekat klase `Group`, raspored objekata na sceni neće biti narušen promenom veličine scene (npr kada korisnik promeni veličinu bine).
Ako se kao koreni čvor postavi objekat koji je „resizable“ (npr `layout` čvor), njegova veličina se menja sa promenom veličine scene.

Koreni čvor ne može biti null. Veličina scene može da se zada prilikom pravljenja scene. U suprotnom, veličina scene se prilagođava samom sadržaju. Scene se prave u okviru aplikativne niti. Nakon dodavanja čvorova, scena se postavlja na binu i poziva se metod **show()** za prikaz bine (prozora aplikacije).

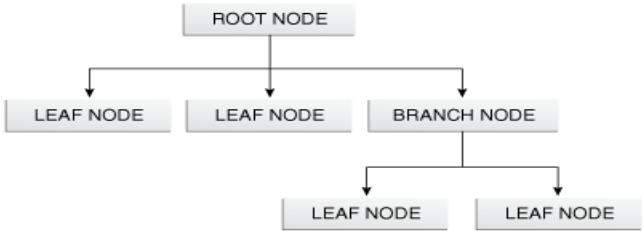
Atributi (definisani kao public final):

- `cursor` – definiše izgled kursora za scenu. Metodi: `getCursor()`, `setCursor(Cursor)`
- `fill` – definiše popunjavanje pozadine scene. Ako je `null`, prva mogućnost je da se ništa ne iscrtava na pozadini, a druga da je podržano transparentno iscrtavanje, ali šta će se tačno prikazati, zavisi od platforme. Podrazumevana vrednost je `BELA` boja. Metodi: `getFill()`, `setFill(Paint)`
- `height` – visina scene. Metod: `getHeight()`
- `root` – definiše koreni čvor grafa scene. Metodi: `getRoot()`, `setRoot(Parent)`
- `width` – širina scene. Metod: `getWidth()`
- `window` – prozor (Window) koji sadrži scenu. Metod: `getWindow()`
- `x` – horizontalna lokacija scene na prozoru. Metod: `getX()`
- `y` – vertikalna lokacija scene na prozoru. Metod: `getY()`
- …

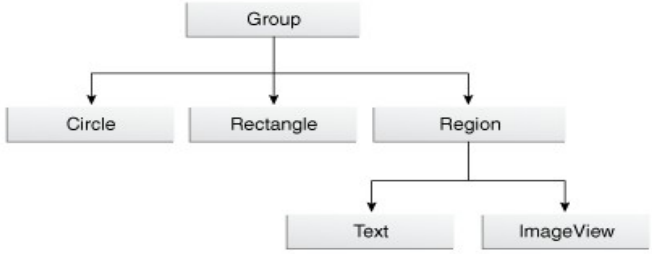
Detalji se mogu pogledati na lokaciji:
https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html

6 javafx.scene.Node

Bazna klasa za predstavljanje čvorova grafa scene.
Graf scene je drvoliki skup podataka gde postoji jedinstven čvor koji nema roditelja, koreni čvor (*root*). Preostali čvorovi imaju jednog roditelja i svaki čvor je ili „list“ (*leaf node*) (bez dece) ili „grana“ (*branch node*) (ima barem jedno dete).



JavaFX API definiše klase koje opisuju objekte koji se mogu ponašati kao koreni čvorovi, listovi ili grane. Primer upotrebe konkretnih klasa za odgovarajuće čvorove prikazan je na narednoj slici.



Svaki čvor je tipa Node ili neke njene potklase.

Čvorovi-grane su specifično tipa **javafx.scene.Parent** čije su potklase **Group**, **Region** (definiše deo ekrana na čiju se decu mogu primeniti stilovi pomoću CSS-a) i **Control**.

Čvorovi-listovi su specifično tipa **Rectangle**, **Circle**, **Text**, **ImageView**, **MediaView** ili bilo koje druge klase koja opisuje čvor koji ne može da ima dece.

Čvor može da se javlja najviše jednom bilo gde u grafu.
Graf scene ne sme da sadrži cikluse, tj. jedan čvor ne sme sam sebi da bude roditelj.

Struktura grafa može da se menja – na primer, poddrvo može da se premesti na drugu lokaciju u grafu. Da bi se ovo postiglo, dovoljno je samo umetnuti poddrvo na novu lokaciju, automatski će se izvršiti uklanjanje sa stare.
Čvorovi mogu da se prave i modifikuju u bilo kojoj niti pre nego što budu postavljeni na scenu. Postavljanje čvorova na scenu ili izmena postojećih čvorova na sceni, vrši se u okviru niti JavaFX Application.

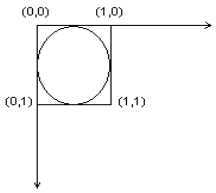
Stirng ID za čvor

Svakom čvoru u grafu može biti dodeljena jedinstvena identifikacija tipa String (`id`). Programer je taj koji se stara da dodeljeni `id` bude jedinstven u grafu. Metod **lookup(String)** može da se koristi za pronalaženje čvora sa datom identifikacijom u okviru grafa ili u okviru poddrveta grafa. ID čvora je sličan atributu „id“ HTML taga.

Koordinatni sistem

Koordinatni sistem je standardno postavljen – vrednosti duž x-ose rastu sleva udesno, a vrednosti duž y-ose odozgo naniže.
Konkretna klasa koja opisuje određenu vrstu čvora ima atribute za opis dimenzije i lokacije čvora. Na primer, klasa `Rectangle` ima atribute `x, y, width i height`.

Kada se posmatra na nivou piksela, celobrojne koordinate odgovaraju uglovima i postavljene su između piksela. Centar svakog piksela se onda nalazi na sredini rastojanja između dve susedne celobrojne koordinate.
Vrednosti koordinata zadate su kao realni brojevi, tako da mogu da se odnose na bilo koju poziciju u okviru piksela. Videti narednu sliku.



Klasa `javafx.scene.shape.Shape` opisuje dodatne infromacije vezane za mapiranje koordinata.

Transformacije čvorova

Nad bilo kojim čvorom mogu se primeniti transformacije: translacija, rotacija, skaliranje ili odsecanje.

Detalji se mogu pogledati na lokaciji:
http://docs.oracle.com/javafx/2/api/javafx/scene/Node.html

7 javafx.scene.Parent

Bazna klasa (apstraktna) za sve čvorove grafa scene koji mogu da imaju decu.
Ima tri direktne potklase:

- **Group** – kolekcija čvorova dece za primenu zajedničkih efekata i transformacija
- **Region** – površina ekrana koja može da sadrži i druge čvorove i da bude stilizovana CSS-om
- **Control** – klasa za čvorove namenjene interakciji korisnika

Detalji se mogu pogledati na lokaciji:
http://docs.oracle.com/javafx/2/api/javafx/scene/Parent.html

8 javafx.scene.Group

Čvor, objekat klase Group, sadrži listu (`ObservableList`) čvorova-dece.
Veličina grupe ne može direktno da se menja (grupa preuzima zajedničke granice shodno čvorovima-deci koje sadrži). Bilo koja transformacija primenjena na grupu, primenjuje se na svu decu.

9 javafx.scene.paint.Color

sRGB model boja

Boja se definiše kao kombinacija tri osnovne boje: crvene, zelene i plave. To su tzv. **RGB** vrednosti za boje.

Standardni konstruktor

```
Color(double red, double green, double blue, double opacity)
```

`red, green, blue, opacity` pripadaju intervalu [0.0 – 1.0].
`opacity` predstavlja stepen neprozirnosti (0.0 – boja je potpuno transparentna, 1.0 – boja je potpuno neprozirna).

Metod Color.color()

```
Color c = Color.color(0,0,1.0); // plava, opacity je implicitno 1.0
Color c = Color.color(0,0,1.0,1.0); // plava, opacity je eksplicitno 1.0
```

Metod Color.rgb()

`red, green, blue` uzimaju vrednosti iz intervala [0 – 255].

```
Color boja = Color.rgb(0,0,0); // crna
Color boja = Color.rgb(255,255,255); // bela
```

HSB model boja

`Color.hsb(double hue, double saturation, double brightness)`
`Color.hsb(double hue, double saturation, double brightness, double opacity)`

Nijansa (*hue*) je izražena u stepenima. Zasićenost (*saturation*), osvetljenost (*brightness*) i neprozirnost (*opacity*) uzimaju vrednosti iz intervala [0.0 – 1.0].

```
Color boja = Color.hsb(270,1.0,1.0);
Color boja = Color.hsb(270,1.0,1.0,1.0);
```

Alpha (opacity) vrednost boje

Svaka boja ima implicitnu **alpha** vrednost (***opacity***) 1.0 ili eksplicitnu vrednost zadatu u konstruktoru. Alpha vrednost definiše transparentnost boje i može da se predstavi dvostrukim vrednosnim opsezima [0.0 - 1.0] ili [0 - 255].

Alpha vrednost 1.0 ili 255 znači da je boja potpuno neprozirna, dok 0.0 ili 0 vrednosti označavaju potpuno transparentnu boju.

```
Color boja = new Color(0,0,1,0.7); // alpha ili opacity je 0.7
```

Drugi način za definisanje boje

Postoji mogućnost da se RGB boja odredi HTML ili CSS string atributom:

`web(java.lang.String colorString)`
`web(java.lang.String colorString, double opacity)`

```
Color boja = Color.web("0x0000FF", 1.0);
// plava, hex web vrednost, eksplicitno zadato alpha
```

Standardne boje

Klasa *Color* definiše i *public final static* promenljive za standardne boje:

```
WHITE (255,255,255)
RED (255,0,0)
GREEN (0,255,0)
BLUE (0,0,255)
BLACK (0,0,0)
MAGENTA (255,0,255)
CYAN (0,255,255)
PINK (255,175,175)
LIGHTGRAY (192,192,192)
ORANGE (255,200,0)
GRAY (128,128,128)
YELLOW (255,255,0)
DARKGRAY (64,64,64)
...
```

Podešavanje intenziteta boje

Za kreirani Color objekat, intenzitet prikazane boje može se podesiti metodima:
- **`brighter()`** - uvećava intenzitet komponenti boje za predefinisani faktor
- **`darker()`** - smanjuje intenzitet komponenti boje za predefinisani faktor

Bitniji metodi klase Color

`getRed()`, `getGreen()`, `getBlue()` – vraćaju komponente boje
`equals()` – za poređenje boja na jednakost komponenti

`deriveColor(double hueShift, double saturationFactor, double brightnessFactor, double opacityFactor)` – pravi novu boju na osnovu postojeće tako što se nijansa pomera za datu vrednost, dok se ostale komponente množe datim vrednostima i potom normalizuju u svom dozvoljenom opsegu.

`desaturate()` - pravi novu boju koja je manje zasićena od tekuće
`saturate()` - pravi novu boju više zasićenu od tekuće
`invert()` - pravi novu boju koja je inverzija tekuće boje

`valueOf(String)` – pravi boju na osnovu String-reprezentacije
`toString()` - vraća String-reprezentaciju boje (koristi se samo u informativne svrhe). Format stringa može da varira.

Za dodatne informacije pogledati:
http://docs.oracle.com/javafx/2/api/javafx/scene/paint/Color.html

10 javafx.scene.Cursor

Klasa `Cursor` definiše static konstante koje određuju standardne tipove kursora. Na primer:

`CLOSED_HAND`	`HAND`
`CROSSHAIR`	`TEXT`
`DEFAULT`	`WAIT`

Metodi

`static Cursor cursor(String identifier)` – vraća kursor za dati identifikator, koji može biti ime standardnog kursora ili validan URL string. U prvom slučaju vraća se odgovarajući standardni kursor, a u drugom objekat `ImageCursor` kreiran za dati URL.
`String toString()` - vraća String-reprezentaciju kursora

Detalji se mogu pogledati na lokaciji:
http://docs.oracle.com/javafx/2/api/javafx/scene/Cursor.html

11 javafx.scene.text.Font

Klasa ***Font*** je obimna, navodimo samo najbitnija svojstva. Font pravi razliku između karaktera i glifa, tj. grafičke reprezentacije karaktera. Različiti fontovi definišu različite glifove za jedan isti karakter. Veličina fonta zadaje se brojem piksela (1 piksel = 1/72 inča).

Konstruktori

`Font(double size)` – font zadate veličine predefinisanog izgleda „System“
`Font(java.lang.String name, double size)` – font zadate veličine i zadatog izgleda (*face*); navodi se puno ime za izgled fonta

Metodi

`font(java.lang.String family, double size)` – vraća odgovarajući font na osnovu date familije i veličine

`font(java.lang.String family, FontPosture posture, double size)` – vraća odgovarajući font na osnovu date familije i datog položaja (da li je font iskošen ili regularan). `FontPosture` je definisan kao tip enumeracije (`ITALIC, REGULAR`)

`font(java.lang.String family, FontWeight weight, double size)` – „težina“ fonta je predstavljena tipom enumeracije `FontWeight` (`BLACK, BOLD, EXTRA_BOLD, EXTRA_LIGHT, LIGHT, MEDIUM, NORMAL, SEMI_BOLD, THIN`)

`font(java.lang.String family, FontPosture posture, FontWeight weight, double size)`

`getDefault()` – vraća podrazumevani font, koji je obično iz familije „System“, stil je najčešće „Regular“, a veličina je saglasna sa okruženjem

`getFamilies()` – vraća listu imena svih familija fontova instaliranih na datoj platformi
`getFontNames()` – vraća listu imena svih fontova instaliranih na datoj platformi
`getFontNames(String family)` – vraća listu imena svih fontova iz date familije fontova koji su podržani na platformi
`getName()` – vraća puno ime fonta
`getFamily()` – vraća familiju tekućeg fonta
`getStyle()` – vraća string koji opisuje stil fonta u okviru familije
`getSize()` – vraća veličinu fonta

Detalji se mogu pogledati na lokaciji:
http://docs.oracle.com/javafx/2/api/javafx/scene/text/Font.html

12 Raspoređivanje elemenata scene (layout panes)

Layout pane za scenu određuje poziciju i veličinu svih čvorova na njoj i njime se zadaje izgled grafičkog korisničkog interfejsa.
Kako se menja veličina prozora, layout pane automatski odgovara i menja veličinu čvorova koje sadrži u skladu sa njihovim osobinama.

JavaFX sadrži niz layout klasa, koje se nalaze u paketu: **`javafx.scene.layout:`**

- **`AnchorPane, BorderPane, StackPane`**
- **`Hbox, Vbox, FlowPane`**
- **`TilePane, GridPane`**

AnchorPane
Čvorovi deca se sidre po ivicama roditelja. Veličina čvorova dece ne može da se menja.

BorderPane
Površina roditelja se deli na 5 oblasti – vrh, levo, desno, dno, centar. Oblasti vrh i dno prostiru se duž cele širine, a visina odgovara zadatoj visini čvorova dece. Oblasti levo i desno prostiru se duž preostale visine, a širina odgovara zadatoj širini čvorova dece. Centralna oblast obuhvata preostali prostor u sredini. Bilo koja oblast može biti null. Sva povećanja veličina vrše se do maksimalne veličine čvora u relevantnom smeru.

StackPane
Čvorovi deca se raspoređuju jedan iznad drugog, kao špil karata. Jedino je komponenta na "vrhu" vidljiva u bilo kom trenutku. Veličina čvorova dece se menja, tako da ispuni veličinu roditelja, uz poštovanje zadate maksimalne širine, odnosno visine za svaki čvor.

Hbox
Čvorovi deca se raspoređuju u jedan red. Veličina čvorova dece se povećava do njihove željene širine (*preferred width*), ali postoji mogućnost da se zada da se pojedini čvorovi deca povećavaju do njihove maksimalne zadate širine.

Vbox
sličan HBox-u, jedino što se čvorovi deca raspoređuju u jednu kolonu.

Padding property upravlja rastojanjem između čvorova. *Margine* takođe mogu da se podese da bi se odredila veličina praznog prostora oko pojedinih čvorova.

FlowPane
Dodaju se komponente u sukcesivnim redovima – kad je red popunjen, počinje se sa novim. Najčešće se koristi za uređivanje dugmeta. Veličina čvorova dece ne može da se menja.

TilePane
Sličan kao `FlowPane`, jedino što se čvorovi deca smeštaju u mrežu u kojoj je svaka „pločica" jednake veličine. Čvorovi deca mogu da budu postavljeni horizontalno (u redovima) ili vertikalno (u kolonama). Veličina čvorova dece menja se, tako da ispuni veličinu „pločice", uz poštovanje zadate maksimalne širine i visine.

GridPane
Komponente se raspoređuju u fleksibilnu pravougaonu mrežu. Promena veličine zavisi od ograničenja zadatih za svaki red, odnosno kolonu, pojedinačno.

13 FlowPane

FlowPane smešta čvorove decu u red, a kada se red napuni, automatski počinje smeštanje u novi red.

Podrazumevana pozicija reda sa komponentama je centar kontejnera, a podrazumevana orijentacija je sleva udesno. Može se zadati podrazumevani razmak među komponentama.
Primer je prikazan na slici.

Konstruktori

`FlowPane()` – kreira horizontalni *FlowPane* sa horizontalnim i vertikalnim razmakom 0 (hgap/vgap = 0)

`FlowPane(double hgap, double vgap)` – kreira horizontalni *FlowPane* sa zadatim hgap i vgap

`FlowPane(Orientation orientation)` – kreira *FlowPane* zadate orijentacije sa horizontalnim i vertikalnim razmakom 0 (hgap/vgap = 0)

`FlowPane(Orientation orientation, double hgap, double vgap)` – kreira *FlowPane* zadate orijentacije sa zadatim hgap i vgap.

Primer horizontalnog FlowPane-a:

```
Image images[] = { ... };
FlowPane flow = new FlowPane();
flow.setVgap(8);
flow.setHgap(4);
flow.setPrefWrapLength(300); // zeljena sirina = 300
for (int i = 0; i < images.length; i++) {
  flow.getChildren().add(new ImageView(image[i]);
}
```

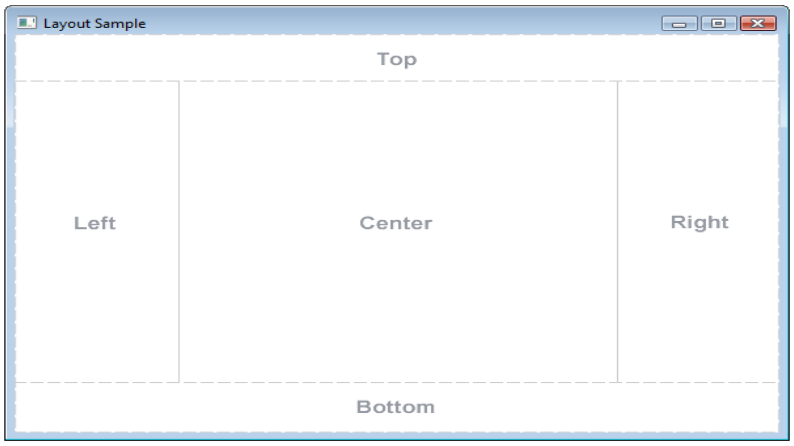
Primer vertikalnog FlowPane-a:

```
FlowPane flow = new FlowPane(Orientation.VERTICAL);
flow.setColumnHalignment(HPos.LEFT); // poravnanje elemenata: levo
flow.setPrefWrapLength(200); // zeljena visina = 200
for (int i = 0; i < titles.size(); i++) {
  flow.getChildren().add(new Label(titles[i]);
}
```

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/scene/layout/FlowPane.html

14 BorderPane

Komponente raspoređene u BorderPane se šire tako da popune raspoloživ prostor u kontejneru.
Bilo koja pozicija može da bude null. Pozadina i ivice BorderPane-a mogu da budu stilizovane CSS-om.



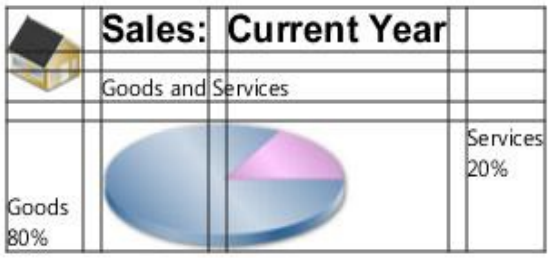
Primer

```
BorderPane borderpane = new BorderPane();
ToolBar toolbar = new ToolBar();
HBox statusbar = new HBox();
Node appContent = new AppContentNode();
borderPane.setTop(toolbar);
borderPane.setCenter(appContent);
borderPane.setBottom(statusbar);
```

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/scene/layout/BorderPane.html

15 GridPane

GridPane raspoređuje čvorove u fleksibilnu pravougaonu mrežu. Čvorovi mogu da budu smešteni u bilo koju ćeliju u mreži. Mreža je korisna za upotrebu kada se čvorovi raspoređuju u redove i kolone.



Primer:

```
GridPane gridpane = new GridPane();

// Postavljanje jedne po jedne osobine...
Button button = new Button();
// dugme se postavlja u red sa indeksom 1 (drugi red)...
GridPane.setRowIndex(button, 1);
// ...i kolonu sa indeksom 2 (treca kolona)
GridPane.setColumnIndex(button, 2);

// Postavljanje svih odjednom
Label label = new Label();
GridPane.setConstraints(label, 3, 1); // kolona = 3 red = 1
// obavezno je postaviti decu na GridPane
gridpane.getChildren().addAll(button, label);

// Moguce je ujedno odrediti pozicije i postaviti decu:
GridPane gridpane = new GridPane();
gridpane.add(new Button(), 2, 1); // kolona=2 red=1
gridpane.add(new Label(), 3, 1); // kolona=3 red=1
```

16 Paket javafx.geometry – hijerarhija klasa

Obezbeđuje skup 2D klasa za definisanje i obavljanje operacija nad objektima u dvodimenzionalnoj geometriji:

- javafx.geometry.**BoundingBoxBuilder<B>**(implements javafx.util.Builder<T>)
- javafx.geometry.**Bounds**
- javafx.geometry.**BoundingBox**
- javafx.geometry.**Dimension2D**
- javafx.geometry.**Dimension2DBuilder<B>** (implements javafx.util.Builder<T>)
- javafx.geometry.**Insets**
- javafx.geometry.**InsetsBuilder<B>** (implements javafx.util.Builder<T>)

- javafx.geometry.**Point2D**
- javafx.geometry.**Point2DBuilder<B>** (implements javafx.util.Builder<T>)
- javafx.geometry.**Point3D**
- javafx.geometry.**Point3DBuilder<B>** (implements javafx.util.Builder<T>)
- javafx.geometry.**Rectangle2D**
- javafx.geometry.**Rectangle2DBuilder<B>** (implements javafx.util.Builder<T>)

17 Paket javafx.geometry – hijerarhija enumeracija

- javafx.geometry.**VPos**
- javafx.geometry.**VerticalDirection**
- javafx.geometry.**Side**
- javafx.geometry.**Pos**
- javafx.geometry.**Orientation**
- javafx.geometry.**HPos**
- javafx.geometry.**HorizontalDirection**

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/geometry/package-summary.html

18 javafx.geometry.Pos

Skup vrednosti za opisivanje vertikalnog i horizontalnog pozicioniranja i poravnanja:

- BASELINE_CENTER, BASELINE_LEFT, BASELINE_RIGHT
- BOTTOM_CENTER, BOTTOM_LEFT, BOTTOM_RIGHT
- CENTER, CENTER_LEFT, CENTER_RIGHT
- TOP_CENTER, TOP_LEFT, TOP_RIGHT

Metodi

`HPos getHpos()` – vraća horizontalnu poziciju/poravnanje
`VPos getVpos()` – vraća vertikalnu poziciju/poravnanje
`static Pos valueOf(java.lang.String name)` – vraća enum konstantu zadatu imenom *name*
`static Pos[] values()` – vraća niz konstanti enumeracije, redom kojim su deklarisane

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/geometry/Pos.html

19 javafx.geometry.HPos

Skup vrednosti za opisivanje horizontalnog pozicioniranja i poravnavanja:
- `CENTER, LEFT, RIGHT`

Metodi:
```
static HPos valueOf(java.lang.String name)
static HPos[] values()
```

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/geometry/HPos.html

20 javafx.geometry.VPos

Skup vrednosti za opisivanje vertikalnog pozicioniranja i poravnavanja:
- `BASELINE, BOTTOM, CENTER, TOP`

Metodi:
```
static VPos valueOf(java.lang.String name)
static VPos[] values()
```

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/geometry/VPos.html

21 javafx.geometry.Orientation

Određivanje horizontalne i vertikalne orijentacije:
- HORIZONTAL (horizontalna „levo – desno“ ili „desno - levo“ orijentacija)
- VERTICAL (vertikalna „vrh – dno“ ili „dno – vrh“ orijentacija)

Metodi:
```
static Orientation valueOf(java.lang.String name)
static Orientation[] values()
```

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/geometry/Orientation.html

22 javafx.geometry.Insets

Skup rastojanja (pomaka) u odnosu na četiri strane pravougaone površine (kontejnerskog čvora).

Konstruktori:

`Insets(double topRightBottomLeft)` - formira se `Inset` objekat koji ima jednaku vrednost rastojanja za sva četiri razmaka

`Insets(double top, double right, double bottom, double left)` - zadaju se 4 različite vrednosti za rastojanja

Detalji na lokaciji: http://docs.oracle.com/javafx/2/api/javafx/geometry/Insets.html