Kombiniranje *Java Swing*-a i *JavaFX*-a, izrada cjelovite aplikacije s grafičkim sučeljem

Matej Mihelčić

Prirodoslovno-matematički fakultet, Sveučilište u Zagrebu

matmih@math.hr

25. siječnja, 2023.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

Koristimo `JFXPanel` iz paketa `javafx.embed.swing.JFXPanel` i dodajemo ga *Swing* formi. Na `JFXPanel` možemo staviti **proizvoljne** *JavaFX* komponente. Komunikacija između *Swing* komponenti i *JavaFX* komponenti se vrši pozivom `Platform.runLater` (iz *Swing*-a radimo nad *JavaFX* komponentama) i `SwingUtilities.invokeLater` (iz *JavaFX*-a radimo nad *Swing* komponentama). Dakle promjene nad ***Swing*** **komponentama vršimo glavnom *Swing* dretvom koja reagira na događaje dok sve promjene nad *JavaFX* komponentama moramo raditi *JavaFX* dretvom**.

Primjer 1

Napravite *Java Swing* aplikaciju koja prikazuje sliku te sadrži 4 gumba za manipulaciju slikom (rotacija ulijevo i udesno, povečavanje i smanjivanje slike). Slika se prikazuje pomoću *JavaFX*-a i odgovarajući *Swing* gumbi pozivaju akcije *JavaFX*-a za transformiranje slike.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

Stvaramo klasu `ProzorSaSlikom` koji nasljeđuje klasu `Frame`.

```
1  public class ProzorSaSlikom extends JFrame {
2      private final JFXPanel jfxPanel = new JFXPanel();
3      private final JPanel panel = new JPanel(new BorderLayout
       ());
4      private ImageView slika;
5      private final JButton btnrr = new JButton("Rotiraj
       udesno");
6      private final JButton btnrl = new JButton("Rotiraj
       ulijevo");
7      private final JButton btnscup = new JButton("Uvecaj");
8      private final JButton btnscdown = new JButton("Umanji");
9
10     ProzorSaSlikom(){ //konstruktor
11         super(); //kreiramo okvir
12         inicijalizirajJavaFX();
13         inicijalizirajSwing(); }
```

Korištenje JavaFX-a unutar Swing aplikacije.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

JavaFX komponente se postavljaju na `jfxPanel` unutar *JavaFX* dretve.

```
1  private void inicijalizirajJavaFX(){
2    Platform.runLater(new Runnable() {
3     @Override public void run() {
4                 AnchorPane ap = new AnchorPane();
5                 Scene s = new Scene(ap);
6                 slika = new ImageView();
7                 Image im = new Image("file:/Drive/Users/..
   Putanja.../Vjezbe9/lake.jpg");
8          slika.setImage(im); slika.setPreserveRatio(true);
9                 ap.getChildren().add(slika);
10                jfxPanel.setScene(s); } }); }
```

Korištenje JavaFX-a unutar Swing aplikacije.

Kao korijen postavljamo `AnchorPane` i na njega dodajemo sliku (`ImageView`). Na `jfxPanel` (koji služi za komunikaciju) dodajemo rezultantnu scenu.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1  private void inicijalizirajSwing(){
2      this.setTitle("Rad sa slikama u Swingu");
3      this.setSize(600,600);
4      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
5
6  //dodajemo funkcije za obradu aktivacije Swing gumbi
7      ActionListener rotirajdesno = new ActionListener() {
8        @Override public void actionPerformed(ActionEvent e) {
9                  Platform.runLater(new Runnable() {
10       @Override public void run() {
11                 slika.setRotate(slika.getRotate() + 90);  }
     }); } }; //obrada u JavaFX dretvi
12
13     ActionListener rotirajlijevo = new ActionListener() {
14       @Override public void actionPerformed(ActionEvent e) {
15                 Platform.runLater(new Runnable() {
16       @Override public void run() {
17                 slika.setRotate(slika.getRotate() - 90);  }
     }); } };
```

Korištenje JavaFX-a unutar Swing aplikacije.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1  ActionListener uvecaj = new ActionListener() {
2     @Override public void actionPerformed(ActionEvent e) {
3                  Platform.runLater(new Runnable() {
4     @Override   public void run() {
5                  slika.setScaleX(slika.getScaleX()*1.2);
6           slika.setScaleY(slika.getScaleY()*1.2); } }); }};
7
8  ActionListener umanji = new ActionListener() {
9    @Override public void actionPerformed(ActionEvent e) {
10                 Platform.runLater(new Runnable() {
11   @Override public void run() {
12                 slika.setScaleX(slika.getScaleX()*0.8);
13          slika.setScaleY(slika.getScaleY()*0.8); } }); } };
14
15       btnrr.addActionListener(rotirajdesno);
16       btnrl.addActionListener(rotirajlijevo);
17       btnscup.addActionListener(uvecaj);
18       btnscdown.addActionListener(umanji);
```

Korištenje JavaFX-a unutar Swing aplikacije.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1          JPanel gumbi = new JPanel(new GridLayout(1,4));
2          gumbi.add(btnrr);//dodajemo gumbe u novi swing Panel
3          gumbi.add(btnrl);//slozen u mrezu dimenzija 1x4
4          gumbi.add(btnscup);
5          gumbi.add(btnscdown);
6          panel.add(jfxPanel,BorderLayout.CENTER);//JavaFX
7          panel.add(gumbi, BorderLayout.SOUTH);
8          this.add(panel);
9      }
10
11  public static void main(String[] args) {
12 //glavni program, pozivamo glavnu Swing dretvu koja iscrtava
       graficko sucelje aplikacije
13         SwingUtilities.invokeLater(new Runnable() {
14         @Override public void run() {
15                 ProzorSaSlikom slika = new ProzorSaSlikom();
16                 slika.setVisible(true);
17             } }); }
```

Korištenje JavaFX-a unutar Swing aplikacije.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

Sliku je moguće staviti na formu i koristeći *Swing* iako su manipulacija i lijepo skaliranje malo kompliciraniji.

```
1  public class SlikaSwing extends Canvas{
2      public void paint(Graphics g) {
3          Toolkit t=Toolkit.getDefaultToolkit();
4          Image i=t.getImage("Drive:\\Users\\...Putanja...\\
       lake.jpg");
5          g.drawImage(i, 0,0,this); }
6          public static void main(String[] args) {
7          SlikaSwing m=new SlikaSwing();
8          JFrame f=new JFrame();
9          f.add(m);
10         f.setSize(400,400);
11         f.setVisible(true);  }  }
```

Iscrtavanje slike na formu koristeći Swing.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

Primjer 2

Kreirajte jednostavan web preglednik unutar *Java Swing* grafičkog sučelja. Za implementaciju preglednika ćemo koristiti *JavaFX*.

```
1  public class JednostavniPretrazivacInterneta extends JFrame
       {
2   private final JFXPanel jfxPanel = new JFXPanel();
3   private WebEngine engine;
4   private final JPanel panel= new JPanel(new BorderLayout());
5   private final JButton btnGo = new JButton("Idi");
6   private final JTextField txtURL = new JTextField();
7   private final JProgressBar progressBar= new JProgressBar();
8
9      public JednostavniPretrazivacInterneta() {
10         super();
11         inicijaliziraj(); }
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1  private void inicijaliziraj() {
2    inicijalizirajJavaFX();//funkcija za inicijalizaciju
     JavaFX-a
3     //slusac koji na akciju ucita zadani URL
4    ActionListener al = new ActionListener() {
5       @Override public void actionPerformed(ActionEvent e)
6                                {ucitajURL(txtURL.getText()); } };
7       //dodamo slusaca akcijama gumba i tekstualne forme
8        btnGo.addActionListener(al);
9        txtURL.addActionListener(al);
10       //inicijaliziramo komponentu koja indicira kolicinu
11       //izvrsenog zadatka (ucitani postotak stranice)
12       progressBar.setPreferredSize(new Dimension(150, 18));
13       progressBar.setStringPainted(true);
14 //konstruiramo JPanel s BorderLayout-om i horizontalnim
     razmakom izmedu komponenti 5 jedinica
15       JPanel topBar = new JPanel(new BorderLayout(5, 0));
16       topBar.setBorder(BorderFactory.createEmptyBorder(3,
     5, 3, 5)); //postavimo prazne granice oko spremnika
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1    topBar.add(txtURL, BorderLayout.CENTER); //dodamo txt
      formu
2    topBar.add(btnGo, BorderLayout.EAST); //dodamo gumb
3    //dodamo spremnik za komponentu statusa izvrsavanja
4    JPanel statusBar = new JPanel(new BorderLayout(5, 0));
5    statusBar.setBorder(BorderFactory.createEmptyBorder(3, 5,
       3, 5));
6    statusBar.add(progressBar, BorderLayout.EAST);
7  //dodamo spremnike na glavni spremnik
8    panel.add(topBar, BorderLayout.NORTH);
9    panel.add(jfxPanel, BorderLayout.CENTER);
10   panel.add(statusBar, BorderLayout.SOUTH);
11   getContentPane().add(panel); //dodamo na formu
12 //postavimo preferiranu velicinu prozora, akciju pri
      pritisku gumba zatvaranja i pozovemo funkciju
      postavljanja komponenti obzirom na preferiran velicinu
13   setPreferredSize(new Dimension(1024, 600));
14   setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
15   pack(); }
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1  private void inicijalizirajJavaFX() {
2  //definiramo niz slusaca JavaFX komponenti
3          Platform.runLater(new Runnable() {
4              @Override public void run() {
5                  WebView view = new WebView();
6                  engine = view.getEngine();
7                 //slusac promjena naslova web engine-a
8                  engine.titleProperty().addListener(
9                  new ChangeListener<String>() {
10                     @Override  public void changed(
     ObservableValue<? extends String> observable, String
     oldValue, final String newValue) {
11 //promjena naslova okvira obavezno u Swing glavnoj dretvi
12                         SwingUtilities.invokeLater(new
13                         Runnable() {
14                             @Override public void run(){
15                             JednostavniPretrazivacInterneta.
16                             this.setTitle(newValue);
17                             } }); } });
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1                   engine.locationProperty().addListener(new
2                   ChangeListener<String>() { //slusac URL-a
3                       @Override public void changed(
    ObservableValue<? extends String> ov, String oldValue,
    final String newValue) {
4                       SwingUtilities.invokeLater(new
5                        Runnable() {//txtURL je swing kontrola
6                         @Override public void run() {
7                            txtURL.setText(newValue); } }); }
    });//mijenjamo u Swing glavnoj dretvi
8               engine.getLoadWorker().workDoneProperty()
9               .addListener(new ChangeListener<Number>() {
10                 @Override public void changed(ObservableValue
11                 <? extends Number> observableValue, Number
12                 oldValue, final Number newValue) {
13                 SwingUtilities.invokeLater(new Runnable() {
14                   @Override public void run() {
15                    progressBar.setValue(newValue.intValue());
16               }  }); } }); //osluskujemo ucitavanje stranice
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1            engine.getLoadWorker().exceptionProperty()
2                .addListener(new ChangeListener<Throwable>() {
3                    @Override public void changed(
    ObservableValue<? extends Throwable> o, Throwable old,
    final Throwable value) {//osluskujemo greske preglednika
4                    if (engine.getLoadWorker().getState()
5                          == FAILED) {  SwingUtilities.
6                          invokeLater(new Runnable() {
7                          @Override public void run() {
8                          JOptionPane.showMessageDialog(panel,
9                         (value != null) ? engine.getLocation()
10                         + "\n" + value.getMessage() : engine.
11                         getLocation() + "\nGreska.", "Greska
12                         pri ucitavanju...", JOptionPane.
13                         ERROR_MESSAGE);} }); } } });
14 //kreiramo scenu koja ima preglednik kao korjen, pridruzimo
       scenu formi
15                   jfxPanel.setScene(new Scene(view)); } }); }
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *JavaFX*-a u *Java Swing* aplikaciju

```
1  public void ucitajURL(final String url) {//ucitava URL
2    Platform.runLater(new Runnable() { //u preglednik
3        @Override public void run() {
4           String tmp = toURL(url);
5           if (tmp == null) {  tmp = toURL("http://" + url); }
6              engine.load(tmp); }  }); }
7
8  private static String toURL(String str) {//pretvara string
9         try { return new URL(str).toExternalForm();// u URL
10        } catch (MalformedURLException exception) {
11           return null; } }
12
13 public static void main(String[] args) {
14     SwingUtilities.invokeLater(new Runnable() {
15         @Override public void run() {
16            JednostavniPretrazivacInterneta browser = new
17            JednostavniPretrazivacInterneta();
18            browser.setVisible(true);
19            browser.ucitajURL("http://oracle.com"); }  }); }
```

Kreiranje jednostavnog web preglednika unutar Swing forme.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

Glavna ideja korištenja *Swing* komponenti u *JavaFX* aplikacijama je **kreiranje posebnog čvora** `SwingNode` **definiranog u paketu** `javafx.embed.swing` **koji omogućava ugrađivanje *Swing* konteksta u *JavaFX* aplikaciju**. Sadržaj objekta `SwingNode` se postavlja metodom `setContent` koja radi nad instancama klase `javax.swing.JComponent`. **Metoda** `setContent` **se može pozvati ili na glavnoj dretvi *Java Swing*-a ili u *JavaFX* aplikacijskoj dretvi**. Manipulaciju *Swing* komponentama **moramo raditi u *Swing* glavnoj dretvi**.

Primjer 3

Kreirajte *JavaFX* aplikaciju koja ima *JavaFX* labelu pozicioniranu u gornjem dijelu prozora i *Swing* gumb pozicioniran u južnom dijelu prozora. Pritiskom na gumb se sadržaj labele promijeni u tekst *Gumb kliknut*. Ponovnim pritiskom se tekst labele prazni i tako ciklički dalje.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

```
1  public class Predavanje12JavaFX extends Application {
2   private Label l;
3   int flag=0;
4
5     @Override public void start (Stage stage) {
6         l = new Label();
7         final SwingNode swingNode = new SwingNode(); //cvor
      za ukljucivanje Swinga u JavaFX
8         swingNode.setCache(true); //omogucava bolje
      prikazivanje Swing sadrzaja
9         swingNode.setCacheHint(CacheHint.SPEED);
10        BorderPane pane = new BorderPane(); //postavimo
      spremnik s dobro definiranim granicama
11        pane.setBottom(swingNode); //Swing cvor na dnu
12        pane.setTop(l); //labela na vrhu
13        stage.setTitle("Swing u JavaFX-u");
14        stage.setScene(new Scene(pane, 250, 150));
15        stage.show();//obavezno prije inicijalizirajSwing
16        inicijalizirajSwing(swingNode); }
```

Kreiranje jednostavne JavaFX aplikacije sa Swing gumbom.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

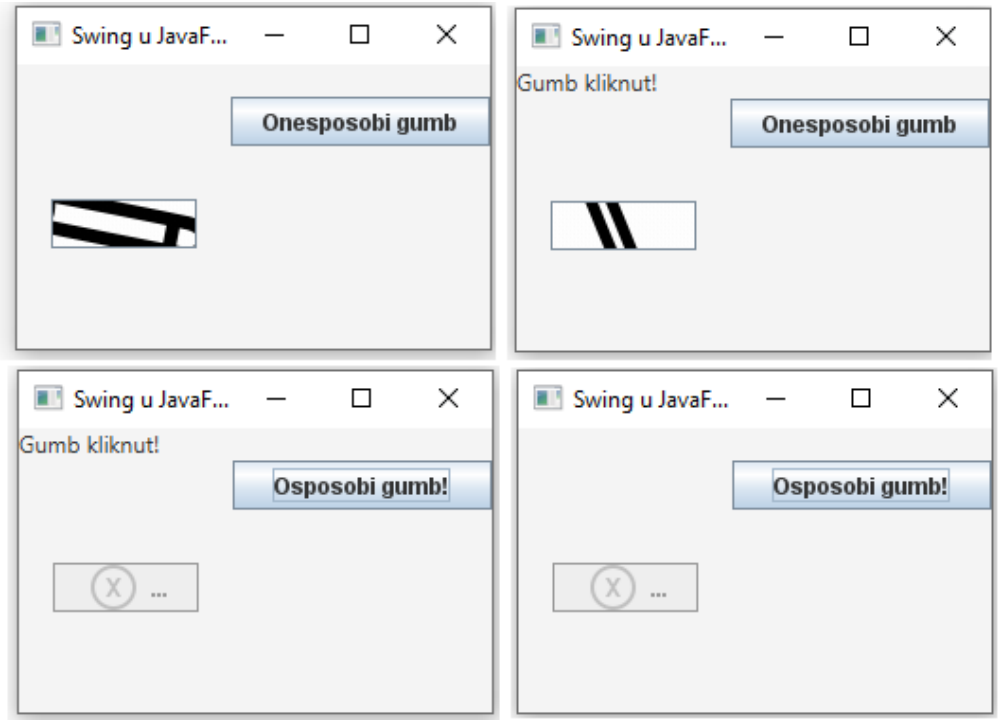
```
1   private void inicijalizirajSwing(final SwingNode
    swingNode) {
2       SwingUtilities.invokeLater(new Runnable() {
3           @Override public void run() {
4               JButton b = new JButton("Pritisni!");
5               swingNode.setContent(b);
6               b.addActionListener(new ActionListener(){
7   @Override public void actionPerformed(ActionEvent e){
8               Platform.runLater(new Runnable(){
9                   @Override public void run(){
10                      if(flag == 0){
11                          l.setText("Gumb kliknut!");
12                          flag = 1; }
13                      else{
14                          l.setText(""); flag = 0; }
15              } } ); }  }); } }); }
16
17  public static void main(String[] args) {
18      launch(args);  }     }
```

Kreiranje jednostavne JavaFX aplikacije sa Swing gumbom.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

Primjer 4

Izmjenite prethodni primjer tako da dodate još jedan *Swing* gumb *Onesposobi gumb* čija akcija je onesposobiti prvi gumb (može biti pozicioniran južno ili centralno uz dodanu `.gif` ikonu) ukoliko je osposobljen a osposobiti ga ukoliko je onesposobljen (tada sadrži `.png` x ikonu). Pri pritisku gumba, mijenja se ikonica drugoga gumba.



Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

Relevantni dijelovi koda su:

```
final SwingNode swingNode1 = new SwingNode();
swingNode1.setCache(true);
swingNode1.setCacheHint(CacheHint.SPEED);
pane.setRight(swingNode1);
//potrebno zbog ispravnog gasenja aplikacije
 stage.setOnCloseRequest(new EventHandler<WindowEvent>() {
        @Override public void handle(WindowEvent t) {
                Platform.exit(); System.exit(0); } });

 //potrebno za ispravno iscrtavanje Swing komponenti
  new Timer().schedule(new TimerTask() {
          public void run() {
                swingNode.getContent().repaint();
                swingNode1.getContent().repaint(); } }, 500L
    );
```

Interakcija dva Swing gumba unutar JavaFX aplikacije.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

```
1  private void inicijalizirajSwing(final SwingNode swingNode,
       final SwingNode swingNode1) {
2          SwingUtilities.invokeLater(new Runnable() {
3              @Override public void run() {
4                  JButton b = new JButton("Pritisni!");
5                  swingNode.setContent(b);
6                  JButton b1 = new JButton("Onesposobi gumb");
7                   b1.setSize(60,60);
8                  swingNode1.setContent(b1);
9                  ImageIcon enabledIcon = new ImageIcon( "
       putanja\\icons8-edit.gif" );
10                 ImageIcon disabledIcon = new ImageIcon( "
       putanja\\icons8-xbox-x-32.png" );
11                 b.setIcon(enabledIcon);
```

Interakcija dva Swing gumba unutar JavaFX aplikacije.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

```
1  //interakcija je u potpunosti u Swing kontekstu
2  b1.addActionListener(new ActionListener(){
3      @Override public void actionPerformed(ActionEvent e){
4  //osposobimo/onesposobimo gumb i postavimo odgovarajucu
       ikonu te promijenimo tekst na gumbu koji izvrsava akciju
5          if(b.isEnabled()){
6            b.setIcon(disabledIcon);
7            b.setEnabled(false);
8            b1.setText("Osposobi gumb!"); }
9          else{
10           b.setIcon(enabledIcon);
11           b.setEnabled(true);
12           b1.setText("Onesposobi gumb!"); } }  });
```

Interakcija dva Swing gumba unutar JavaFX aplikacije.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

Primjer 5

Dodajte u prethodni primjer još jedan *JavaFX* gumb. Zadani gumb postaje aktivan nakon što prvi gumb postavi tekst labeli (kada poništi tekst postaje neaktivan). Navedeni gumb može osposobiti/onesposobiti drugi gumb zadužen za osposobljavanje/onesposobljavanje prvog gumba. Ukoliko je prvi gumb onesposobljen, treći gumb ne može onesposobiti drugi gumb.

```
1 Button jfxbutton = new Button("Onesposobi1");
2 pane.setBottom(jfxbutton);
3
4 EventHandler<javafx.event.ActionEvent> event = new
      EventHandler<javafx.event.ActionEvent>() {
5     @Override public void handle(javafx.event.ActionEvent e){
6         SwingUtilities.invokeLater(new Runnable() {
7          @Override public void run() {
8              JButton b = (JButton)swingNode.getContent();
9              JButton b1  = (JButton)swingNode1.getContent();
```

Interakcija dva Swing i jednog JavaFX gumba.

Ugrađivanje *Java Swing*-a u *JavaFX* aplikaciju

```
1            if(b.isEnabled()){
2              if(b1.isEnabled()){
3                b1.setEnabled(false);
4                Platform.runLater(new Runnable(){
5                  @Override public void run(){
6                    if(jfxbutton.getText().equals("
7                       Onesposobi1"))
8                     jfxbutton.setText("Osposobi1");
9                    else jfxbutton.setText("Onesposobi1"); }
10                   } ); }
11                    else{ b1.setEnabled(true);
12                        Platform.runLater(new Runnable(){
13                    @Override public void run(){
14                        if(jfxbutton.getText().equals("
15                            Onesposobi1"))
16                    jfxbutton.setText("Osposobi1");
17                   else jfxbutton.setText("Onesposobi1"); }
18               } ); } } } } );} };
19         jfxbutton.setOnAction(event);
```

Interakcija dva Swing i jednog JavaFX gumba.

Korištenje pozadinskih dretvi u *Java Swing*-u

Regularnim *Java* dretvama kojima smo računali zadatke kod aplikacija bez korisničkog sučelja **ne smijemo mijenjati grafičke komponente** *Java Swing*-a. Ukoliko koristimo regularne *Java* dretve, moramo korisničko sučelje ažurirati **glavnom *Swing* dretvom**, kao što smo to radili i kod interakcije *Java Swing*-a i *JavaFX*-a.

```
1  private void start() {
2    Thread dretvaRadnik = new Thread() {
3     public void run() {
4      // Neki zadatak
5      for(int i=0; i<=10; i++) {
6       final int brojac = i;
7       //poziv glavne Swing dretve
8       SwingUtilities.invokeLater(new Runnable() {
9        public void run() {
10         labelaBrojaca.setText(Integer.toString(brojac));}});
11      try { Thread.sleep(1000);
12      } catch (InterruptedException e) {  }
13    }
```

Višedretvenost kod Java Swing-a.

Korištenje pozadinskih dretvi u *Java Swing*-u

```
1     SwingUtilities.invokeLater(new Runnable() {
2      public void run() { labelaStatusa.setText("Zadatak
      obavljen"); } }); } };
3      dretvaRadnik.start(); }
```

Višedretvenost kod Java Swing-a.

Ukoliko koristimo standardne *Java* dretve, **moramo se sami brinuti za sinkronizaciju**.

Postoji i bolji način izrade višedretvenih programa u *Java Swing*-u - **korištenjem *Swing* dretvi radnika**. Ta vrsta dretvi klase koja nasljeđuje `SwingWorker` je posebno dizajnirana za rad u *Swing*-u. *Swing* dretve se **ne mogu ponovo upotrebljavati**, stoga **za svaki zadatak moramo kreirati novu**.

Korištenje pozadinskih dretvi u *Java Swing*-u

Pozadinski zadatak možemo izvršiti koristeći *Swing* dretve tako da nadjačamo metodu `doInBackground()` klase `SwingWorker`.

```
1 SwingWorker<Void, Void> radnik = new SwingWorker<Void, Void
    >() {//funkcija vraca Void i ne prijavljuje nikakve
    vrijednosti tijekom izvrsavanja (drugi Void)
2   @Override  protected Void doInBackground() throws
    Exception {
3    // Neki zadatak.
4    for (int i = 0; i <= 10; i++) {
5     Thread.sleep(1000);
6     System.out.println("Izvrseno " + i); }
7    return null; } }; //definiramo zadatak
8  radnik.execute(); //izvrsimo zadatak
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu Swing dretvu.

Swing dretve **imaju ugrađeni mehanizam kojim se može mijenjati grafičko sučelje ovisno o rezultatu izračuna** `doInBackground()`.

Korištenje pozadinskih dretvi u *Java Swing*-u

To možemo napraviti **vraćanjem vrijednosti iz** `doInBackground` i **nadjačavanjem metode** `done()` (kojom **na siguran način** možemo napraviti promjene grafičkog sučelja). Korištenjem metode `get()` **dohvaćamo vrijednost** iz `doInBackground`. Prvi parametar definira povratni tip `doInBackground` i `get`.

```
1 SwingWorker<Boolean, Void> radnik = new SwingWorker<Boolean,
      Void>() {
2     @Override
3     protected Boolean doInBackground() throws Exception {
4      for (int i = 0; i <= 10; i++) {
5       Thread.sleep(1000);
6       System.out.println("Running " + i); }
7      return true; } //vracamo vrijednost
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu Swing dretvu.

Korištenje pozadinskih dretvi u *Java Swing*-u

```
1   // Na siguran nacin mozemo azurirati graficko sucelje
2   protected void done() {
3
4    boolean status;
5    try {
6     // Dohvacamo vracenu vrijednost.
7     status = get();
8     labelaStatusa.setText("Dovrseno, status: " + status);
9    } catch (InterruptedException e) {
10   } catch (ExecutionException e) {
11       /*iznimka vracena u doInBackground*/  } } };
12  radnik.execute();
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu Swing dretvu.

Drugi parametar *Swing* dretve koristimo za **ažuriranje grafičkog sučelja tijekom izvođenja *Swing* dretve**. Korištenjem metode `publish` **vraćamo vrijednosti s kojima želimo osvježiti korisničko sučelje** (može biti proizvoljnog tipa specificiranog drugim parametrom). Sljedeći korak je **nadjačati metodu** `process` koja dohvaća vrijednosti koje vraća `publish`.

Korištenje pozadinskih dretvi u *Java Swing*-u

`process()` prima listu vrijednosti zato što može biti objavljeno nekoliko vrijednosti prije poziva metode `process`.

```
1  SwingWorker<Boolean, Integer> radnik = new SwingWorker<
       Boolean, Integer>() {
2      @Override protected Boolean doInBackground() throws
       Exception {
3       for (int i = 0; i <= 10; i++) {
4        Thread.sleep(1000);
5        publish(i);}//objavljena vrijednost (tip jednak drugom
       generickom tipu radnika)
6       return true;  }
7
8      protected void done() {
9       boolean status;
10      try { status = get();
11     labelaStatusa.setText("Zavrseno statusom: " + status);
12      } catch (InterruptedException e) {
13      } catch (ExecutionException e) { } }
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu Swing dretvu.

Korištenje pozadinskih dretvi u *Java Swing*-u

```
1 //mozemo na siguran nacin mijenjati graficko sucelje
2     @Override protected void process(List<Integer> dijelovi)
      {
3      //vrijednosti
4      int zadnjaVrijednost = dijelovi.get(dijelovi.size()-1);
5     labelaBrojac.setText(Integer.toString(zadnjaVrijednost));
       } };
6   worker.execute();
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu Swing dretvu.

Korištenje pozadinskih dretvi u *JavaFX*-u

Pozadinske dretve *JavaFX*-a su sadržane u paketu `javafx.concurrent`. *JavaFX* scenski graph **nije siguran za korištenje bez sinkronizacije i treba mu se pristupati iz glavne dretve *JavaFX***. Pozadinske dretve *JavaFX*-a imaju sličnu funkcionalnost i dobrobiti kao i *Java Swing* pozadinske dretve.

Paket `javafx.concurrent` sadrži sučelje `Worker` i dvije klase `Task` i `Service` koje implementiraju `Worker`. Sučelje `Worker` sadrži API za komunikaciju pozadinske dretve s korisničkim sučeljem

Klasa `Task` je implementacija klase `java.util.conurrent.FutureTask`. **Omogućava implementaciju asinkronih zadataka u *JavaFX* aplikacijama**. Klasa `Service` **izvodi zadatke**. Klasa `WorkerStateEvent` specificira što se događa pri promjeni stanja pozadinske dretve (`failed`, `running`, `cancelled`, itd). Te događaje stanja mogu osluškivati instance klase `Task` i `Service` jer implementiraju sučelje `EventTarget`.

Korištenje pozadinskih dretvi u *JavaFX*-u

Sučelje `Worker` **definira objekt koji izvodi zadatak na jednoj ili više pozadinskih dretvi**. Stanje tog objekta je uočljivo aplikacijskoj *JavaFX* dretvi. Pri kreiranju je objekt `Worker` u stanju `READY`. Kada se radniku dodijeli zadatak, stanje mu postaje `SCHEDULED`, pri izvršavanju `RUNNING`, kod uspješnog završetka izvođenja `SUCCEEDED`, ukoliko je došlo do iznimaka pri izvršavanju `FAILED` i `CANCELLED` ukoliko je dretva dobila signal za prekid.

Trenutni status izvršavanja objekta `Worker` se može provjeriti svojstvima `totalWork`, `workDone` i `progress`.

Zadaci se koriste za **implementaciju logike zadatka** koji se treba izračunati od strane pozadinske dretve. Zadatak se definira nadjačavanjem `call` metode klase `Task` (smije manipulirati samo onim stanjima kod kojih se čitanje i pisanje može raditi na siguran način, inače dolazi do iznimke pri izvršavanju). Klasa osigurava da se promjene svojstava, dojave greške, otkazivanja, obrađivači događaja i stanja **događaju u *JavaFX* aplikacijskoj dretvi**.

Korištenje pozadinskih dretvi u *JavaFX*-u

Unutar metode `call` se mogu koristiti metode `updateProgress`, `updateMessage` i `updateTitle` koje **ažuriraju odgovarajuće svojstvo na *JavaFX* aplikacijskoj dretvi**. **Ukoliko je zadatak otkazan, povratna vrijednost se ignorira**.

Pošto `Task` nasljeđuje klasu `java.utils.concurrent.FutureTask` koja implementira sučelje `Runnable`, `Task` se može koristiti s **izvršiteljima** (`ExecutorService.submit(task);`), **kao parametar dretvi** (`Thread th = new Thread(task); th.setDaemon(true); th.start();`) ili se može **pozivati direktno koristeći naredbu** `FutureTask.run()` (omogućuje poziv iz pozadinske dretve). Klasa `Task` **definira zadatak koji se izvršava samo jednom** (kao i kod *Swing* dretvi). Klasa `Service` **nudi funkcionalnost radnika koji se može koristiti više puta**.

Ne postoji pouzdan način za zaustavljanje dretve koja izvodi zadatak, međutim dretva bi trebala često provjeravati signal o otkazivanju (metoda `isCancelled`).

Korištenje pozadinskih dretvi u *JavaFX*-u

```
1  import javafx.concurrent.Task;
2
3  Task<Integer> zadatak = new Task<Integer>() {
4    @Override protected Integer call() throws Exception {
5       int iteracije;
6       for (iteracije = 0; iteracije < 500; iteracije++) {
7               if (isCancelled()) { updateMessage("Otkazano!");
8                   break; }
9               updateMessage("Iteracija " + iteracije);
10              updateProgress(iteracije, 500);
11
12   //provjera iznimke InterruptedException i svojstva
       isCancelled
13              try {  Thread.sleep(100);
14              } catch (InterruptedException interrupted) {
15                  if (isCancelled()) { updateMessage("Otkazano
       !");
16                      break; } } }
17          return iterations; } };
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu JavaFX dretvu.

Korištenje pozadinskih dretvi u *JavaFX*-u

Sljedeći dio koda pokazuje kako ažurirati `progress bar` koji pokazuje stanje izvršavanja pozadinske dretve.

```
1  import javafx.concurrent.Task;
2
3  Task zadatak = new Task<Void>() {
4      @Override public Void call() {
5          static final int maks = 1000000;
6          for (int i=1; i<=maks; i++) {
7              if (isCancelled()) {  break; }
8              updateProgress(i, max); } //azurira progress bar
       ali i svojstva totalWork i workDone
9          return null; } };
10 ProgressBar bar = new ProgressBar();
11 bar.progressProperty().bind(zadatak.progressProperty()); //
       povezujemo progress bar i zadatak
12 new Thread(zadatak).start();
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu JavaFX dretvu.

Korištenje pozadinskih dretvi u *JavaFX*-u

Kod svake promjene stanja radnika (`Worker`) **dolazi do odgovarajućeg događaja**. Npr. kada se zadatak uspješno izvrši (instanca klase `Task` prijeđe u stanje `SUCCEEDED`) a `Worker` u stanje `WORKER_STATE_SUCCEEDED`. Tada se **poziva obrađivač događaja** `onSucceeed` nakon čega se **poziva zaštičena metoda** `succeded` na aplikacijskoj `JavaFX` dretvi. Postoji **više različitih vrsta zaštičenih metoda** kao `cancelled`, `failed`, `running`, `scheduled`, `succeeded`. Te metode se mogu nadjačati podklasama klasa `Task` i `Service`.

```
1 import javafx.concurrent.Task;
2
3 Task<Integer> task = new Task<Integer>() {
4     @Override protected Integer call() throws Exception {
5         int iteracije = 0;
6         for (iteracije = 0; iteracije < 500; iteracije++) {
7             if (isCancelled()) { break; }
8             System.out.println("Iteracija " + iteracije); }
9         return iteracije; }
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu JavaFX dretvu.

Korištenje pozadinskih dretvi u *JavaFX*-u

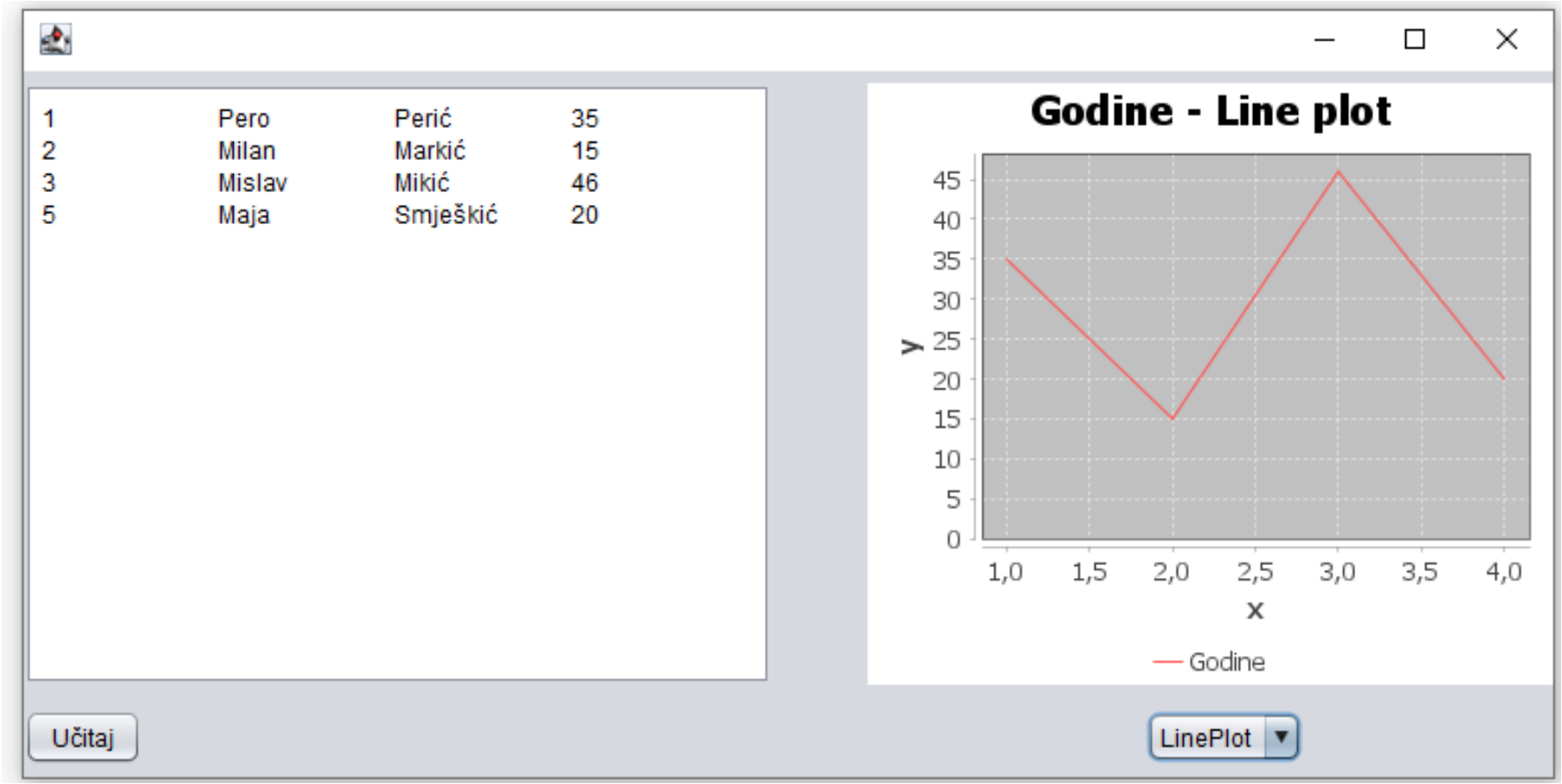
```
1      @Override protected void succeeded() { //pozove se kod
       uspjesnog izvrsavanja
2           super.succeeded();
3           updateMessage("Uspjeh!"); }
4
5      @Override protected void cancelled() { //poziva se kada
       je zadatak otkazan
6           super.cancelled();
7           updateMessage("Otkazano!"); }
8
9  @Override protected void failed() { //poziva se kada se
       zadatak nije uspio izvrsiti (doslo je do greske ili
       iznimke)
10     super.failed();
11     updateMessage("Neuspjelo!"); } };
```

Izvršavanje pozadinskog zadatka koristeći pomoćnu JavaFX dretvu.

Izrada cjelovite aplikacije sa sučeljem

Primjer 6

Napravite aplikaciju s grafičkim sučeljem koja čita podatke iz baze podataka (u pozadinskoj dretvi) i vizualizira ih koristeći `jFreeChart` biblioteku.



Izrada cjelovite aplikacije sa sučeljem

Na formu grafičkog sučelja ćemo staviti spremnik slobodnog dizajna (sami ćemo poslagati komponente). Na desnom dijelu je spremnik sa zadanim razmještajem `BorderLayout` (spremnik sa slobodnim razmještajem ne prikazuje grafove). Na pritisak gumba *Ucitaj* se otvara prozor za odabir datoteke baze (tip datoteke `.db`). Nakon učitavanja se podaci ispišu unutar komponente `textArea` i iscrtaju na grafu tipa izabranog na `combo box` komponenti.

```
1 public Kombinacija() {
2         initComponents();//automatski generirana funkcija
3         FileNameExtensionFilter filter = new
    FileNameExtensionFilter("SQLITE BAZE", "db", "database")
    ; //postavljamo filter na komponentu odabira datoteke
4         jchooser1.setFileFilter(filter);
5         osobe = new ArrayList<>(); //definiramo pomocnu
    listu elemenata klase osoba
6     }
```

Povezivanje raznih funkcionalnosti.

Izrada cjelovite aplikacije sa sučeljem

```
1  private ArrayList<Osoba> osobe;
2  private String imeDatoteke;
3  private  JFileChooser jchooser1 = new JFileChooser(); //
       dodatne potrebne komponente
4    private void jButton1ActionPerformed(java.awt.event.
       ActionEvent evt) {
5            int izbor = jchooser1.showOpenDialog((Component)
       evt.getSource());
6     if (returnVal == JFileChooser.APPROVE_OPTION) {
7         File dokument = jchooser1.getSelectedFile();
8         try {
9            imeDatoteke = file.toString();
10        } catch (Exception e) {
11           //obrada iznimke otvaranja datoteke
12        }
13
14        jButton1.setEnabled(false); //posto pristupamo u
       pozadinskoj dretvi, onemogucimo pristup bazi dok se
       zadatak ne izvrsi
```

Povezivanje raznih funkcionalnosti.

Izrada cjelovite aplikacije sa sučeljem

```
1   String url = "jdbc:sqlite:" + file.getAbsolutePath();
2        SwingWorker<ResultSet, Void> worker = new
    SwingWorker<ResultSet, Void>() {
3   @Override
4   protected ResultSet doInBackground() throws Exception {
5     String sql = "SELECT id, ime, prezime, godine FROM
    osobe";
6       Connection conn = null;
7       ResultSet result = null;
8        try {
9            conn = DriverManager.getConnection(url);
10           Statement stmt  = conn.createStatement();
11            result    = stmt.executeQuery(sql);
12       } catch (SQLException e) {
13           System.out.println(e.getMessage()); }
14   return result; } //citanje podataka iz baze
```

Povezivanje raznih funkcionalnosti.

Izrada cjelovite aplikacije sa sučeljem

```
1    @Override protected void done() { //pri zavrsetku citanja
      vrati podatke
2     ResultSet result;
3     try {
4      result = get();
5      String txt = "";
6      String ime, prezime;
7      int godine, id;
8       while (result.next()) {
9
10            id = result.getInt("id");
11            ime = result.getString("ime");
12            prezime = result.getString("prezime");
13            godine =  result.getInt("godine");
14
15            Osoba o = new Osoba(ime,prezime,godine,id);
16            osobe.add(o); }
```

Povezivanje raznih funkcionalnosti.

Izrada cjelovite aplikacije sa sučeljem

```
1     for(int i=0;i<osobe.size();i++){
2         txt+=osobe.get(i).id +  "\t" + osobe.get(i).ime + "\t
    " + osobe.get(i).prezime+"\t"+osobe.get(i).godine;
3 txt+="\n"; }
4
5      jTextArea1.setText(txt);  //postavimo tekst
6      jButton1.setEnabled(true); //osposobimo gumb za ponovo
    citanje
7      jComboBox1.setEnabled(true);} //osposobimo combo box
    posto sada imamo podatke
8     catch(SQLException e){ } //iznimka pristupa bazi
9     catch (InterruptedException e) {
10     // obrada prekida dretve
11    } catch (ExecutionException e) {
12     //obrada iznimaka iz doInBackground
13    } } };
14      worker.execute(); }
15    else return; //korisnik je otkazao ucitavanje datoteke
16    }
```

Povezivanje raznih funkcionalnosti.

Izrada cjelovite aplikacije sa sučeljem

```
1  void jComboBox1ActionPerformed(java.awt.event.ActionEvent
     evt) {
2          if(((String)jComboBox1.getSelectedItem()).equals("
     LinePlot")){
3                  DefaultXYDataset ds = new DefaultXYDataset();
4                  double data[][] = new double[2][osobe.size()];
5
6                      for(int i=0;i<osobe.size();i++){
7                          data[0][i] = (i+1);
8                          data[1][i] = osobe.get(i).godine; }
9
10                     ds.addSeries("Godine", data);
11       JFreeChart chart = ChartFactory.createXYLineChart("
     Godine - Line plot", "x", "y", ds, PlotOrientation.
     VERTICAL, true, true, false);
12
13                    ChartPanel cp = new ChartPanel(chart);
14                    cp.setVisible(true);
15                    cp.setPreferredSize(new Dimension(300,300));
```

Povezivanje raznih funkcionalnosti.

Izrada cjelovite aplikacije sa sučeljem

```
1          jPanel2.add(cp,BorderLayout.CENTER);
2          this.pack();
3          this.setVisible(true); } }
4
5  public class Osoba {
6      String ime, prezime;
7      int id, godine;
8
9      public Osoba(String i, String p, int g, int idd){
10         ime = i; prezime = p; godine = g; id = idd; }
11
12     public String getIme(){ return ime; }
13     public String getPrezime(){ return prezime; }
14     public double getGodine(){ return godine; }
15     public int getID(){ return id; } }
```

Povezivanje raznih funkcionalnosti.