NAPOMENA: Vežbe podrazumevaju da je student ovladao teorijom iz dela "Swing" (1-Swing.ppt).

Java Swing

Java Swing biblioteka omogućava pisanje *Java* aplikacija koje poseduju grafički korisnički interfejs (engl. *GUI*). Sastavni je deo *Java* platoforme.

Skup komponenti dostupnih u *Swingu* proteže se od jednostavnih – bazičnih (dugmad, labele, tekstualno polje) do veoma kompleksih (tabele, drvo prikaza, itd.). Ukoliko ni jedna od ponuđenih komponenti ne zadovoljava potrebe korisnika, *Swing* pruža mogućnost izgradnje sopstvenih komponenti oslanjajući se na bazične (*Swing* omogućava kreiranje sopstvenih komponeti nasleđivanjem postojećih klasa).

Korisnični interfejs je *Event-Driven* – te se *Swing* aplikacijom upravlja preko događaja izvršenim u okviru komponenti (klik miša, pritisak tastera tastature, itd.). Potrebo je da se napišu samo procedure koje bi izvršili po nastanku određenog događaja korisničkog interfejsa. U okviru biblioteke postoji 18 paketa i svi počinju sa **javax.swing**. Nazivi klasa počinju sa velikim slovo J.

Svaka *Swing* aplikacija započinje prikazom barem jednog *top-level* kontejnera (kontejnera najvišeg nivoa). *Top-level* kontejneri su osmišljeni sa ciljem da se u njima skladište i prikažu komponente korisničkog interfejsa (dugmad, labele, polja unosa teksta, itd.). Komponente korisničnog interfejsa moraju biti u okviru nekog kontejera.

Najčešće korišćeni kontejneri najvišeg nivoa su:

1. **JFrame** - Koristi se za implementaciju glavnog prozora aplikacije. Glavni prozor poseduje naslovnu liniju, tri pozadinska dugmeta i prostor za izcrtavanje komponenti. *JFrame* može imati menije, toolbarove itd.
2. **JDialog** - Koristi se za kreiranje dijaloga aplikacije. Dijalozi uglavnom služe za prikupljanje informacija od korisnika ili prikazivanje poruka. Za razliku od prozora (*JFrame*), dijalozi uglavnom imaju samo *Ok* i *Cancel* programsku dugmad (iako nije obavezno), isto tako, dijalozi mogu da budu modalni, dok prozori nemaju tu mogućnost.

Primer kreiranja glavnog prozora aplikacije upotrebom klase *JFrame*:

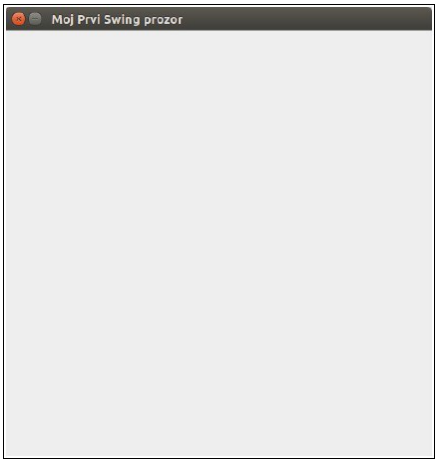
```
public class MojPrviProzor extends JFrame {

	// Sva inicijalizacija se vrsi u konstruktoru
	public MojPrviProzor() {
		// Naslov prozora
		setTitle("Moj Prvi Swing prozor");
		// Sirina i visina prozora u pikselima
		setSize(500, 500);
		/*
			Centrira prozor u odnosu na prosledjenu komponentu.
			Ako se prosledi null, prozor se prikazuje na sredini ekrana.
			Ako se ne pozove ova funkcija, prozor se prikazuje u
			gornjem levom uglu ekrana.
		 */
		setLocationRelativeTo(null);
		/*
		 Specificira kako ce se prozor ponasati kada se zatvori (klik na
		 dugme X).
		 Opcije:
			- DISPOSE_ON_CLOSE: Zatvara prozor i oslobadja zauzete
				memorijske resurse. Ako je ovo bio jedini otvoreni
				prozor u programu, program se prekida.
			- EXIT_ON_CLOSE:  Zatvara prozor i prekida program.
			- HIDE_ON_CLOSE:  Samo vizuelno sakriva prozor.
			- DO_NOTHING_ON_CLOSE:  Dugme za zatvaranje prozora ne radi
				nista, ocekuje se od korisnika da napise reakciju na
				klik na ovo dugme.
		 */
		setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		setResizable(false);
	}
}
```

Pokretanje aplikacije se vrši instanciranjem klase *MojPrviProzor* i pozivanjem metode *setVisible()*.

```
public static void main(String[] args) {
	MojPrviProzor prozor = new MojPrviProzor();
	prozor.setVisible(true);
}
```

Izgled prozora sa primera:



Dodavanje komponenti u kontejner

Vrši se metodom add(Component c):

```
JButton btnOk = new JButton("OK");
add(btnOk, BorderLayout.SOUTH);
```

Rezultat:

Rukovanje stanjem prozora

Atribut *ExtendedState* (int) je nasleđen iz klase *Frame* koji definiše vizuelno stanje prozora (minimizovan, maksimizovan, …). Modifikuje se odgovarajućim get i set metodama. Moguće vrednosti:

- `Frame.NORMAL`
- `Frame.ICONIFIED`
- `Frame.MAXIMIZED_HORIZ`
- `Frame.MAXIMIZED_VERT`
- `Frame.MAXIMIZED_BOTH`

Ukoliko se postavi setResizable(false), promena stanja prozora se neće videti.

Pomoćna klasa Toolkit

Predstavlja interfejs za rukovanje trenutnim grafičkim okruženjem pozivanjem statičke metode *getDefaultToolkit()* klase *Toolkit*. Primer upotrebe je dat u nastavku:

```
Toolkit toolkit = Toolkit.getDefaultToolkit();
Dimension screenSize = toolkit.getScreenSize();
int screenHeight = screenSize.height;
int screenWidth = screenSize.width;
```

Dijalozi

Dijalozi se kreiraju proširenjem klase *JDialog* i postavljanjem komponenti na isti način kao i za *JFrame*. Međutim, za standardne dijaloge o ispisu poruke i potvrdi se mogu koristiti generički dijalozi. Klasa *JoptionPane* sadrži statičke metode za kreiranje tipičnih dijaloga.

Dijalog za prikaz poruke

Metoda *showMessageDialog()* klase *JoptionPane* služi za prikazivanje dijaloga sa određenom porukom. Moguće je navesti naslov prozora, poruku i tip poruke. Izgled dijaloga se menja na osnovu prosleđenog tipa poruke.

```
JoptionPane.showMessageDialog( null, "Primer INFORMATION_MESSAGE poruke",
                               "Info", JoptionPane.INFORMATION_MESSAGE);
```


```
JoptionPane.showMessageDialog( null, "Primer ERROR_MESSAGE poruke", "Greška",
                               OptionPane.ERROR_MESSAGE);
```


```
JoptionPane.showMessageDialog( null, "Primer WARNING_MESSAGE poruke", "Warning",
                               JoptionPane.WARNING_MESSAGE);
```

Dijalog za potvrdu određene akcije

Metoda *showConfirmDialog()* klase *JoptionPane* služi za prikazivanje dijaloga gde se očekuje potvrda korisnika (klik) za neku od ponuđenih opcija. Pored tipa poruke, zadaje se i broj ponuđenih opcija.

```
JOptionPane.showConfirmDialog(null, "Da li ste sigurni?", "Pitanje",
                              JOptionPane.YES_NO_CANCEL_OPTION);
```

Metoda vraća celobrojnu vrednost (int) koja označava odbaranu opciju.

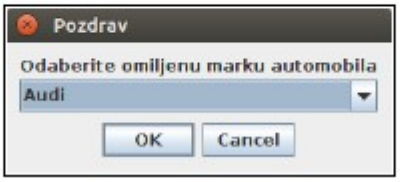
```
int choice = JOptionPane.showConfirmDialog(null, "Da li ste sigurni?",
                  "Pitanje", JOptionPane.YES_NO_CANCEL_OPTION);

if(choice == JOptionPane.YES_OPTION) {
      System.out.println("Kliknuto dugme 'YES'.");
}else if(choice == JOptionPane.NO_OPTION) {
      System.out.println("Kliknuto dugme 'NO'.");
}else if(choice == JOptionPane.CANCEL_OPTION) {
      System.out.println("Kliknuto dugme 'CANCEL'.");
}
```

Dijalog za prikupljanje podataka

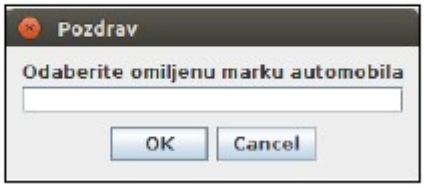
Metoda *showInputDialog()* klase *JoptionPane* služi za prikupljanje podataka od korisnika putem tekstualnog polja ili „padajuće“ liste.

```
JoptionPane.showInputDialog(null, "Odaberite omiljenu marku automobila",
                            "Pozdrav", JOptionPane.PLAIN_MESSAGE, null,
                             new Object[] {"Audi", "Volvo", "Mazda"}, "Audi");
```



Ukoliko se niz sa opcijama zameni sa *null*, prikazuje se verzija sa tekstualnim poljem:

```
JOptionPane.showInputDialog(null, "Odaberite omiljenu marku automobila",
                            "Pozdrav", JOptionPane.PLAIN_MESSAGE, null,
                             null, "");
```



Preuzimanje unesene ili odabrane vrednosti:

```
String choice = (String)JOptionPane.showInputDialog(null, "Odaberite omiljenu
marku automobila", "Pozdrav", JOptionPane.PLAIN_MESSAGE, null, null, "");

System.out.println("Odabrano je: " + choice);
```

Upravljanje događajima

U programima sa grafičkim korisničkim interfejsom, izvršavanje programskog koda se bazira na događajima koje generiše korisnik (klik mišem, pritisak tastera na tastaturi, itd.). Na nama je da isprogramiramo reakcije na željene događaje. Ovaj način predstavljanja aplikacije se zove *Event-Driven*.

Svaka akcija nad komponentama korisničkog interfejsa izaziva generisanje događaja (instanci naslednika *EventObject* klase). Navedene komponente korisničkog interfejsa se u ovom kontekstu nazivaju izvori događaja (engl. *event sources*). Događaji se prosleđuju svim "osluškivačima" (engl. *Listener*s) događaja koji su se kod izvora događaja registrovali da ih dati događaj zanima. Slično *event-driven* ponašanje se postiže primenom šablona *Observer-Observable* (rađenom na prethodnim vežbama).

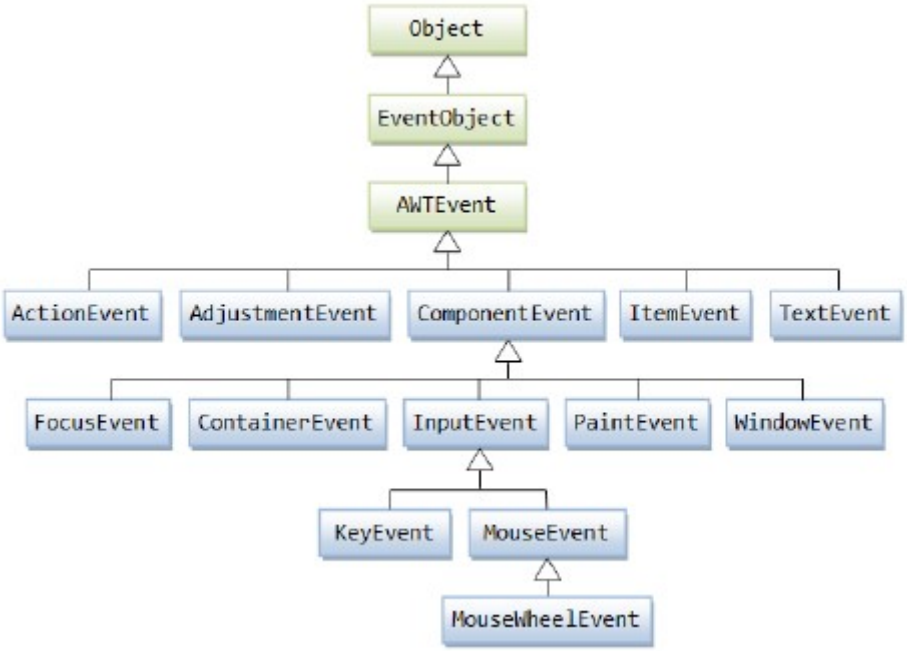
Primer osluškivača (*listener*-a) događaja klika na dugme:

```
JButton btnOk = new JButton("OK");

btnOk.addActionListener(new ActionListener() {
      @Override
      public void actionPerformed(ActionEvent e) {
            System.out.println("KLIK!");
      }
});
```

Na dugme *btnOk* se dodaje osluškivač događaja metodom *addActionListener()* u kojoj se definiše klasa *ActionListener* za obradu *ActionEvent*-a. Dakle, pri kliku se kreira objekat navedene klase *ActionListener* unutar kog se nalazi implementirana metoda za obradu događaja. Događaj *e* metode *actionPerformed()* sadrži podatke komponente koja je izazvala događaj (u ovom slučaju dugme *OK*).

Slično ovome, za sve vrste korisničkih akcija postoji odgovarajuća Event klasa.



Za svaku Event klasu postoji barem jedan *Listener* interfejs:

Događaj	Dodeljeni interfejs
ActionEvent	ActionListener
AdjustmentEvent	AdjustmentListener
ComponentEvent	ComponentListener
ContainerEvent	ContainerListener
FocusEvent	FocusListener
KeyEvent	KeyListener
MouseEvent	MouseListener, MouseMoveListener
WindowEvent	WindowListener, WindowStateListener
ItemEvent	ItemListener
TextEvent	TextListener

Adapter klase

Adapter klase su uvedene za sve *xxxListener* interfejse koje imaju više od jedne metode, sa ciljem da olakšaju pisanje reakcija na događaje. One implementiraju *Listener* interfejs i obezbeđuju podrazumevane reakcije na događaje (najčešće prazna tela metoda). Koriste se isto kao i Listener-i.

```
JTextField txtProba = new JTextField();


txtProba.addKeyListener(new KeyAdapter() {
	@Override
	public void keyReleased(KeyEvent e) {
		if(e.getKeyCode() == KeyEvent.VK_A) {
			System.out.println("Pritisnuto slovo 'a'.");
		}
	}
});
```

Zadaci

- Kreirati *Java Swing* prozor sa dugmićima OK i CANCEL postavljenih na dnu prozora. Veličina prozora treba da predstavlja četvrtinu veličine ekrana i prozor treba da je centriran i vertikalno i horizontalno.
- Prilikom zatvaranja prozora (kreiranog u prethodnom koraku) pitati korisnika da potvrdi da je siguran da želi da zatvori prozor.
- Dodati JLabel komponentu u koju će se upisivati trenutne koordinate miša kada se miš pomera unutar prozora (pogledati dokumentaciju za klasu *MouseMotionAdapter*).
- Kreirati novi prozor sa naslovom “Drugi prozor”. Omogućiti da se taj prozor otvara klikom na dugme OK prethodnog prozora.