Java Swing

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

Set komponenti dostupnih u Swingu proteže se od jednostavnih – bazičnih (dugmad, labele, tekstualno polje) do veoma kompleksih (tabele, drvo prikaza). 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...). 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 slovom J.

Svaka Swing aplikacija započinje prikazom barem jednog top-lavel kontejnera (kontejnera najvišeg nivoa). Top-lavel kontejneri su osmišljeni sa ciljem da se u njima skladište i prikažu komponente korisničkog interfejsa (dugmad, labele, polja unosa teksta...). 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:

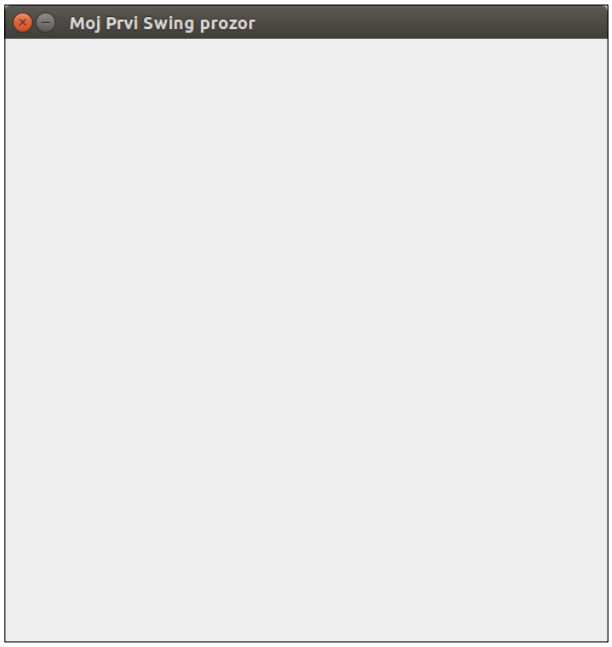
```
public class MojPrviProzor extends JFrame {

	// 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);
}
```

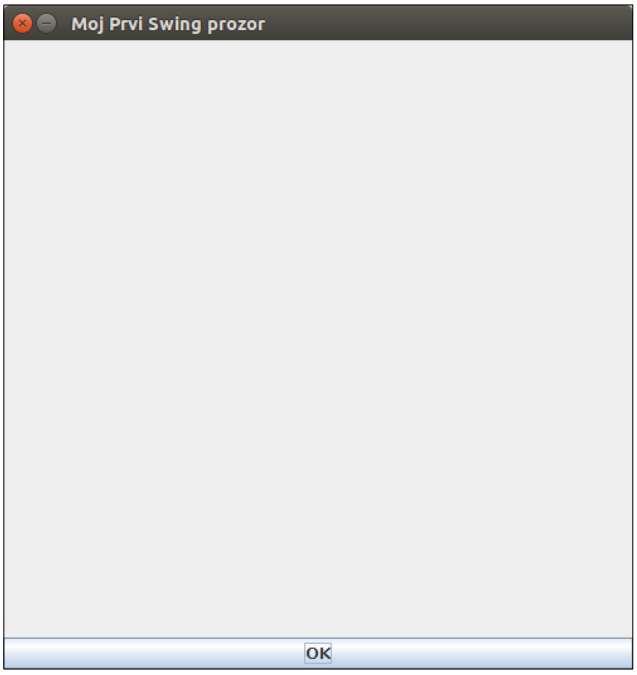
Izgled prozora sa primera:



Dodavanje komponenti u kontejner

Vrši se metodom **add**(Component c):

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



Pomoćna klasa `Toolkit`

Predstavlja interfejs za rukovanje trenutnim grafičkim okruženjem.

PRIMER: Preuzimanje dimenzija ekrana:

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

ZADATAK

Kreirati Java Swing prozor sa OK dugmetom kao sa prikazanog primera. Veličina prozora treba da predstavlja četvrtinu veličine ekrana i prozor treba da je centriran i vertikalno i horizontalno.

Dijalozi

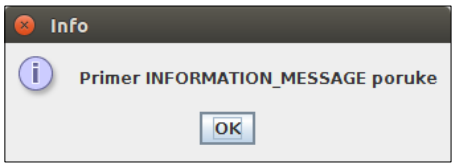
Za prikaz generičkih dijaloga možemo koristiti neku od statičkih metoda `JoptionPane` klase.

`showMessageDialog`

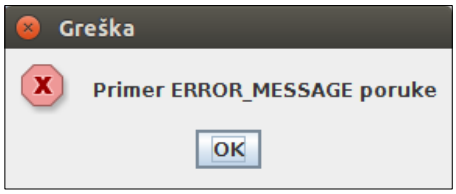
Služi za prikazivanje dijaloga sa porukom. Moguće je navesti naslov prozora, poruku i tip poruke. Izgled dijaloga se menja na osnovu prosleđenog tipa poruke.

PRIMERI:

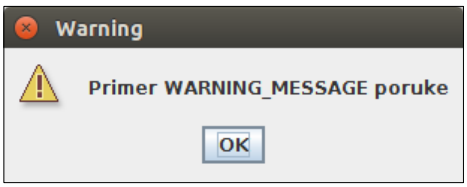
```
JoptionPane.showMessageDialog( null, "Primer INFORMATION_MESSAGE poruke",
                               "Info", JoptionPane.INFORMATION_MESSAGE);
```




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



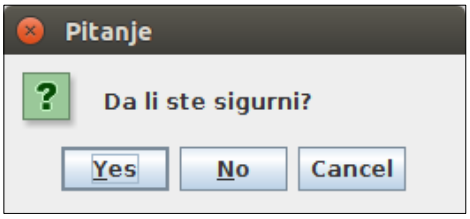

```
JoptionPane.showMessageDialog( null, "Primer WARNING_MESSAGE poruke", "Warning",
                               JoptionPane.WARNING_MESSAGE);
```



showConfirmDialog

Služi za prikazivanje dijaloga u kojem se prikazuje poruka nakon čega se očekuje da korisnik klikne na 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 int koji nam govori koja opcija je odabrana:

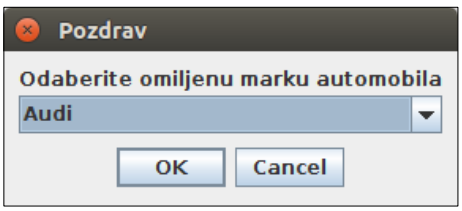
```
int choice = JOptionPane.showConfirmDialog(null, "Da li ste sigurni?",
                   "Pitanje", JOptionPane.YES_NO_CANCEL_OPTION);

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'.");
}
```

showInputDialog

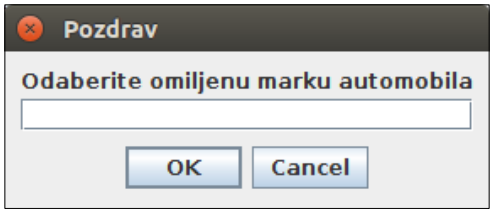
Služi za prikazivanje dijaloga u kojem se očekuje da korisnik unese nešto u tekstualno polje ili izabere neku od ponuđenih vrednosti iz padajuće liste.

```
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, …). 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).

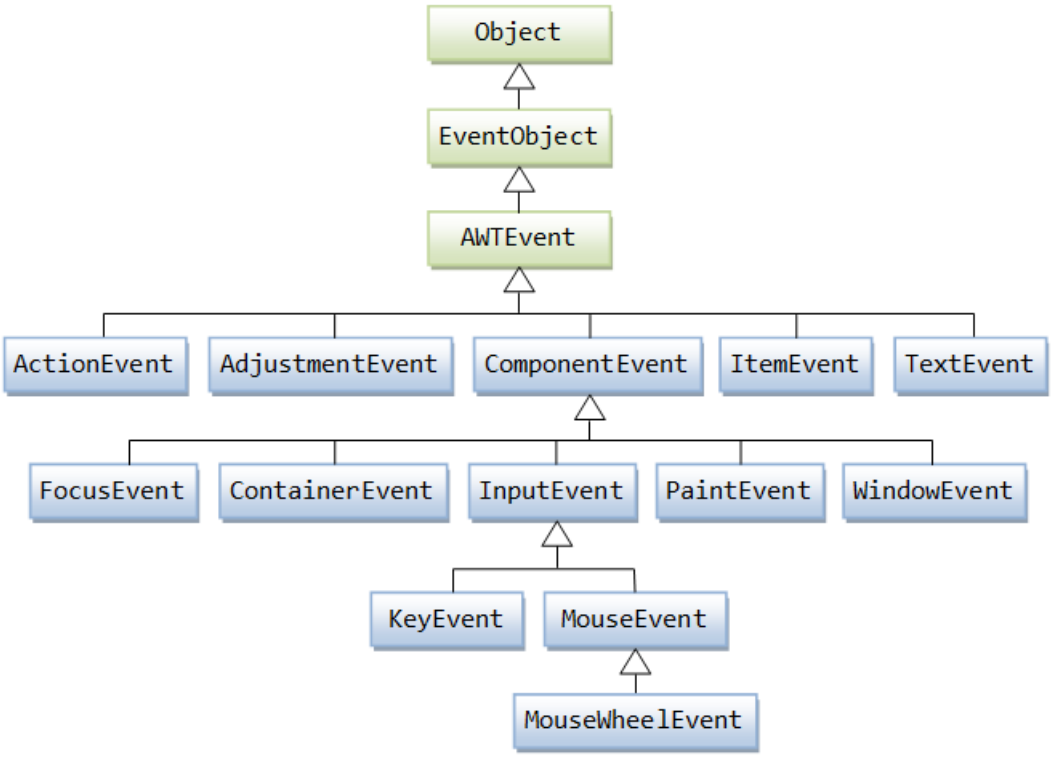
Navedene komponente korisničkog interfejsa se u ovom kontekstu nazivaju izvori događaja (event sources). Događaji se prosleđuju svim "osluškivačima" (*Listener*) događaja koji su se kod izvora događaja registrovali da ih dati događaj zanima.

Realizacija listenera

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


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

Možemo videti da se za klik kreira događaj tipa `ActionEvent`. Slično ovome, za sve vrste korisničkih akcija postoji odgovarajuća `Event` klasa.



Dalje, 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 interfejsa i obezbeđuju podrazumevane reakcije na događaje (najčešće prazna tela metoda). Koriste se isto kao i Listener-i.

Primer:

```
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

Izmeniti prozor kreiran u prvom zadatku tako da omogućava sledeće funkcionalnosti:

1. Prilikom zatvaranja prozora pitati korisnika da potvrdi da je siguran da želi da zatvori prozor.
2. Na poziciju BorderLayout.***NORTH*** dodati JLabel komponentu u koju će se upisivati trenutne koordinate miša kada se miš pomera unutar prozora (pogledati dokumentaciju za klasu MouseMotionAdapter).
3. Kreirati novi prozor sa naslovom “Drugi prozor”. Omogućiti da se taj prozor otvara klikom na dugme OK starog prozora.