11

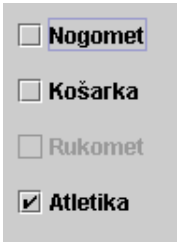
Složeniji elementi grafičkog korisničkog sučelja

- Kvadratići za odabir opcija
- Kružići za odabir opcija
- Područja za tekst
- Trake za pomicanje
- Padajuće liste
- Izbornici

U 6. poglavlju smo se upoznali s osnovnim elementima korisničkog sučelja (gumbi, Labele i okviri za tekst), osim navedenih, u programima često susrećemo i niz drugih elemenata korisničkog sućelja, a o najčešćima ćemo govoriti u ovom poglavlju. S obzirom da smo se već susretali s elementima grafičkog korisničkog sućelja, ovdje se nećemo puno zadržavati na pojedinom elementu.

Kvadratići za odabir opcija

Kvadratiće za odabir opcija koristimo kada u programu trebamo odabratu jednu, više ili niti jednu od ponuđenih opcija. Nakon što pojedinu opciju odaberemo pojavi se kvačica u kvadratiću ispred nje.



Slika 11 – 1: Primjer kvadratića za odabir opcija

Kvadratić za odabir kreirat ćemo koristeći klasu **`JCheckBox`**.

Najčešće metode koje ćemo koristiti pri radu s kvadratićima za odabir opcija su:

`public void setSize (int s, int v)` – definira veličinu kvadratića i područja za tekst koje se ispisuje pokraj kvadratića. Kvadratić je uvijek iste veličine, bez obzira na definiranu veličinu.
`public void setLocation (int x, int y)` – definira poziciju kvadratića za odabir na *Containeru.*
`public void setText (String s)` – definira tekst koji će se pojaviti pokraju kvadratića za tekst
`public String getText ()` – vraća tekst koji je upisan pokraj kvadratića za odabir
`public void setEnabled (boolean b)` – ukoliko je *b* postavljen na *true* kvadratić će biti aktivan i moći će se označiti (odznačiti), inače će biti neaktivan i nećemo nad njim napraviti nikakvu radnju.
`public void setSelected (boolean b)` – ukoliko je *b true* kvadratić će biti označen, inače neće biti označen. Inicijalno kvadratić neće biti označen.
`public void addActionListener (Object o)` – dodaje listener na kvadratiću za odabir.

Primjer 11 – 1:
Dio programa koji će na ekranu iscrtati gumbe kao na slici **11 – 1**.

Rješenje:

```
...
      private JCheckBox nog, kos, ruk, atl;
...
      nog = new JCheckBox ();
      nog.setText ("Nogomet");
      nog.setSize (80, 25);
      nog.setLocation (10, 10);
      c.add (nog);
```

```
      kos = new JCheckBox ();
      kos.setText ("Košarka");
      kos.setSize (80, 25);
      kos.setLocation (10, 40);
      c.add (kos);

      ruk = new JCheckBox ();
      ruk.setText ("Rukomet");
      ruk.setSize (80, 25);
      ruk.setLocation (10, 70);
      ruk.setEnabled (false);
      c.add (ruk);

      atl = new JCheckBox ();
      atl.setText ("Atletika");
      atl.setSize (80, 25);
      atl.setLocation (10, 100);
      atl.setSelected (true);
      c.add (atl);
...
```

Kružići za odabir opcija

Osim kvadratića o kojima smo upravo govorili za odabir možemo koristiti i kružiće. Kružiće ćemo u pravilu koristiti u slučajevima kada ćemo trebati od ponuđenih opcija odabrati točno jednu.

Slika 11 – 2: Primjer kružića za odabir opcija

Kružiće za odabir kreirat ćemo koristeći klasu **`JRadioButton`**.

Najčešće metode koje ćemo koristiti pri radu s kružićima za odabir su:

`public void setSize (int s, int v)` – definira veličinu kružića i područja za tekst koje se ispisuje pokraj njega. Kružić je uvijek iste veličine, bez obzira na definiranu veličinu.
`public void setLocation (int x, int y)` – definira poziciju kružića na *Containeru*.
`public void setText (String s)` – definira tekst koji će se ispisati pokraj kružića za odabir.
`public String getText ()` – vraća tekst koji je upisan pokraj kružića za odabir
`public void setEnabled (boolean b)` – ukoliko je *b* postavljen na *true* kružić će biti aktivan i moći će se označiti (odznačiti), inače će biti neaktivan i nećemo nad njim napraviti nikakvu radnju.
`public void setSelected (boolean b)` – ukoliko je *b true* kružić će biti označen, inače neće biti označen. Inicijalno kružić neće biti označen.
`public void addActionListener (Object o)` – dodaje listener na kružić za odabir.

Primjer 11 – 2:
Dio programa koji će na ekranu iscrtati gumbe kao na slici **11 – 2**.

Rješenje:
```
...
```

```
      private JRadioButton m, z;
...
      m = new JRadioButton ();
      m.setText ("Muški");
      m.setSize (80, 25);
      m.setLocation (10, 10);
      m.setSelected (true);
      c.add (m);

      z = new JRadioButton ();
      z.setText ("Ženski");
      z.setSize (80, 25);
      z.setLocation (10, 40);
      c.add (z);
...
```

Pokusamo li u gornjem primjeru označiti samo *Ženski* spol, neće nam bać poći za rukom, što god napraviti bit će označen i *Muški* spol. Najčešće to neće biti ono što želimo. Najčešće ćemo htjeti označiti samo jedan od gumbova. Kako to ne bismo morali programirati možemo koristiti jedan poseban element kojim ćemo grupirati gumbe i na taj način omogućiti označavanje samo jednog od gumba. Radi se o elementu koji neće biti vidljiv na samom prozoru, a kreirat ćemo ga koristeći klasu **`ButtonGroup`**.
Jedina metoda klase **`ButtonGroup`** koju ćemo koristiti je:

`public void add (Object o)` – dodaje gumb na grupu gumba, pri ćemu je *o* kružić ili pravokutnik za odabir.

Primjer 11 – 3:
Izmijenimo prethodni primjer tako da gumbe grupiramo te na taj način omogućimo da u svakom trenutku može biti označen samo jedan gumb.

Rješenje:

```
...
      private JRadioButton m, z;
      private ButtonGroup b;
...
      m = new JRadioButton ();
      m.setText ("Muški");
      m.setSize (80, 25);
      m.setLocation (10, 10);
      m.setSelected (true);
      c.add (m);

      z = new JRadioButton ();
      z.setText ("Ženski");
      z.setSize (80, 25);
      z.setLocation (10, 40);
      c.add (z);

      b = new ButtonGroup ();
      b.add (m);
      b.add (z);
...
```

Područja za tekst

Do sada smo se upoznali s okvirom za tekst. Za razliku od okvira za tekst, koji ćemo koristiti za unos manje količine teksta (ime, prezime, naslov,...), za unos veće količine teksta koristit ćemo tzv područja za tekst.

Slika 11 – 3: Primjer područja za tekst

Područje za tekst kreirat ćemo pomoću klase **`JTextArea`**.

Najčešće metode koje ćemo koristiti pri radu s klasom **`JTextArea`** su:

`public void setSize (int s, int v)` – definira veličinu područja za tekst.
`public void setLocation (int x, int y)` – definira poziciju područja za tekst na *Containeru*.
`public void setText (String s)` – definira tekst koji će biti upisan u području za tekst.
`public String getText ()` – vraća tekst koji je upisan u područje za tekst.
`public void setEditable (boolean b)` – ukoliko je *b* postavljen na *true* u područje za tekst će se moći unositi tekst te ga mijenjati, inače će područje za tekst biti neaktivno i nećemo na njemu moći ništanapraviti.
`public void addActionListener (Object o)` – dodaje listener na područje za tekst.

Trake za pomicanje

Napravimo li područje za tekst na način na koji smo to naučili u prethodnom primjeru malo ćemo se iznenaditi. Naviknuli smo da kada se jednom područje za tekst popuni, pojavljuju se trake za pomicanje teksta gore – dolje odnosno lijevo – desno. Ovdje ih medutim nema. Da bi se trake za pomicanje teksta pojavile oko područja za tekst, moramo ih eksplicitno postaviti. Kao i za bilo koji drugi element grafičkog korisničkog sučelja i za njih postoji posebna klasa koja ih generira. Klasa koja će generirati trake za pomak je **`JScrollPane (objekt o)`**, pri čemu je o objekt oko kojega postavljamo trake za pomicanje.

Najčešće metode koje ćemo koristiti pri radu s klasom **`JScrollPane`** su:

`public void setSize (int s, int v)` – definira veličinu traka za pomicanje.
`public void setLocation (int x, int y)` – definira poziciju područja za tekst na *Containeru*.

Primjer 11 – 4:
Kreirajmo područje za tekst oko kojega će se nalaziti trake za pomak.

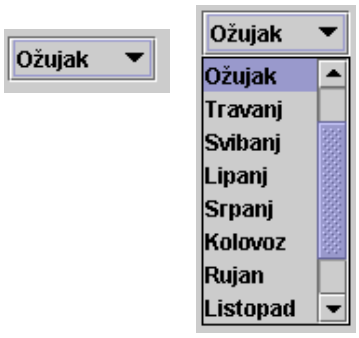
Rješenje:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class poglavlje11 extends JFrame
{
      private final int sirina = 220;
      private final int visina = 200;
      private Container c;
      private JScrollPane p;
      private JTextArea ta;

      public poglavlje11()
      {
            setTitle ("Editor");
            setSize (sirina, visina);
            setDefaultCloseOperation (EXIT_ON_CLOSE);

            c = getContentPane ();
            c.setLayout (new GridLayout (1, 1));

            ta = new JTextArea ();

            p = new JScrollPane (ta);
            c.add (p);

            setVisible (true);
      }

      public static void main (String[] s)
      {
            poglavlje11 z = new poglavlje11 ();
      }
}
```

Primjetimo da kada oko nekog elementa dodajemo trake za pomicanje, onda taj element ne trebamo posebno dodavatai na *Container*, već ga dodajemo automatski s trakama za pomicanje.

Padajuće liste

Padajuće liste ćemo koristiti kada trebamo odabrati jednu od opcija. Opcije se pojavljuju klikom na gumb s desne strane padajuće liste.



Slika 11 – 4: Primjer padajuće liste

Padajuću listu na ekranu generirat će klasa **JOptionPane**. Opcije padajuće liste moguće je dodati automatski pri kreiranju instance klase **JOptionPane**. Odnosno, jedan od konstruktora klase **JOptionPane** ima prametar tipa **Object[ ]**, a koji predstavlja opcije koje će se inicijalno pojaviti nakon kreiranja instance klase **JOptionPane**. Mi ćemo najčešće u padajuću listu dodavati tekst kao opcije pa će umjesto **Object [ ]** pisati **String [ ]**.

Isto tako klasa **JOptionPane** ima jos i niz drugih metoda. Neke od njih su:

`public void setSize (int s, int v)` – definira veličinu padajuće liste.
`public void setLocation (int x, int y)` – definira poziciju padajuće liste na *Containeru*.
`public void setSelectedIndex (int n)` – opcija s rednim brojem *n* će biti inicijalno označena.
`public void setSelectedItem (Object o)` – opcija *s* će biti inicijalno označena
`public int getSelectedIndex ()` – vraća redni broj označene opcije.
`public object getSelectedItem ()` – vraća označenu opciju.
`public void setEnabled (boolean b)` – ukoliko je *b* postavljen na *true* padajuća lista će biti aktivna, inače će biti neaktivan i nećemo nad njom moži napraviti nikakvu radnju.
`public void addItem (object o)` – dodaje opciju *o* na padajuću listu.
`public void addActionListener (Object o)` – dodaje listener na padajuću listu.

Primjer 11 – 5:
Dio programa koji će kreirati padajuću listu čije će stavke biti nazivi prvih 5 mjeseci u godini.

Rješenje:

```
...
private JComboBox cb;
private String[] mjeseci = {"Siječanj", "Veljača", "Ožujak", "Travanj",
"Svibanj"};
...
cb = new JComboBox (mjeseci);
cb.setLocation (10, 150);
```

```
cb.setSize (100, 25);
cb.setSelectedIndex (3);
cb.addActionListener (this);
cb.addItem ("...");
c.add (cb);
...
```

Izbornici

Izbornike možemo vidjeti u gotovo svim desktop aplikacijama. Pogodni su jer zauzimaju vrlo malo prostora pri vrhu prozora te na taj način povećavaju područje radne površine programa.
Za kreiranje izbornika koristit ćemo nekoliko klasa:

- **JMenuBar** – za kreiranje izborničke trake, koja se nalazi pri vrhu prozora
- **JMenu** – za kreiranje glavnih izbornika na izborničkoj traci (*File*, *Edit*,...)
- **JMenuItem** – za krieranje podstavki unutar glavnih izbornika (npr. unutar izbornika *File* najčešće se nalazi podstavke *Save*, *Open*,...)

Izborničku traku stavljamo direktno na prozor (*JFrame*) i za to koristimo metodu **`setJMenuBar`** klase *JFrame*.
Konstruktor klase *JMenu* ima jedan parametar i to je tekst koji će se ispisati na glavnom izborniku.
Glavne izbornike na izborničku traku dodajemo metodom **`add`** klase *JMenuBar*.
Podstavke unutar glavnog izbornika kreirat ćemo kreirajući instancu klase *JMenuItem*. Jedan od konstruktota klase *JMenuItem* kao parametar ima string koji u stvari predstavlja tekst koji će biti ispisan na podstavci. Podstavku ćemo na glavni izbornik dodati metodom **`add`** klase *JMenu*.
Obično se odabirom neke stavke unuta izbornika dogodi neka akcija, stoga ćemo nad stavkaka unutar izbornika pozivati metode **`addActionListener`**, na način kao što smo to radili nad drugim elementima.
Glavni izbornici i stavke unutar glavnih izbornika bit će poslagani onim redosljedom kako ih dodajemo odgovarajućim **`add`** metodama.

Primjer 11 – 6:
Unutar prozora kreirajmo izborničku traku koja će sadržavati dva glavna izbornika: *Dokument* i *Uređivanje*. Unutar izbornika *Dokument* trebaju se nalaziti stavke: *Novi*, *Spremi*, *Otvori* i *Kraj*; unutar izbornika *Uređivanje* trebaju se nalaziti stavke: *Označi sve*, *Izreži*, *Kopiraj* i *Zalijepi*.

Rješenje:

```
...
mb = new JMenuBar ();
setJMenuBar (mb);

dokument = new JMenu ("Dokument");
mb.add (dokument);

uredivanje = new JMenu ("Uređivanje");
```

```
mb.add (uredivanje);

novi = new JMenuItem ("Novi");
dokument.add (novi);

spremi = new JMenuItem ("Spremi");
dokument.add (spremi);

otvori = new JMenuItem ("Otvori");
dokument.add (otvori);

kraj = new JMenuItem ("Kraj");
dokument.add (kraj);

oznaci = new JMenuItem ("Oznaci sve");
uredivanje.add (oznaci);

kopiraj = new JMenuItem ("Kopiraj");
uredivanje.add (kopiraj);

izrezi = new JMenuItem ("Izrezi");
uredivanje.add (izrezi);

zalijepi = new JMenuItem ("Zalijepi");
uredivanje.add (zalijepi);
...
```