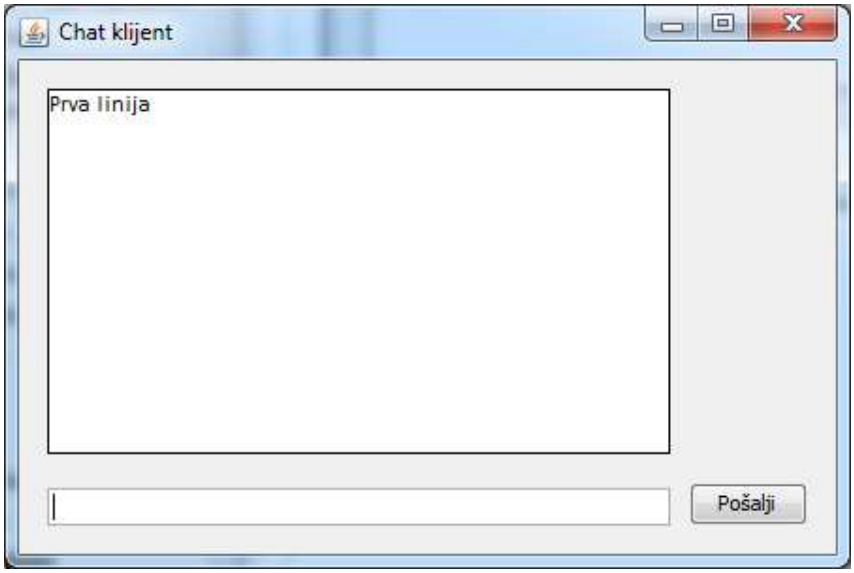
1. Chat klijent – layout i eventi

1.1. Cilj vježbe

Cilj vježbe je dizajnirati prozor chat klijenta. Prozor mora prilagođavati dimenzije elemenata u ovisnosti o veličini prozora. Treba i napisati kod koji će zalijepiti tekst napisan u donjem tekst polju (JText field) u gornje tekst polje (JText area) pritiskom na gumb „Pošalji“.



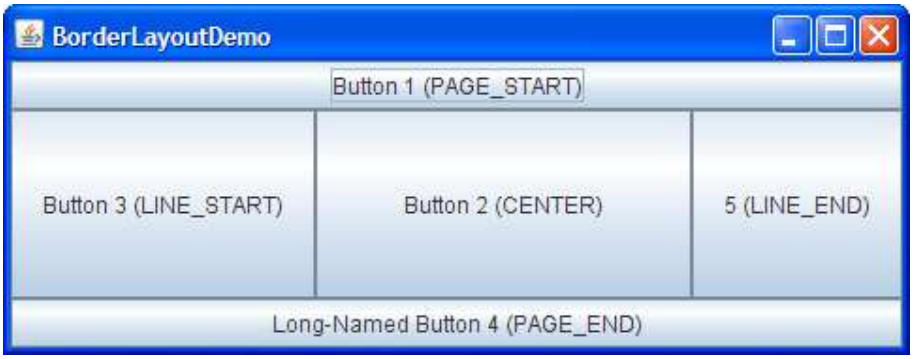
1.2. Layout manageri

Prilikom dizajniranja prozora treba izabrati layout managera. Layout manager je klasa koja se brine za raspoređivanje elemenata u kontejneru (JPanel) prilikom inicijalnog iscrtavanja i prilikom promjene veličine prozora. Ako nije izabran neki layout manager onda se elementi uvijek iscrtavaju na istim pozicijama koje su definirane prilikom kreiranje elementa (fixed layout).

U swingu se mogu izabrati različiti layout manageri kao što su:

BorderLayout

BorderLayout smješta Swing komponente na sjever, jug, istok i zapad kontejnera. A ostali prostor je smješten u središtu. Korisnik može dodati horizontalne i vertikalne razmake između područja. Svaki panel sadržaja (content pane) po defaultu koristi BorderLayout. U BorderLayout-u položaj komponente se definira sa drugim argumentom u metodi add kontejnera.

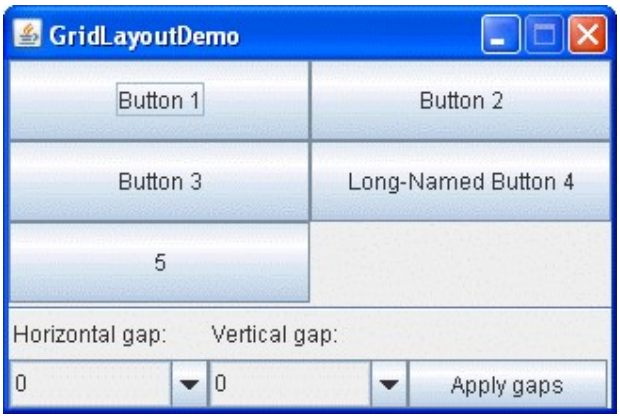


FlowLayout

FlowLayout smješta swing komponente s lijeva na desno sve dok ima slobodnog prostora. Kada mu nestane prostora on započinje novi redak i isto se pomiče s lijeva na desno. Svaka komponenta dobiva samo onoliko prostora koliko joj treba.

GridLayout

GridLayout smješta komponente u mrežu pravokutnika. Kontejner se podijeli na jednake pravokutnike i u svaki se pravokutnik može smjestiti samo jedna komponenta.

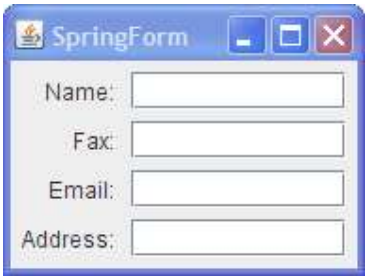


GridBagLayout

GridBagLayout smješta komponente u mrežu pravokutnika, ali sada svaka komponenta može zauzeti jedan ili više pravokutnika (područje prikaza). Komponente se poravnavaju horizontalno i vertikalno i ne moraju biti iste veličine.

SpringLayout

Fleksibilan layout manager napravljen da se koristi od strane GUI alata. Omogućava da se precizno odredi odnos između rubova komponenata (ograničenja, *constraints*). Npr. može se odrediti da je lijevi rub jedne komponente na određenoj udaljenosti od desnog ruba neke druge komponente. SpringLayout iscrtava elemente prema skupu ograničenja koja su definirana. Za ovu vježbu koristi će se ovaj layout manager.



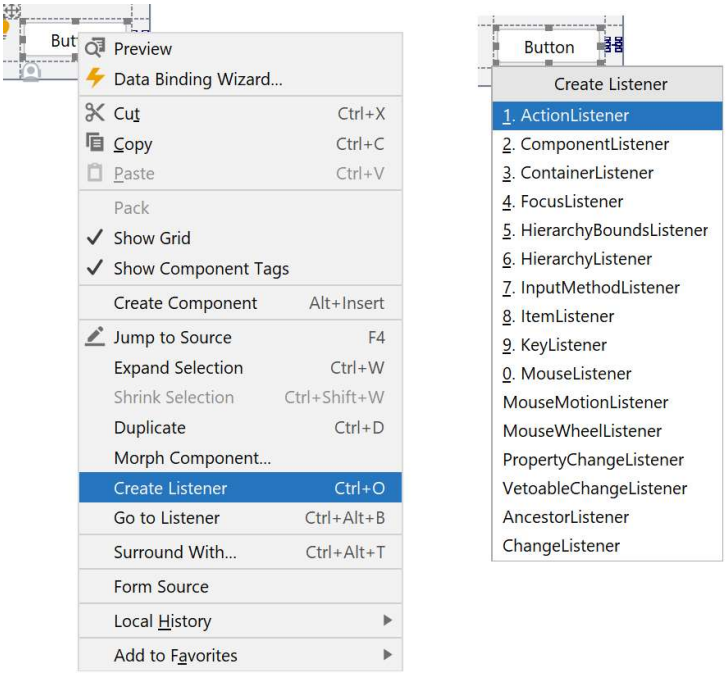
1.3. Obrada GUI događaja (event handlers)

Da bi prozor (program) bio funkcionalan moraju se obrađivati GUI događaji kao što su klik mišem na gumb (button). Događaji se u Javi obrađuju pomoću modela delegacije (Delegation model). Prema tom modelu izvor (source) generira događaj a slušač (listener) obrađuje taj događaj. Model se sastoji od tri dijela:

- **Izvor događaja** – izvor događaja su komponente (npr. gumbi). Događaji su Java objekti koje stvara komponenta i šalje ih kao argument slušačima.
- **Slušač događaja** – Slušač se povezuje sa izvorom događaja. Kada se dogodi neki događaj on poziva metodu koja pripada slušaču (metoda obrađuje događaj).
- **Sučelje** – sučelje sadrži metode koje slušač mora implementirati kako bi obradio događaj.

Na primjer kada korisnik klikne na gumb, gumb generira događaj (ActionEvent). Gumb poziva actionPerformed() metodu na svakom slušaču koji je povezan tj. registriran nad tim gumbom, i kao argument metodi šalje ActionEvent objekt. Gumb je izvor događaja, sučelje je ActionListener, a slušač je bilo koja metoda koja implementira ActionListener i registrira se nad gumbom.

Dva su načina da se napravi metoda obrade događaja. Prvi je direktno pisanje koda a drugi je putem „Properties“ prozora. Najprije se treba izabrati komponenta čije događaje želimo obraditi (Izvor događaja). Desnom tipkom miša se pojavljuje izbornik:



Nakon izbora događaja alat će generirati inicijalni kod koji će se izvršavati svaki put kada se desi izabrani događaj na elementu:

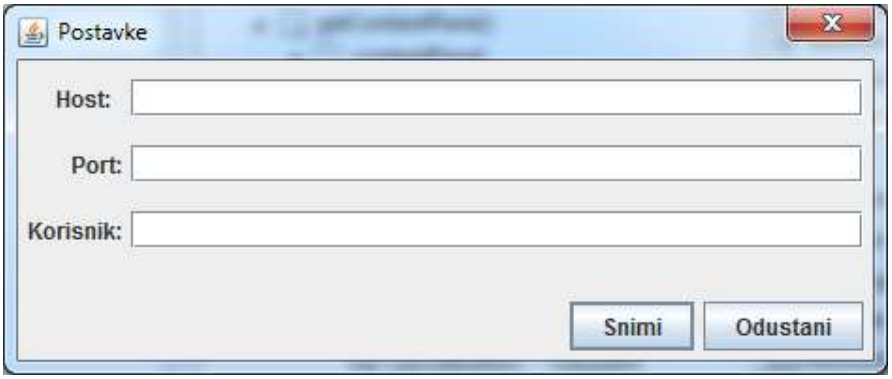
```
button1.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent actionEvent) {

    }
});
```

2. Chat klijent – konfiguracija

2.1. Cilj vježbe

Cilj vježbe je kreirati dodatni prozor za unos osnovnih postavki aplikacije. Postavke se mogu snimiti u datoteku chat.properties. Prilikom slijedećeg pokretanja aplikacije iz properties datoteke učitavaju se osnovne postavke:



2.2. Opis

2.2.1. Učitavanje i zapisivanje postavki

Postavke aplikacije su jedinstvene pa se potrebe dohvata i postavljanje postavki treba napraviti novu klasu **`UserConfig`** koja će imati statičke varijable za čuvanje pojedine postavke. Imamo tri postavke **host** (String), **port** (int) i **korisnik** (String). Varijable su definirane kao private pa se za svaku trebaju napraviti getteri i setteri (Code->Generate...->Getter and setters). Za potrebe čitanja i zapisivanja postavki u datoteku imamo slijedeće dvije metode:

```
private static final String propertiesFile = "chat.properties";
private static final String hostPropertieName = "host";
private static final String portPropertieName = "port";
private static final String userPropertieName = "user";

public static void loadParams() {
	Properties props = new Properties();
	InputStream is = null;
	// Najprije pokušavamo učitati iz lokalnog direktorija
	//
	try {
		File f = new File(propertiesFile);
		is = new FileInputStream(f);
	} catch (Exception e) {
		e.printStackTrace();
		is = null;
	}

	try {
		// pokušavaju se učitati parametri
		props.load(is);
	} catch (Exception e) {
```

```
        }
        // prvi parametar: naziv postavke
        // drugi parametar: ako nije nađena vrijednost onda se vraća drugi
        // parametar
        host = props.getProperty(hostPropertieName, "192.168.0.1");
        port = new Integer(props.getProperty(portPropertieName, "8080"));
        korisnik = props.getProperty(userPropertieName, "anonymous");
    }

    public static void saveParamChanges() {
        try {
            Properties props = new Properties();
            props.setProperty(hostPropertieName, host);
            props.setProperty(portPropertieName, "" + port);
            props.setProperty(userPropertieName, korisnik);
            File f = new File(propertiesFile);
            OutputStream out = new FileOutputStream(f);
            props.store(out, "Opcionalni header komentar");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

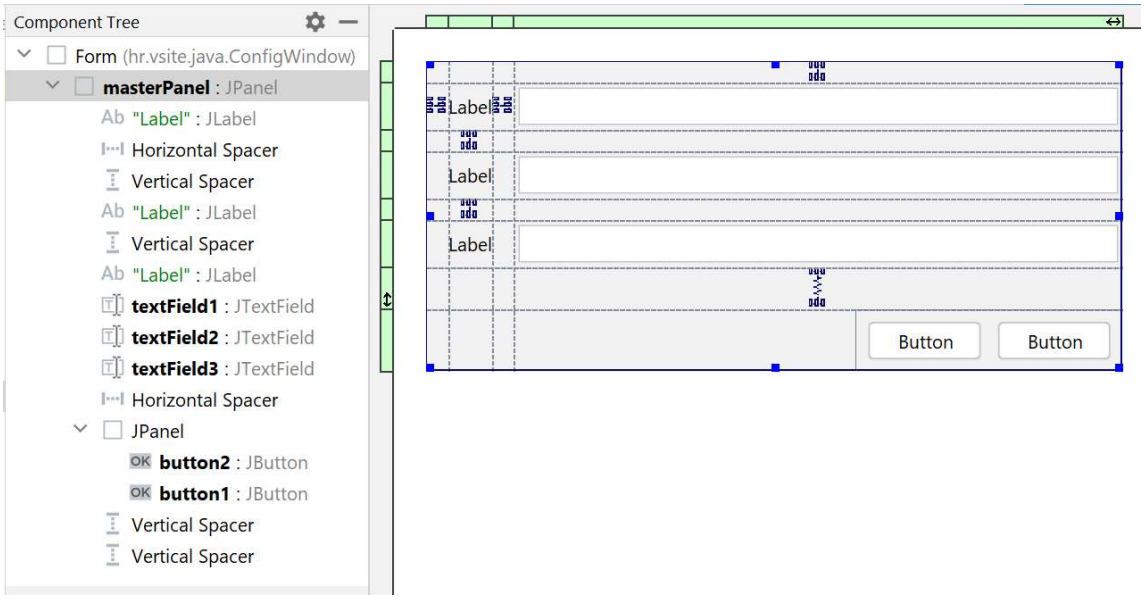
Postavke trebaju biti dostupne prije prvog korištenja novo kreirane klase pa se mora napraviti i statički inicijalizatorski blok u kojem će se pozivati metoda za učitavanje postavki (**loadParams**()). Datoteka sa postavkama *chat.properties* se treba nalaziti u direktoriju iz kojeg se pokreće aplikacija.

2.2.2. Prozor za uređivanje postavki

Za potrebe uređivanje postavki potrebno je kreirati novi prozor. Pošto je to jednostavan prozor i želimo ga pokrenuti u „modal“ načinu (fokus se ne može prebaciti na niti jedan drugi prozor aplikacij) novi prozor ćemo pokrenuti kao JDialog. U Intelij kreiramo novu GUI formu sa GridBagLayout-om ali potrebno je modificirati generiranu main metodu da se ne koristi JForm nego JDialog klasa:

```
public static void main(String[] args) {
    JDialog dialog = new JDialog();
    dialog.setContentPane(new ConfigWindow().masterPanel);
    dialog.setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
    dialog.pack();
    dialog.setVisible(true);
}
```

Potrebno je posložiti komponente na formu na slijedeći način:



Za svaku komponentu može se i definirati „težina“ (weight) pojedinog retka ili stupca. Težina određuje koliko će slobodnog prostora taj redak ili stupac preuzeti. Ako je težina 0 onda se redak/stupac neće preuzeti nikakav slobodan prostor. Ako svi retci i stupci imaju težinu 0 onda se slobodan prostor podjeli na dva jednaka dijela koji su pozicionirani ispred i nakon redaka/stupaca.

Ono što želimo postići je da drugi stupac i treći redak preuzmu cijelo slobodno područje. To se postiže tako se toj komponenti postavi težina veća od nula.:

Property	Value
field name	**textField1**
Custom Create	☐
Horizontal Align	Fill
Vertical Align	Center
⊞ Insets	[0, 0, 0, 0]
Weight X	**1.0**
Weight Y	0.0
Ipad X	0
Ipad Y	0
⊟ Client Properties	

2.2.3. Upravljenje prozorom

Prije zatvaranja ekrana i ako imamo promjene postavki koje nisu snimljene treba postaviti pitanje korisniku da li je siguran da želi zatvoriti prozor bez simanja. Da bi to postigli moramo definirati što će se desiti prilikom zatvaranja prozora. U swingu se to može postaviti putem metode prozora **setDefaultCloseOperation(int).** Metoda prima samo jedan parametar tipa int a vrijednosti koje se

mogu koristiti se su definirane u WindowConstants klasi (postoje kopije vrijednosti sa istim imenom i u klasama JFrame i JDialog):

– DO_NOTHING_ON_CLOSE – ignorira zahtjev za zatvaranjem

– HIDE_ON_CLOSE CLOSE– samo sakrij prozor

– DISPOSE_ON_CLOSE – sakrij prozor pa oslobodi sve njegove resurse

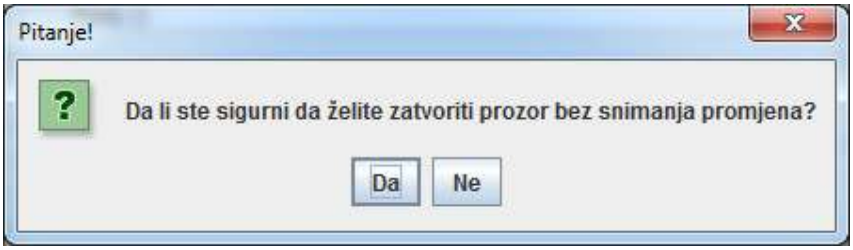
– EXIT_ON_CLOSE – terminiraj samu aplikaciju (može izazvati iznimku!) (ova opcija nije podržana na JDialog prozoru)

Pošto mi želimo imati konstrolu kada ćemo zatvoriti prozor treba se postaviti opcija:

```
setDefaultCloseOperation(JDialog.DO_NOTHING_ON_CLOSE);
```

Tada se prozor neće zatvoriti nego će se samo generirati događaj (event) window->closing. To je događaj koji mi moramo obraditi i tu postaviti pitanje.

Da bi se pratilo da li je došlo do promjena u postavkama može se obrađivati događaje focus gained i focus lost na svakom tekst polju te pratiti da li je bilo promjena u polje. Drugi je način da se provjeravaju vrijednosti na ekranu sa vrijednostima u Config klasi (spremljene postavke). Ako je bilo promjena onda se na zatvaranju prozora (u obradi događaja) treba postaviti pitanje inače se zatvara prozor bez pitanja:



Takve male prozore za postavljanje poruka se mogu pozivati sa pomoćnom klasom JOptionPane. Poziv koji otvara ovakav prozor izgleda ovako:

```
int confirmed = JOptionPane.showOptionDialog(contentPanel,
"Da li ste sigurni da želite zatvoriti prozor bez snimanja promjena?", "Pitanje!",
JOptionPane.YES_NO_OPTION, JOptionPane.QUESTION_MESSAGE,
null, new String[] { "Da", "Ne" }, "Ne");
```

Prvi parametar - element (Component) iznad kojeg će se pozicionirati novi pozor.
Drugi parametar - tekst koji se ispisuje na ekranu
Treći parametar - naslov prozora
Četvrti parametar – Koje će se opcije (gumbi) iscrtati na ekranu: DEFAULT_OPTION, YES_NO_OPTION, YES_NO_CANCEL_OPTION, OK_CANCEL_OPTION (DEFAUL_OPTION je isto kao i OK_CANCEL_OPTION)
Peti parametar – tip ekrana: ERROR_MESSAGE, INFORMATION_MESSAGE, WARNING_MESSAGE, QUESTION_MESSAGE, PLAIN_MESSAGE
Šesti parametar – ikona koja se prikazuje lijevo od teksta. Proslijeđujemo null da se icrta zadana ikona
Sedmi parametar – niz stringova koji sadržava nazive koji se pojavljuju na gumbima (ako nije definira ispisuju se zadane vrijednosti: OK, Cancel, Yes, No, ...)

Osmi parametar- Označava koja je default opcija (koja je opcija ako se npr. pritisne tipka enter)

U ovisnosti o opcijama koje su iscrtane (četvrti parametar) i o tome što je korisnik izabrao vraća se jedna od ovih vrijednosti:

YES_OPTION, NO_OPTION, CANCEL_OPTION, OK_OPTION, CLOSED_OPTION

(više o JOptionPane http://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html)

Samo zatvaranje prozora se radi sa metodom **dispose()** i tu metodu treba pozvati ako je korisnik siguran da želi izaći iz ekrana bez spremanja promjena ili ako nije niti bilo promjena.

Pritisak na gumb „Odustani“ treba obaviti sve operacije koje se dešavaju prilikom zatvaranja prozora tako da je najbolje da se kod koji radi provjeru, postavljanje pitanja i zatvaranja prozora izdvoji u posebnu metodu. Ta metoda će se pozivati i prilikom obrade događaja zatvaranja prozora i obrade događaja kojeg generira gumb „Odustani“.

Gumb „Spremi“ treba spremiti postavke u datoteku (pomoću klase koja čuva postavke).

U glavnom prozoru Chat klijenta treba dodati novi gumb koji će otvoriti novi prozor. Prozor se otvara na slijedeći način:

```
//ConfigWindow je naziv klase koja nasljeđuje JDialog
ConfigWindow dialog = new ConfigWindow();
dialog.setVisible(true);
```