

Развој графичких корисничких апликација у програмском језику Java

Душан Тошић

Термин Графички кориснички интерфејс (надаље краће ГКИ) односи се на низ графичких елемената преко којих корисник (човек) обавља комуникацију са рачунаром. У графичке елменте (које ћемо најчешће називати графичке компоненте) спадају: прозори, иконе, дугмад, текстовна поља итд. Важну улогу код ГКИ-а имају уређаји, као што је миш, који омогућавају обележавање, показивање и избор графичких компоненти.

Самостални рачунарски програм (апликација) који човек користи обављајући комуникацију са рачунаром преко ГКИ –а, називаћемо ГКИ–апликација. Скоро сви савремени оперативни системи засновани су на ГКИ-у. Природно је да и програми који се покрећу помоћу ових оперативних система буду засновани на ГКИ-у. Прављење ГКИ-апликација може да буде мучан посао уколико сваку компоненту треба посебно дизајнирати и смештати на одређено место. Зато већина савремених програмских језика омогућава поједностављење поступка оперисања са компонентама ГКИ-а обезбеђујући низ класа, интерфејса и метода за рад са графичким компонентама.

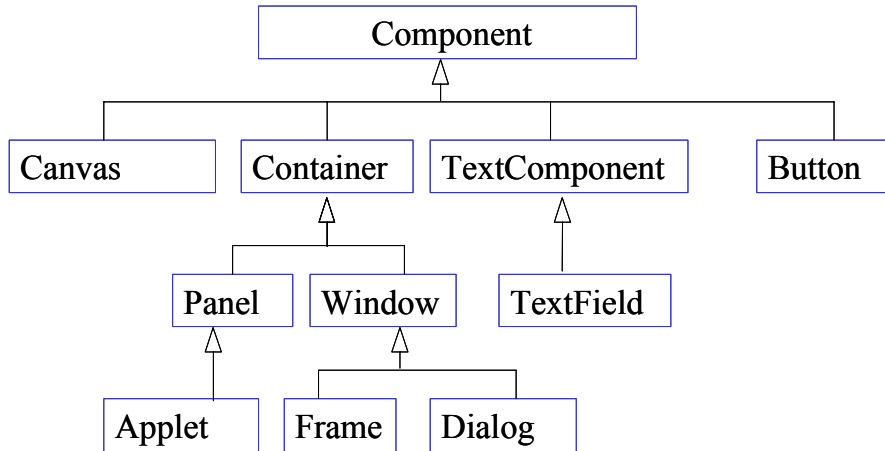
Програмски језик Java подржава развој ГКИ-апликација још од настанка. У првим верзијама Java-е развој ГКИ-апликација омогућавао је `awt`-пакет који је припадао скупу основних пакета. Међутим, овај пакет је пре свега коришћен за прављење аплета (несамосталних програма који су обично покретани из HTML-докумената). Тек са увођењем пакета `javax.swing` (који постаје стандардни од верзије 1.2) повећане су могућности коришћења ГКИ-а и омогућен лакши развој ГКИ-апликација. Пакет `javax.swing` је направљен тако да представља надградњу пакета `java.awt`. У ГКИ-апликацијама, поред `swing`-компоненти, можемо користити и `awt`-компоненте. ГКИ-компоненте се на исти начин користе у `javax.swing` пакету као и у `awt`-пакету.

1. Компоненте `awt`-пакета

Пакет `awt`, између осталих, садржи класу `Component`, а која обухвата (представља наткласу) низ поткласа које омогућавају рад са разним компонентама. На слици 1. приказане су најзначајније поткласе класе `Component`.

У једној ГКИ-апликацији може се појавити више компоненти и оне се смештају у неку другу графичку компоненту. Компоненту која може да садржи једну или више других графичких компоненти, називаћемо контејнерска компонента (краће, контејнер). Уколико има више компоненти у једном контејнеру, важно је како су распоређене (да ли су у угловима контејнера, при врху, на средини итд.). За распоређивање компоненти у контејнерима у Java-и се користе посебни објекти који се називају менаџери распоређивања (краће, распоређивачи). Распоређивачи одређују распоред компоненти у контејнеру, али и изглед појединих

компоненти. (Напоменимо да распоређивачи нису присутни у свим програмским језицима који омогућавају развој ГКИ-апликација).



Слика 1.

У компоненте контејнерског типа спадају само оне које су поткласе класе Container, док су остале компоненте неконтејнерског типа, тј. не могу садржати друге компоненте. У компоненте неконтејнерског типа спадају: лабеле, дугмад, контролна дугмад, радио дугмад, текстовна поља, текстовне зоне, слајдери итд. Свака од компоненти је специфична и за сваку од њих постоји један или више конструктора. Мађутим, начин рада са свим коментентама је исти, треба да декларишемо њено име, да је иницијализујемо (креирамо) и додамо у контејнер. У следећем примеру је показано како се то може урадити.

```

public void init(){
    Button dugme;
    dugme = new Button("STRAT");
    add(dugme);
}
    
```

Декларација компоненте није обавезна па уместо 3 наредбе у претходном методу, можемо писати само једну:

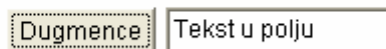
```

add(new Button("START"));
    
```

али у овом случају компонента неће бити именована.

У примеру 1 као контејнер коришћен је аплет и у њга су смештене 3 компоненте: лабела, дугме и текстовно поље. Покретањем овог аплета приказују се ове 3 компоненте као на слици 2.

Ovde su 3 komponente: labela, dugme i tekstovno polje



Слика 2.

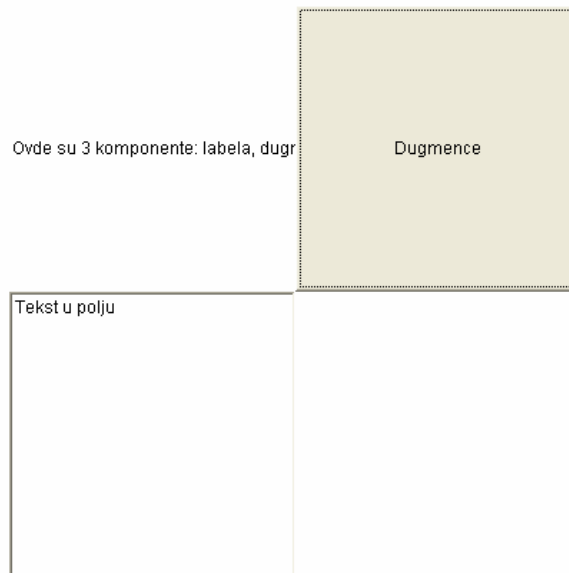
Пример 1.

```
import java.awt.*;
public class Prim1 extends java.applet.Applet{
    public void init(){
        Label labela;
        labela = new Label("Ovde su 3 komponente: labela, dugme i tekstovno polje");
        add(labela);
        Button dugme = new Button("Dugmence");
        add(dugme);
        add(new TextField("Tekst u polju"));
    }
}
```

Да би могле да се користе класе пакета `awt`, треба да се увезе овај пакет преко наредбе `import java.awt.*`. У примеру 1 није (експлицитно) наведен ни један распоређивач. Уколико се не наведе експлицитно распоређивач, користи се подразумевани распоређивач: *FlowLayout*. Овај распоређивач ређа компоненте једну за другом у оном редоследу како се креирају. Међутим, поред овог распоређивача, постоје и други, као што су: *GridLayout*, *BorderLayout*, *CardLayout* и *GridbagLayout*. Ако бисмо у примеру 1 у метод `init()`, додали наредбу:

```
setLayout(new GridLayout(2,2));
```

онда саопштавамо да користимо мрежни распоређивач (*GridLayout*), при чему мрежа има 2×2 ћалија. Тада бисмо добили приказ као на слици 3.



Слика 3

Из примера се може видети да распоређивачи не распоређују само компоненте, већ мењају и њихов изглед.

Креирањем и убацивањем компоненти у контејнер, не значи да смо креирали и апликацију. Да бисмо један програм (у којем се појављује елементи

ГКИ-а) могли да назовемо и апликацијом, потребно је да, на неки начин, можемо и да употребимо ове компоненте. Употреба компоненти омогућава се повезивањем компоненти са догађајима (од прирпде компоненти зависи како ће бити употребљене, тј. не морају увек бити везане за догађаје). Рад са догађајима у Јава-и заснива се на коришћењу интерфејса ослушкивача догађаја. Ови интерфејси се налазе у пакету `java.awt.event`. Они садрже декларације метода чијем предефинисањем описујемо како се поједине компоненте повезују са догађајима. У примеру 2 дугме на којем пише "Krug" повезано се са интерфејском `ActionListener` помоћу наредбе:

```
krugDugme.addActionListener(this);
```

Интерфејс `ActionListener` садржи метод `actionPerformed()` чијим предефинисањем смо омогућили да се кликом на дугме "Krug" увек нацрта црвени круг пречника 60 пиксела, на случајним позицијама у прозору аплета.

Пример 2.

```
import java.awt.*;
import java.awt.event.*;
public class SlucKrug extends java.applet.Applet
    implements ActionListener{
    public void init(){
        Button krugDugme = new Button("Krug");
        add(krugDugme);
        krugDugme.addActionListener(this);
    }
    public void paint (Graphics g) {
        int x,y, vel=300;
        x = (int) (vel*Math.random());
        y = (int) (vel*Math.random());
        g.setColor(Color.red);
        g.fillOval(x, y, 60, 60);
    }
    public void actionPerformed(ActionEvent dog){
        repaint();
    }
}
```

Покретањем наведеног аплета (нпр. из неке HTML-презентације), добија се приказ попут оног на слици 4 (само је положај круга другачији).

Krug

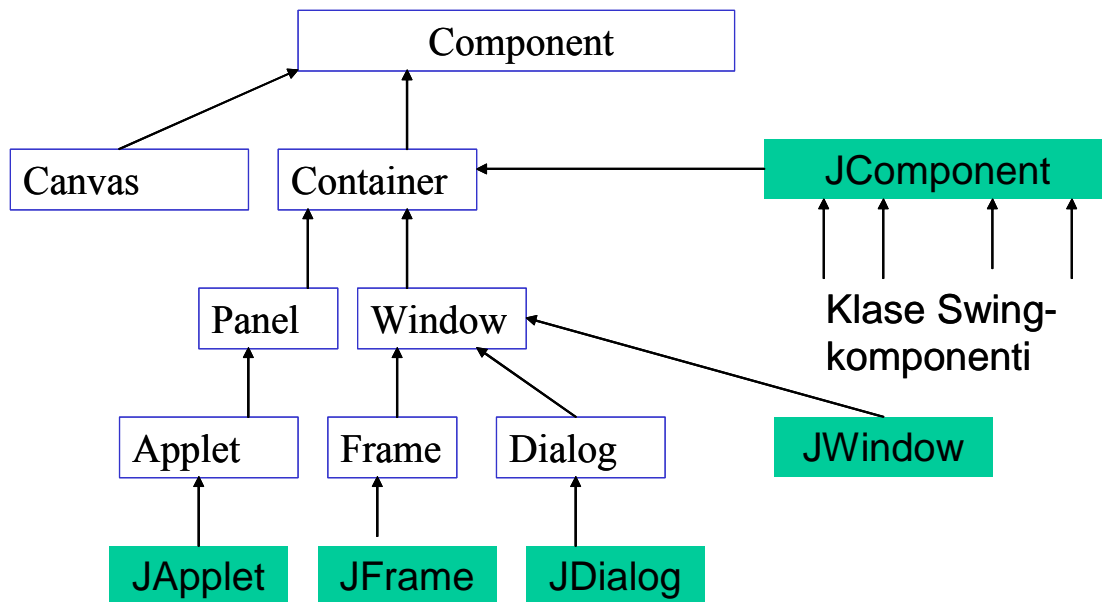


Слика 4

2. Коришћење *javax.swing* пакета

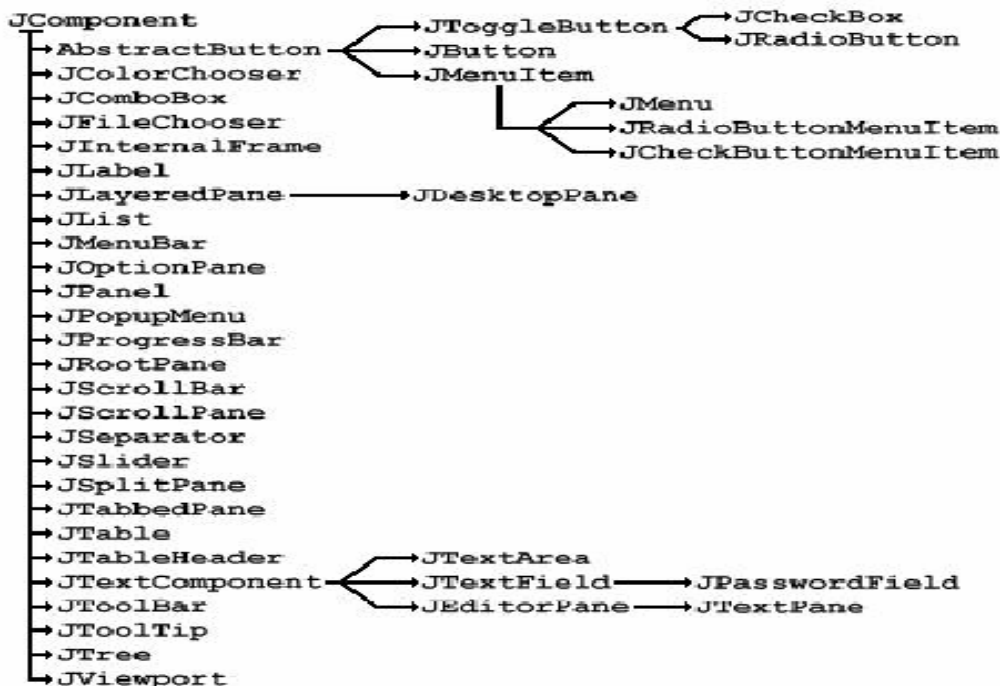
Swing-класе су функционалније и флексибилније у поређењу са класама из *awt*-пакета. Класе Swing-пакета чине део ГКИ-а у Java-и. ГКИ у Java-и је заснован на скупу пакета који носи назив JFC (Java Foundation Class). Поред Swing-а, овај скуп класа обезбеђују пакете за 2D графику, затим класе за повлачење и спуштање компоненти итд. Најважнији пакет у оквиру JFC-а је Swing-пакет. Он обезбеђује мању зависност ГКИ-апликација од платформи на којима се извршавају, омогућава да се прилагоди изглед компоненти околини у којој се користе, подржава MVC (Model View Controllers) архитектуру итд.

Већ смо рекли да су класе Swing-пакета направљене тако да представљају надградњу класа *awt*-пакета. Имена свих класа из Swing-пакета почињу словом J. По томе се ове класе разликују од одговарајућих класа из *awt*-пакета. На слици 5 приказане су класе Swing и *awt*-пакета.



Слика 5.

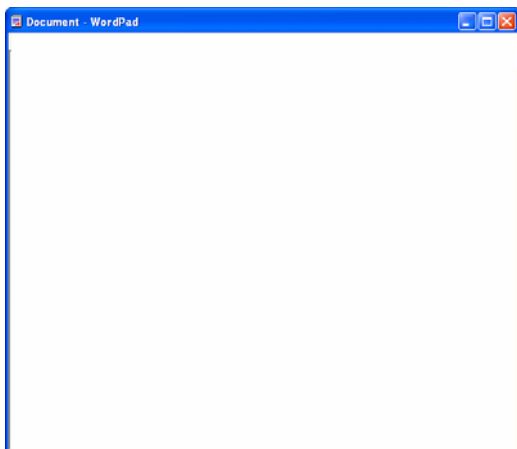
Са слике 5 се може запазити да је класа **JComponent** поткласа класе **Container**, што значи да су све компоненте у Swing-у контејнерског типа, тј. у сваку компоненту се може додати нека друга. Сама класа **JComponent** садржи знатно већи број компоненти, него ли одговарајућа класа у *awt*-пакету. На слици 6 приказане су класе које припадају класи **JComponent**. Запажа се да класа **JComponent** садржи низ нових компоненти, као што су: **JOptionPane**, **JTable**, **JTree**, итд. Новододате класе омогућавају прављење разноврсних апликација и пружају већу удобност у раду. Природно, понегде се појављују и разлике у коришћењу компоненти. На пример, компоненти **TextArea** из *awt*-пакета аутоматски су додавани слајдери (клизачи) ако је ред дужи од видљивог дела зоне (односно, ако има редова који се не виде), док код **JTextArea** то није случај, већ корисник мора сам да дода клизаче ако жели да се оне појаве у текстовној зони.



Слика 6.

3. Креирање ГКИ-апликација

Главни графички елемент ГКИ-апликације је прозор. Све обраде и примене апликације одвијају се преко једног или више прозора на екрану монитора. Дакле, један од првих корака у креирању ГКИ-апликације је обезбеђивање главног прозора. Пожељно је да тај прозор садржи назив апликације, дугмад за затварање, минимизацију и промену величине, као и иконицу апликације (слика 7).



Слика 7.

Главни прозор представља оквир у који ће бити смештене све остале компоненте. За креирање овог прозора користи се класа JFrame. То је класа која садржи преко 200 метода за рад са прозорима, као и разне константе. Прозор из класе JFrame је основни и не садржи се ни у једном другом прозору. Приликом креирања основног прозора треба задати димензије прозора, као и позицију прозора. Уколико се не задају димензије прозора, претпоставиће се да имају вредност 0, и он неће бити приказан.

Креирање основног прозора апликације (димензија 400x200 са координатама горњег левог угла (50, 50)) може се извршити помоћу програма из примера 3.

Пример 3.

```
import javax.swing.*;
public class Gkia1 {
    static JFrame prozor = new JFrame("Ovo je naslov aplikacije");
    public static void main(String[] args) {
        prozor.setBounds(50,50, 400,200); // Pocetak i velicina
        prozor.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        prozor.setVisible(true);
    }
}
```

У наведеном програму нема компоненти са којима се може нешто радити па се он и не може назвати апликацијом (тј. он представља тривијалну апликацију). Ако замислимо да ивице креираног прозора представљају рам, онда између њих можемо поставити панел и на њега додавати разне компоненте. Дакле, следећа важна компонента (контејнерског типа) је панел у који можемо убацити друге компоненте. У примеру 4, основном прозору из примера 3, додати су панел и 2 компоненте: дугме и текстовно поље.

Пример 4.

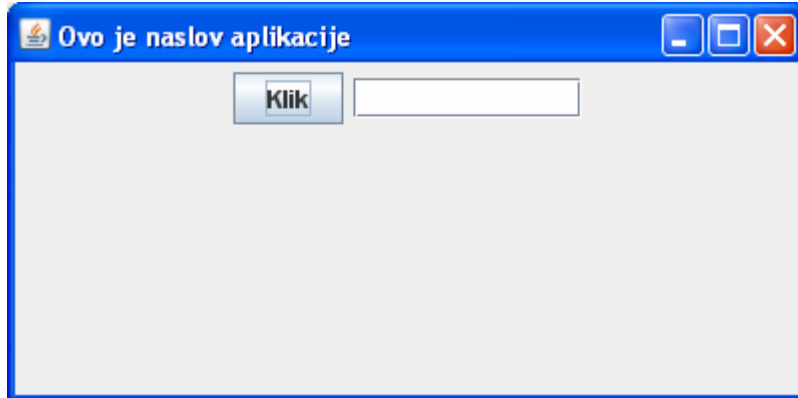
```
import javax.swing.*;
public class Gkia2 {
    static JFrame prozor = new JFrame("Ovo je naslov aplikacije");
    public static void main(String[] args) {
        prozor.setBounds(50,50, 400,200); // Pocetak i velicina

        JButton prvo = new JButton("Klik");
        JTextField polje = new JTextField(10);
        JPanel pan = new JPanel();
        pan.add(prvo);
        pan.add(polje);
        prozor.setContentPane(pan);

        prozor.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        prozor.setVisible(true);
    }
}
```

Покретањем наведеног програма добија се приказ прозора као на слици 8. Овде имамо прозор са панелом на којем се налазе компоненте. Кликом на дугме Klik или притиском тастера ENTER у текстовном пољу, ништа се не дешава. Стога ни овај

програм не можемо назвати правом апликацијом јер је основно обележје апликације могућност примене.



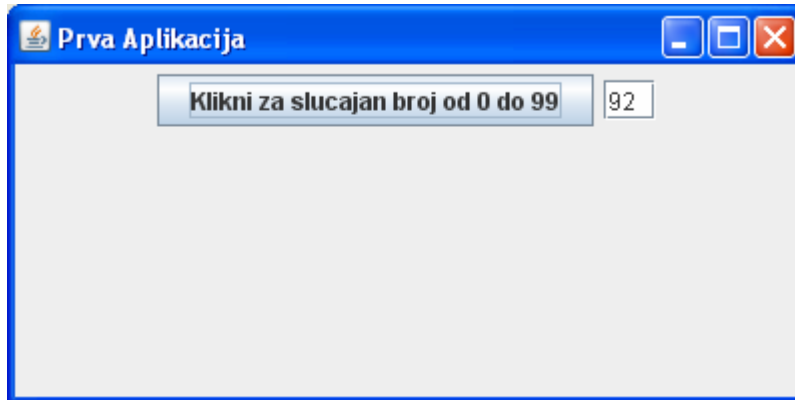
Слика 8.

Уколико компоненте у панелу повежемо са догађајима, онда бисмо добили праву апликацију. То је урађено у примеру 5 тако што су извршене модификације примера 4. (Направљен је конструктор `Gkia3` у којем се креирају компоненте. Компонента `prvo` се везује за догађај генерисања случајног броја од 0 до 99 и уписа добијеног броја у текстовно поље.

Пример 5.

```
import javax.swing.*;
import java.awt.event.*;
public class Gkia3 extends JFrame implements ActionListener {
    JTextField polje;
    public Gkia3(String s) {
        super(s);
        JButton prvo = new JButton("Klikni za slucajan broj od 0 do 99");
        polje = new JTextField(2);
        prvo.addActionListener(this);
        JPanel pan = new JPanel();
        pan.add(prvo);
        pan.add(polje);
        this.setContentPane(pan);
    }
    public void actionPerformed(ActionEvent dog) {
        polje.setText((int)(100*Math.random())+" ");
    }
    public static void main(String[] args) {
        Gkia3 prozor= new Gkia3("Prva Aplikacija");
        prozor.setBounds(50,50, 400,200);
        prozor.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        prozor.setVisible(true);
    }
}
```

За наведени програм можемо да кажемо да представља једноставну ГКИ-апликацију. Када покренемо програм, кликом на дугме **Klikni za slucajan broj od 0 do 99**, случајан број из интервала [0,99] биће уписан у текстовно поље на панелу (слика 9).



Слика 9.

4. Примери једноставних ГКИ-апликација

Скоро свака апликација која није заснована на ГКИ-у може се модификовати тако да постане ГКИ-апликација. (Битно обележје ГКИ-апликација је коришћење графичких компоненти. Од природе апликације зависи које компоненте ће бити коришћене.) Нека имамо једноставан задатак да направимо програм у којем се сабирају 2 случајно генерисана броја и резултат штампа на излазу. То можемо урадити као у примеру 6.

Пример 6.

```
public class Sab{
    public static void main(String[] args) {
        double p1, p2, z;
        p1 = Math.random();
        p2 = Math.random();
        z= p1+p2;
        System.out.println(" p1 = "+p1+" p2 = "+p2 + "p1+p2 = " +z);
    }
}
```

Ово је типична не ГКИ-апликација. Резултат се штампа иза командне линије са које је програм покренут (слика 10).

```
C:\Documents and Settings\Dusan.MF-E087DC064E22\Desktop\Proba>java Sab
p1 = 0.28441915344853386 p2 = 0.8029526270151777p1+p2 = 1.0873717804637115
C:\Documents and Settings\Dusan.MF-E087DC064E22\Desktop\Proba>
```

Слика 10

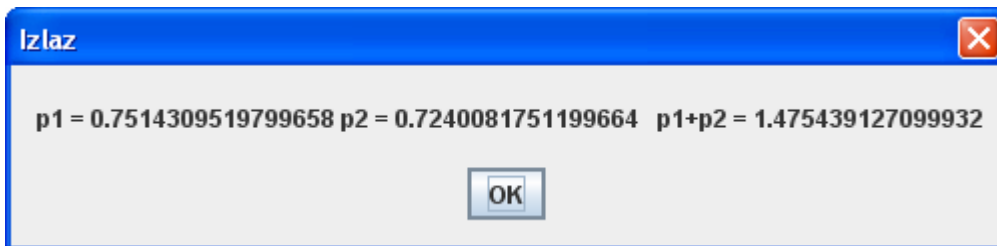
Коришћењем класе JOptionPane и метода showMessageDialog(...), ову апликацију претварамо у ГКИ-апликацију. Модификовани програм приказан је у примеру 7, а добијени резултат на слици 11.

Пример 7.

```
import javax.swing.JOptionPane;

public class Sab1 {
    public static void main(String[] args) {
        double p1, p2, z;
        p1 = Math.random();
        p2 = Math.random();
        z = p1+p2;
        JOptionPane.showMessageDialog(
            null, " p1 = "+p1+" p2 = "+p2 + "  p1+p2 = " +z,
            "Izlaz", JOptionPane.PLAIN_MESSAGE );
    }
}
```

Покретањем наведеног програма добија се резултат попут следећег:



Слика 11.

За креирање једноставних ГКИ-апликација можемо користити ма које компоненте. Нека имамо задатак да израчунамо и прикажемо факторијеле свих природних бројева мањих од n. (Природно је увести неко ограничење за n – нека n буде мање од 20 да бисмо радили са дугим целим бројевима, не користећи низове.) Решење постављеног задатка је приказано у примеру 8.

Пример 8.

```
import javax.swing.JOptionPane;
import javax.swing.JTextArea;

public class Faktoriјeli {
    public static void main( String args[] )
    {
        String sb;
        long fk, i, n;
```

```

JTextArea izlaznaZona = new JTextArea();
sb = JOptionPane.showInputDialog( "Do kojeg broja zelite faktorijele (n<20)" );
n = Integer.parseInt( sb );
izlaznaZona.setText( "n\tFaktorijel(n)\n" );
fk=1;
for ( i = 1; i <= n; i++ ) {
    fk*=i;
    izlaznaZona.append( i + "\t" + fk + "\n");
}
JOptionPane.showMessageDialog( null, izlaznaZona,
    "Faktorijeli do 12", JOptionPane.INFORMATION_MESSAGE );
}
}

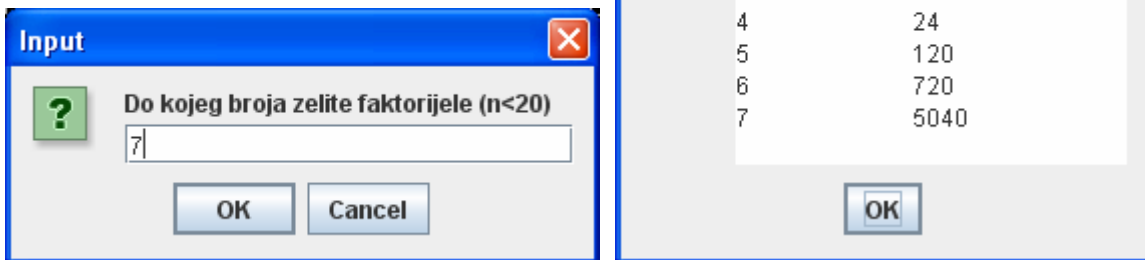
```

У наведеном решењу користе се 2 графичке компоненте: `JOptionPane` и `JTextArea`.

Најпре се од корисника тражи да унесе број до којег жели факторијеле преко комуникационог прозора добијеног помоћу метода

`JOptionPane.showInputDialog(...)`.

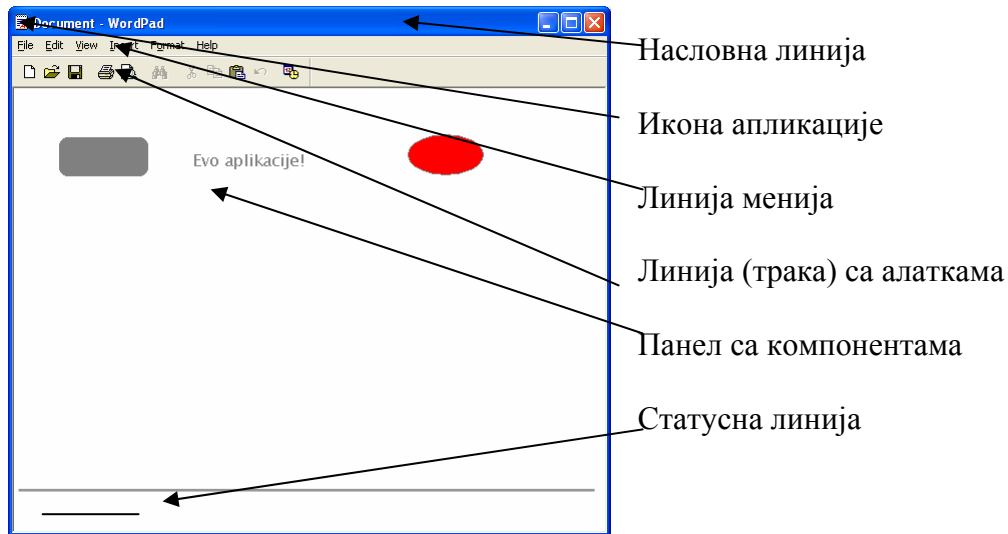
Потом се у текстовној зони приказују факторијели (слика 12).



Слика 12.

5. Делови сложенијих ГКИ-апликације

Иоле сложеније ГКИ-апликације, поред основног панела са компонентама, садрже: линију менија, линију са алатима, статусну линију, ... (слика 13). Коришћењем програмског језика Java сви ови елементи могу се додати ГКИ-апликацији. Линије менија могу садржати падајуће меније са пречицама и убрзивачима за приступ командама преко тастатуре. Постоји читав низ класа и метода за креирање и додавање ових елемената апликацији.



Слика 13.

Да бисмо креирали линију менија користимо следеће наредбе:

```
JMenuBar meniLinija = new JMenuBar();
prozor.setJMenuBar(meniLinija);
```

За додавање команди линији менија (нпр. команде File и Edit) користимо клаузу Jmenu:

```
JMenu dat = new JMenu("File");
meniLinija.add(dat);
JMenu edi = new JMenu("Edit");
meniLinija.add(edi);
```

Ако желимо да креирамо поткоманде у командама, тј. да формирамо падајући мени за неку команду, користимо класу JMenuItem. На пример, ако желимо да команди File, додамо поткоманде: Nova, Otvori и Print, тако да последња буде одвојена сепаратором од прве две, писаћемо :

```
JMenuItem nova = dat.add("Nova");
JMenuItem otvori = new JMenuItem("Otvori");
dat.add(otvori);
dat.addSeparator();
JMenuItem stampa = dat.add("Print");
```

Ако све ове команде убацимо у програм из примера 4, главни прозор апликације садржаће и линију менија. Цео програм приказан је у примеру 9.

Пример 9.

```
import javax.swing.*;
public class Gkia5 {
    static JFrame prozor = new JFrame("Aplikacija sa linijom menija");
    public static void main(String[] args) {
        prozor.setBounds(50,50, 400,200); // Pocetak i velicina

        JMenuBar meniLinija = new JMenuBar();
```

```

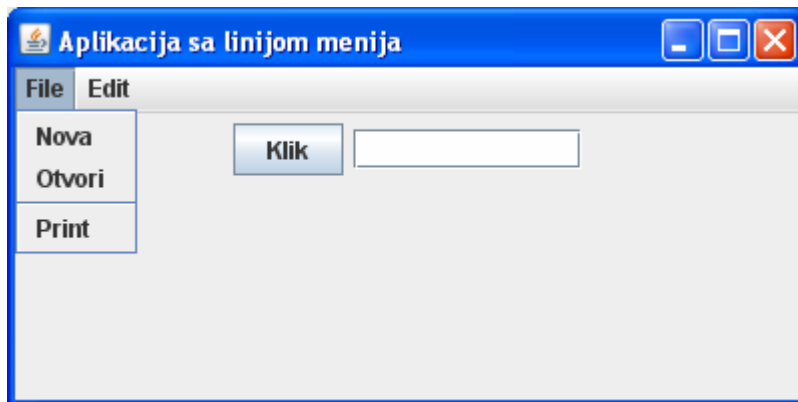
prozor.setJMenuBar(meniLinija);
JMenu dat = new JMenu("File");
meniLinija.add(dat);
JMenu edi = new JMenu("Edit");
meniLinija.add(edi);
JMenuItem nova = dat.add("Nova");
JMenuItem otvori = new JMenuItem("Otvori");
dat.add(otvori);
dat.addSeparator();
JMenuItem stampa = dat.add("Print");

JButton prvo = new JButton("Klik");
JTextField polje = new JTextField(10);
JPanel pan = new JPanel();
pan.add(prvo);
pan.add(polje);
prozor.setContentPane(pan);

prozor.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
prozor.setVisible(true);
}
}

```

Покретањем овог програма и кликом на команду File, добија се приказ као на слици 14.



Слика 14.

Све компоненте у линији менија (и падајућем менију) су елементи класе `AbstractButton` па те компоненте можемо третирати као дугмад. Како свако дугме можемо повезати са неким догађајем, то важи и за ове компоненте. Команде (и поткоманде) у линији менија имају смисла ако су повезане са догађајима. Везивање ових елемената за догађаје врши се на исти начин као и за остале компоненте па се тиме нећемо овде бавити.

За додавање линије алата (то је линија са иконама чијим избором се реализује некаква акција), користи се наредба облика:

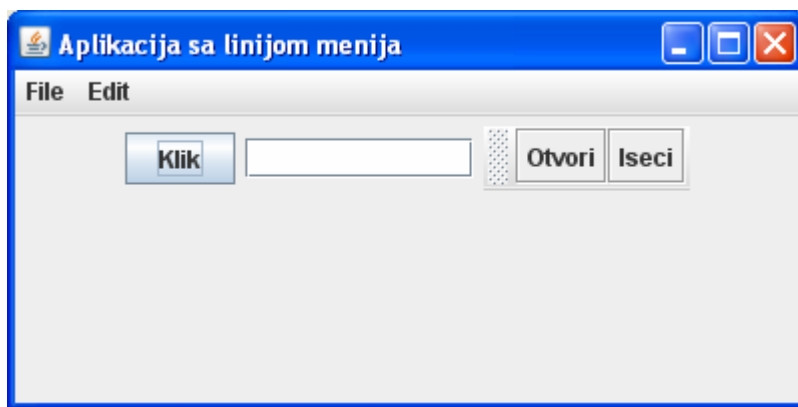
```
JToolBar linijaAlata = new JToolBar();
```

На линију алата постављају се дугмад. За те сврхе могу се искористити команде попут следећих:

```
JButton adugme1= new JButton("Otvori");  
linijaAlata.add(adugme1);  
JButton adugme2= new JButton("Iseci");  
linijaAlata.add(adugme2);
```

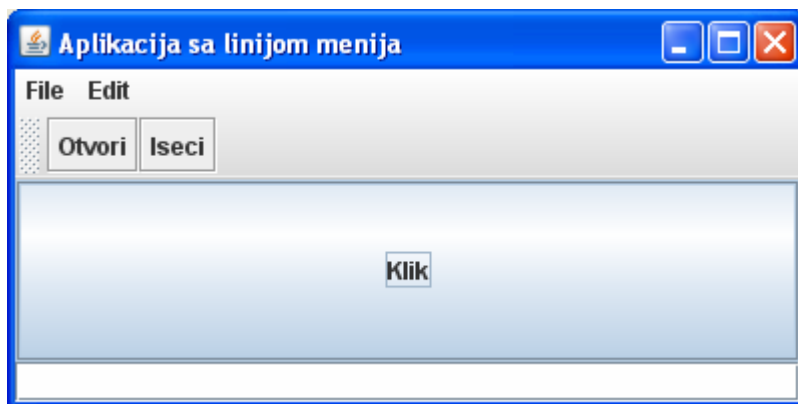
Након тога се линија алата убацује у панел као и остале компоненте, командом:
pan.add(linijaAlata);

Запажамо да су иконе на линији алата дугмад и везују се за догађаје као и остала дугмад. Ова дугмад се могу посебно обрадити, тако да садрже сличице које најбоље представљају акције везане за иконе. Покретањем измењеног програма, добија се прозор као на слици 15.




Слика 15.

Запажамо да је линија алата додата иза осталих компоненте (јер се користи FlowLayout-распоређивач). Повлачењем она се може поставити било где на екрану, али уобичајено је да линија алата стоји одмах испод линије менија. Да би се линија алата поставила испод линије менија, може се искористити распоређивач BorderLayout. Коришћењем овог распоређивача, добија се прозор као на слици 16.



Слика 16.

У свим наведеним примерима као икона апликације појављује се икона Java-е . То је подразумевана икона за Java-апликацију ако корисник не изабере неку другу. Као икона може се употребити сличица GIF или JPEG формаата (могу се користити и други формати, али ови су најраспрострањенији). Претпоставимо да у истом фолдеру (где се налази програм) имамо сличицу коју желимо да употребимо као икону апликације (нпр. ikona.gif). Коришћењем метода getImage() из класе Toolkit, постављамо своју икону као икону апликације. То се може урадити помоћу следећег скупа наредби:

```
Toolkit alat = Toolkit.getDefaultToolkit();  
Image img = alat.getImage("ikona.gif");  
prozor.setIconImage(img);
```

Приликом покретања програма добија се прозор, који се од прозора са слике 16 разликује само по сличици у горњем левом углу.

У наведеним примерима смо приказали само неке могућности које пружа програмски језик Java у креирању ГКИ-апликација. У приложеним примерима употребљене су само неке компоненте из великог скупа компоненти које постоје у Swing-у. Свака од тих компоненти има неке специфичности и потребно је знатно више простора да би се показало шта нам све омогућавају. У наведеним примерима претпостављено је познавање основних конструкција програмског језика Java. .