



Funded by the
Erasmus+ Programme
of the European Union



This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.



KI105 - JAVA 3: PROGRAMIRANJE KORISNIČKOG INTERFEJSA

Grafika u Javi

Lekcija 02

PRIRUČNIK ZA STUDENTE

KI105 - JAVA 3: PROGRAMIRANJE KORISNIČKOG INTERFEJSA

Lekcija 02

GRAFIKA U JAVI

- ✓ Grafika u Javi
- ✓ Poglavlje 1: Klasa Graphics
- ✓ Poglavlje 2: Crtanje teksta, linija, pravougaonika i elipsi
- ✓ Poglavlje 3: Crtanje lukova
- ✓ Poglavlje 4: Crtanje poligona i polilinja
- ✓ Poglavlje 5: Centriranje teksta upotrebom klase FontMetrics
- ✓ Poglavlje 6: Prikazivanje slika
- ✓ Poglavlje 7: Domaći zadaci
- ✓ Zaključak

Copyright © 2017 – UNIVERZITET METROPOLITAN, Beograd. Sva prava zadržana. Bez prethodne pismene dozvole od strane Univerziteta METROPOLITAN zabranjena je reprodukcija, transfer, distribucija ili memorisanje nekog dela ili čitavih sadržaja ovog dokumenta., kopiranjem, snimanjem, elektronskim putem, skeniranjem ili na bilo koji drugi način.

Copyright © 2017 BELGRADE METROPOLITAN UNIVERSITY. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, without the prior written permission of Belgrade Metropolitan University.

▼ Uvod

UVOD

Cilj lekcije

Ova lekcija ima za cilj da naučite da:

- crtate grafičke elemente uz pomoć metoda klase **Graphics**
- crtate i bojite grafičke elemente u okviru GUI pomoću metoda **paintComponent**
- koristite panel kao platno za crtanje grafike
- crtate stringove, linije, pravougaonike, elipse, lukove i poligone
- dobijete svojstva fonta upotrebom **FontMetrics** i prikaz teksta u panelu
- prikažete sliku na nekoj GUI komponenti
- razvijete višestruko upotrebljive GUI komponente **FigurePanel**, **MessagePanel**, **StillClock**, i **ImageViewer**

▼ Poglavlje 1

Klasa Graphics

UML MODEL KLASSE GRAPHICS

Svaka GUI komponenta ima određeni grafički kontekst, koji je objekt klase Graphics. Klasa Graphics sadrži metode za crtanje različitih oblika.

Klasa **Graphics** obezbeđuje metode za ispisivanje tekstova, linija, pravougaonika, elipsi, lukova, poligona i polilinja, kao što je prikazano na slici 1. Ovi metodi omogućuju da nacrtate grafičke elemente koje koriste GUI komponente. Svaka komponenta ima svoj koordinatni sistem sa koordinatnim početkom u gornjem levom uglu. Koordinata x raste udesno, a koordinata y - nadole (slika 1). Klasa Graphics je apstraktna klasa koja obezbeđuje grafički interfejsa koji je nezavistan od konkretnog uređaja/sistema a koji omogućava prikaz različitih geometrijskih oblika i slika na ekranu na različitim platformama.

Java.awt.Graphics	
<pre>+setColor(color: Color): void +setFont(font: Font): void +drawString(s: String, x: int, y: int): void +drawLine(x1: int, y1: int, x2: int, y2: int): void +drawRect(x: int, y: int, w: int, h: int): void +fillRect(x: int, y: int, w: int, h: int): void +drawRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void +fillRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void +draw3DRect(x: int, y: int, w: int, h: int, raised: boolean): void +fill3DRect(x: int, y: int, w: int, h: int, raised: boolean): void +drawOval(x: int, y: int, w: int, h: int): void +fillOval(x: int, y: int, w: int, h: int): void +drawArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void +fillArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void +drawPolygon(xPoints: int[], yPoints: int[], nPoints: int): void +fillPolygon(xPoints: int[], yPoints: int[], nPoints: int): void +drawPolygon(g: Polygon): void +fillPolygon(g: Polygon): void +drawPolyline(xPoints: int[], yPoints: int[], nPoints: int): void</pre>	<p>Postavljanje nove boje za naredne crteže Postavljanje novog fonta za naredne crteže Crta string iz položaja (x, y) Crta liniju od (x1, y1) do (x2, y2) Crta pravougaonik sa gornjim levim uglom (x,y) i širinom w i visinom h Crta obojeni pravougaonik sa gornjim levim uglom (x,y) i širinom w i visinom h Crta pravougaonik sa zaobljenim uglovima visine aw i visine ah Crta obojeni pravougaonik sa zaobljenim uglovima visine aw i visine ah Crta 3D pravougaonik iznad ili ispod površine Crta obojeni 3D pravougaonik iznad ili ispod površine Crta elipsu definisanu spoljnim pravougaonikom sa parametrima x,y,w,h Crta obojenu elipsu definisanu spoljnim pravougaonikom sa parametrima x,y,w,h Crta luk kao deo elipse definisane sa parametrima x,y,w i h Crta obojeni luk kao deo elipse definisane sa parametrima x,y,w i h Crta zatvoreni poligon definisan nizom x i y koordinata tj. tačkama (x(i), y(i)) Crta obojeni zatvoreni poligon definisan nizom x i y koordinata tj. tačkama (x(i), y(i)) Crta zatvoren poligon definisan objektom Polygon Crta obojen poligon definisan objektom Polygon Crta poliliniu definisanu nizom x i y koordinata, tj. tačkama (x(i), y(i))</p>

Slika 1.1 Koordinatni sistema Java, u kome su x i y koordinate date u pikselima

Uvek kada se neka GUI komponenta prikaže, JVM automatski kreira **Graphics** objekat. On ima zadatak da za tu komponentu da za konkretnu platformu sa svojim metodom **paintComponent** prikazuje crteže, tj. geometrijske oblike koji čine korisnički interfejs (GUI). Potpis metoda **paintComponent** je:

```
protected void paintComponent(Graphics g)
```

Ovaj metod je definisan u klasi **Component**, a poziva se uvek kada se želi da prikaže (ili osveži prikaz)

PRIMER 1 - TESTPAINTCOMPONENT

*Da bi se nacrtala neka komponenta, potrebno je da definišete klasu koja proširuje klasu **JPanel** i prekriva svoj metod **paintComponent** radi specifikacije onoga šta treba da nacrtati*

Da bi se nacrtala neka komponenta, potrebno je da definišete klasu koja proširuje klasu **JPanel** i prekriva svoj metod **paintComponent** radi specifikacije onoga šta treba da nacrtati. Listing klase **TestPaintComponent** predstavlja program za crtanje linije i string na panelu, kao što je prikazano na slici 2 .

```
import javax.swing.*;
import java.awt.Graphics;

public class TestPaintComponent extends JFrame {

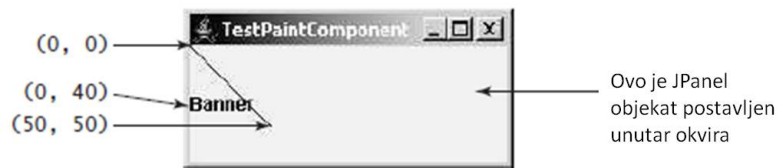
    public TestPaintComponent() {
        add(new NewPanel());
    }

    public static void main(String[] args) {
        TestPaintComponent frame = new TestPaintComponent();
        frame.setTitle("TestPaintComponent");
        frame.setSize(200, 100);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

class NewPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawLine(0, 0, 50, 50);
        g.drawString("Banner", 0, 40);
    }
}
```

Kada se u programu pozove metod **paintComponent**, automatski počinje crtanje i bojenje grafičkih elemenata kada se GUI komponenta prikaže na ekranu prvi put, ili kada se osveži njen prikaz (zbog nekih izmena u programu ili u interakciji). Izrazom `super.paintComponent(g)` (linija 22) poziva se metod **paintComponent** metod definisan u super klasi . To je neophodno da bi se očistio prostor prikazivanja pre nego što se prikaže novi crtež.



Slika 1.2 Linija i tekst su nacrtani na panelu

Onda se u liniji 23 sa pozivom metoda `drawLine` crta linija između tačaka (0, 0) do (50, 50). Linija 24 poziva metod **`drawString`** koji crta tekst (string) „Banner“ na lokaciji (0,40). Sve unete dimenzije su u pikselima.

JVM (Java virtualna mašina) crta komponente. Ne treba da korisnik direktno poziva **`paintComponent`**. Zato je metod **`paintComponent`** definisan sa ograničenim pristupom (protected).

ZADATAK 1

Napravite samostalno korekcije u Primeru 1.

Pokušajte samostalno, u primeru Primer1, da postavite dve ukrštene linije (da precrtate) preko stringa Banner.

▼ Poglavlje 2

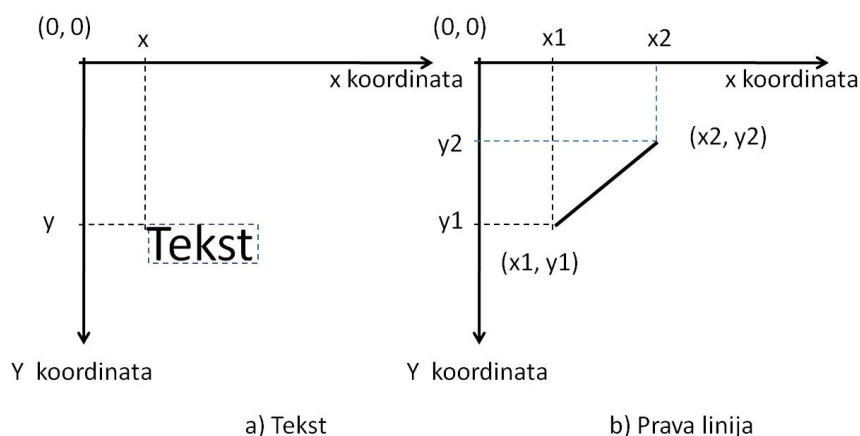
Crtanje teksta, linija, pravougaonika i elipsi

CRTANJE LINIJE I TEKSTA

Možete nacrtati tekstove, linije, pravougaonike i elipse u nekom grafičkom kontekstu

Paneli su nevidljivi i koriste se kao mali kontejneri u kom se grupišu grafičke komponente da bi se ostvario željeni njihov raspored. Kreiranjem objekta klase **JPanel**vi kreirate „platno“ za crtanje. Ako kao „platno“ (engl. canvas) koristite objekt klase **JPanel**, onda možete da pozadinu jednostavno obojite pozivom metoda **setBackground(Color color)**. Ako za „platno“ koristite objekat klase **JComponent**, što je tehnički moguće, onda to ne možete tako jednostavno uraditi, jer bi morali da napravite mali program da bi to „platno“ obojili.

Pozivanjem metoda **drawString(String s, int x, int y)** možete nacrtati tekst koji počinje u tački (x, y), kao što je pokazano na slici 1 a. Pozivanjem metoda **drawLine(int x1, int y1, int x2, int y2)** crtate pravu liniju iz tačke (x1, y1) u tačku (x2, y2), kao što je pokazano na slici 1 b.



Slika 2.1.1 Ispisivanje teksta i crtanje prave linije metodama `drawString` i `drawLine`

ZADACI ZA SAMOSTALAN RAD

1. Pretpostavimo da želite da nacrtate novu poruku uspod postojeće poruke. Da li će te povećati x-koordinatu, y-koordinatu ili obe?
2. Kako se kreira objekat klase **Graphics**?
3. Kako se poziva metod **paintComponent**?

4. Zašto metod **paintComponent** ima vidljivost `protected`? Šta bi se desilo ako bi to promenili u `public` ili u `private` u podklasi? Zašto je pozvan metod **super.paintComponent(g)** u liniji 22 LISTINGA?

PRIMER 2

Mogućnost crtanja linija u odnosu na veličinu ekrana

Podeliti ekran na 9 delova koristeći metodu `drawLine` `Graphics2D` klase tako da kada se prozor širi i smanjuje automatski se i delovi povećavaju i smanjuju.

Na slici 1 (ispod) se vidi očekivani rezultat



Slika 2.1.2 - Zadatak 1

Treba napraviti novu klasu koja nasleđuje klasu `JPanel`. Klasa `JPanel` ima metod **`paintComponent(Graphics)`** koju prevazilazimo u našoj implementaciji klase **`PanelKlasa`**. Parametar metode kastujemo u `Graphics2D` kako bismo mogli da koristimo metod `drawLine` za crtanje linije. Za override metode možemo da koristimo **`Alt+Insert`** u NetBeansu i da biramo `Override method...` a zatim da nađemo `paintComponent`.

Prvo računamo razmak između linija, to je ovde jedna trećina visine ili širine prozora. Metoda `drawLine` prima koordinate prve tačke i koordinate druge tačke između kojih se povlači linija. Ako je širina i visina prozora 30, prva horizontalna linija je određena sledećim tačkama: A (0,10) i B (30,10). Prva vertikalna linija tačkama: A(10,0) i B(10,30). Druga horizontalna linija tačkama: A(0,20) i B(30,20) itd.

Kako želimo da se mreža prilagođava promeni veličine prozora koristimo metode `getHeight` i `getWidth` kako bi smo dobili visinu i širinu prozora.

Klasa `PanelKlasa`:

```
package zadatak1;

import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JPanel;
```

```
public class PanelKlasa extends JPanel {
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        int y = this.getHeight() / 3;
        int x = this.getWidth() / 3;
        g2.drawLine(0, y, this.getWidth(), y);
        g2.drawLine(0, y * 2, this.getWidth(), y * 2);
        g2.drawLine(x, 0, x, this.getHeight());
        g2.drawLine(x * 2, 0, x * 2, this.getHeight());
    }
}
```

PRIMER 2 - MAIN

Implementacija klase Main

Testiramo implementiranu klasu PanelKlasa

Postavljamo kreiranu rešetku pomoću **setContentPane** metode.

Trenutno rešenje ima fiksiran broj horizontalnih i vertikalnih linija u rešetki na 3. Pokušati izmeniti klasu PanelKlasa tako da sada ima konstruktor sa dva argumenta, rows i cols koji određuju broj linija.

Klasa Main:

```
package zadatak1;
import javax.swing.JFrame;
public class Main extends JFrame {

    /**
     * @param args the command line arguments
     */
    public Main() {
        setSize(500, 500);
        setTitle("Resetka");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new PanelKlasa());
        setLocationRelativeTo(null);
        setVisible(true);
    }

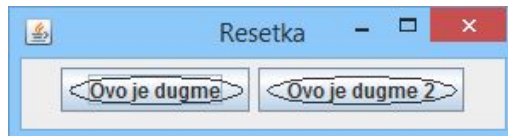
    public static void main(String[] args) {
        new Main();
    }
}
```

PRIMER 3

Mogućnost crtanja preko postojećih komponenti

Napraviti grafički interfejs koji ima na sebi dva dugmeta smeštena u FlowLayout. Dugmići trebaju da imaju nacrtan krug oko teksta.

Napomena: Za crtanje kruga koristiti override metode `paintComponent`.



Slika 2.1.3 - Zadatak 2

U Javi imamo mogućnost da menjamo ponašanje kao i rad postojećih komponenti koristeći nasleđivanje i override metode pa ćemo to ovde iskoristiti da napravimo našu vrstu dugmeta - Klasa **OvalButton**. Naša klasa nasleđuje klasu `JButton`.

Svaka GUI komponenta u okviru Java Swing biblioteke koja nasleđuje `JComponent` može da override-uje `paintComponent` metodu pri čemu je moguće da sami definišemo način icrtavanja komponente.

U ovom slučaju smo za klasu `OvalButton` definisali da se pored standardnog icrtavanja dugmeta (poziv `super.paintComponent(g);`) iscrta i dodatni krug preko dugmeta pozivom metode `drawOval`.

Metod `drawOval` crta krug ili elipsu koji staju u zadati pravougaonik. Prva dva argumenta su koordinate gornjeg levog temena pravougaonika, a druga dva su širina i visina.

Klasa `OvalButton`:

```
package zadatak2;

import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JButton;

public class OvalButton extends JButton {

    public OvalButton() {
    }

    public OvalButton(String text) {
        super(text);
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
```

```
        g2.drawOval(5, 5, this.getWidth()-10, this.getHeight()-10);  
    }  
}
```

Klasa Main:

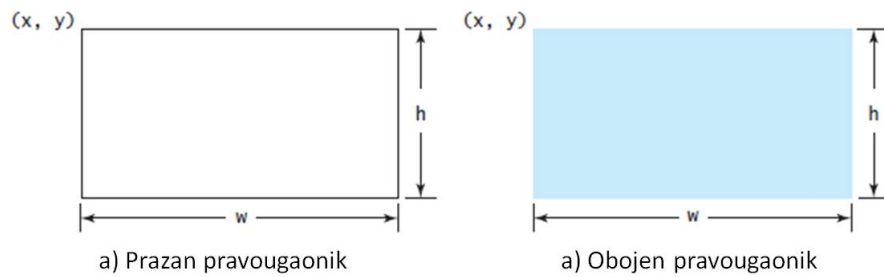
```
import java.awt.FlowLayout;  
import javax.swing.JFrame;  
import javax.swing.JPanel;  
  
public class Main extends JFrame {  
  
    OvalButton btn1 = new OvalButton("Ovo je dugme");  
    OvalButton btn2 = new OvalButton("Ovo je dugme 2");  
  
    public Main() {  
        setSize(300, 80);  
        setTitle("Resetka");  
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
        JPanel panel = new JPanel();  
        setContentPane(panel);  
        setLayout(new FlowLayout());  
        setLocationRelativeTo(null);  
        setVisible(true);  
        panel.add(btn1);  
        panel.add(btn2);  
    }  
  
    public static void main(String[] args) {  
        new Main();  
    }  
}
```

CRTANJE PRAVOUGAONIKA

Java obezbeđuje šest metoda za crtanje pravougaonika: drawRect(), fillRect(), drawRoundRect(), fillRoundRect(), draw3DRect(), fill3DRect(). Metod drawOval() crta elipsu.

Java obezbeđuje šest metoda za crtanje pravougaonika. Možete nacrtati prazan (neobojeni) pravougaonik sa normalnim ili sa zaobljenim uglovima, i isto to, ali umesto da pravougaonik bude prazan (bez boje) - da bude obojen. To isto možete da uradite i u prostoru (dodavanjem treće koordinate z), tj. crtanjem praznog i obojenog paralelopipeda sa normalnim ili zaobljenim uglovima.

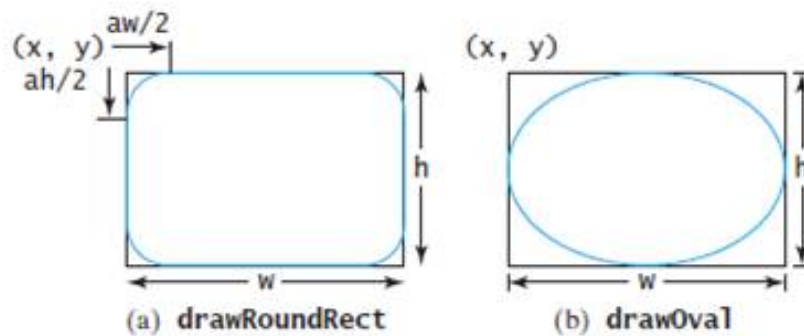
Pozivanjem metoda **drawRect(int x, int y, int w, int h)** crta prazan pravougaonik (slika 2 a), a metod **fillRect(int x, int y, int w, int h)** crta obojen pravougaonik (slika 2 b). Parametri x i y predstavljaju gornji levi ugao pravougaonika, w i h su širina i visina pravougaonika



Slika 2.1.4 Crtanje pravougaonika sa normalnim i zaobljenim uglovima metodima `drawRect` i `fillRect`

Pozivanjem metoda **`drawRoundRect(int x, int y, int w, int h, int aw, int ah)`** crta pravougaonik sa zaobljenim uglovima – prazan, bez boje (slika 3 a), a pozivanjem metoda **`fillRoundRect(int x, int y, int w, int h, int aw, int ah)`** crta se obojeni pravougaonik, ali sa zaobljenim uglovima.

Pozivanjem metoda **`drawOval(int x, int y, int w, int h)`** ili **`fillOval(int x, int y, int w, int h)`** crta se elipsa unutar spoljnog pravougaonika definisanog dimenzijama w (za širinu) i h (za visinu). Parametri x i y označavaju gornji levi ugao tog spoljnog pravougaonika elipse. (slika 3 b).



Slika 2.1.5 Crtanje pravougaonika sa zaobljenim uglovima metodom `drawRoundRect` i elipse sa metodom `drawOval`

CRTANJE PARALELOPIPEDA

Metod `draw3DRect()` crta prostorni pravougaonik, tj. paralelopiped, a metod `fill3DRect()` ga boji.

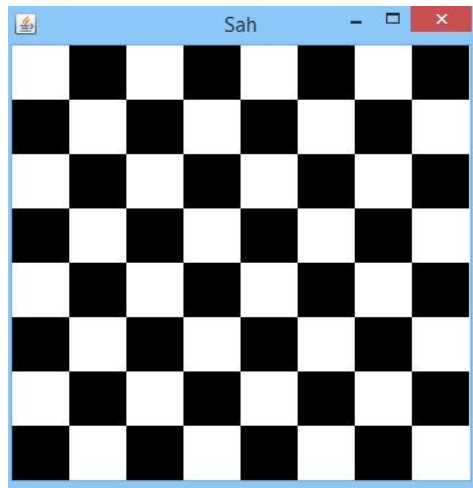
Sa metodom **`draw3DRect(int x, int y, int w, int h, boolean raised)`** crta se prostorni, 3D pravougaonik (paralelopiped), tj. kvadar, i to bez boja, a sa metodom **`fill3DRect(int x, int y, int w, int h, boolean raised)`** – obojeni paralelopiped. Logička vrednost `raised` označava da li je paralelopiped podignut ispod površine ili je „potopljen“ ispod površine koja čini osnovnu ravan prostornog prikaza.

PRIMER 4

Upotreba petlji pri crtanju

Napraviti grafički interfejs koji će na sebi da ima iscrtanu šahovsku tablu.

Napomena: Koristiti petlju unutar paintComponent metode panela kako bi izcrtali tabelu 8x8 koja liči na šah.



Slika 2.1.6 - Zadatak 3

I u ovom zadatku će pokretačka klasa nasleđivati JFrame klasu i kreiraćemo odvojenu klasu u okviru koje ćemo definisati **JPanel** na kome se iscrtava šahovska tabla.

Klasa Main:

```
package zadatak3;

import javax.swing.JFrame;

public class Main extends JFrame {

    public Main() {
        setSize(335, 360);
        setLocationRelativeTo(null);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        add(new ChessPanel());
        setVisible(true);
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Izmeniti klasu ChessPanel tako da se iscrtavanje započinje sa crnim kvadratom.

PRIMER 4 - KLASA CHESSPANEL

Implementacija klase ChessPanel

Kako bi smo iscrtali šahovsku tablu potrebno je da se prilikom svakog iscrtavanja vrši promena boje. Za te potrebe smo napravili **toggleColor** metodu koja vrši smenjivanje menjanje crne i bele boje. Za rad sa bojama koristimo klasu Color. Veličina panela se određuje objektom klase **Dimension** koji sadrži širinu i visinu.

U metodi **paintComponent** vršimo iscrtavanje pojedinačnih kvadrata koji predstavljaju polja na šahovskoj tabli. Dakle, koristimo dve petlje, jedna predstavlja pomera kvadrata po x koordinati, a druga po y koordinati. U konstruktoru smo takođe izračunali veličinu stranice jednog polja koji ćemo koristiti kao parametar prilikom iscrtavanja kvadrata kao i prilikom računanja x i y koordinata u svakoj iteraciji petlje.

Primer: Ako su širina i visina prozora 80, onda je stranica kvadrata 10. U prvom redu su koordinate x i y: (0,0) (10,0) (20,0) itd. U drugom redu: (0,10) (10,10) (20,10) itd.

Na nivou svakog reda crna i bela boja se uzajamno smenjuju, ali je bitno zapaziti da se prilikom prelaska u novi red vrši promena boje prvog polja (ako je u prvom redu boja bela u drugom će biti crna), pa je zbog toga potrebno da pre prelaska u novi red ponovo pozovemo kolonu **toggleColor**.

Crtanje obojenog kvadrata se vrši pomoću fillRect. Prvo je potrebno postaviti boju pomoću setColor. Pri crtanju se navode koordinate x i y gornjeg levog temena kao i visina i širina pravougaonika.

```
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JPanel;

public class ChessPanel extends JPanel {

    private Color tempColor;
    private int spotSize;

    public ChessPanel() {
        setSize(new Dimension(320, 320));
        spotSize = getWidth() / 8;
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        tempColor = Color.BLACK;
        int x = 0;
        int y = 0;
        Graphics2D g2d = (Graphics2D) g;
        for (int i = 0; i < 8; i++) {
```

```
        for (int j = 0; j < 8; j++) {
            toggleColor();
            x = j * spotSize;
            y = i * spotSize;
            g2d.setColor(tempColor);
            g2d.fillRect(x, y, spotSize, spotSize);
        }
        toggleColor();
    }
}

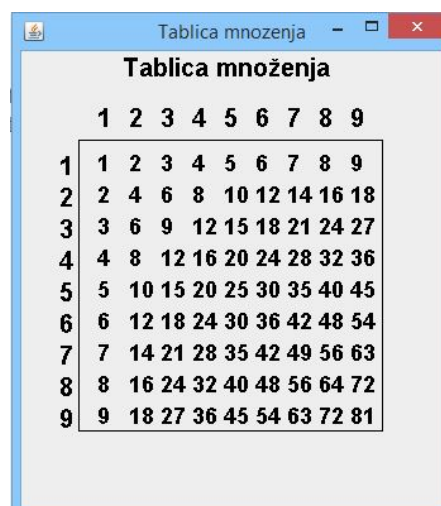
public void toggleColor(){
    if(tempColor == Color.WHITE){
        tempColor = Color.BLACK;
    } else {
        tempColor = Color.WHITE;
    }
}
}
```

PRIMER 5

Ispisivanje teksta na Graphics2D komponenti

Napraviti grafički interfejs koji će na sebi da ima iscrtanu tablicu množenja.

Napomena: Koristiti petlju unutar **paintComponent** metode panela kako bi izcrtali tablicu množenja kao na sledećoj slici:



The screenshot shows a Java Swing window titled "Tablica mnozenja". Inside the window, there is a 9x9 multiplication table. The table has a header row and a header column, both labeled with numbers 1 through 9. The body of the table contains the products of these numbers. The table is styled with a light gray background and black text. The window has a standard Mac OS X-style title bar with a red close button, a yellow maximize button, and a green window control button.

	1	2	3	4	5	6	7	8	9
1	1	2	3	4	5	6	7	8	9
2	2	4	6	8	10	12	14	16	18
3	3	6	9	12	15	18	21	24	27
4	4	8	12	16	20	24	28	32	36
5	5	10	15	20	25	30	35	40	45
6	6	12	18	24	30	36	42	48	54
7	7	14	21	28	35	42	49	56	63
8	8	16	24	32	40	48	56	64	72
9	9	18	27	36	45	54	63	72	81

Slika 2.1.7 - Zadatak 4

U Main klasi standardno definišemo parametre za prikaz JFrame prozora, dok će se icrtavanje same tablice realizovati u posebnoj klasi **PanelTablica**.

Klasa Main:


```
package zadatak4;

import javax.swing.JFrame;

public class Main extends JFrame {

    public Main() {
        setSize(350, 400);
        setTitle("Tablica mnozenja");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new PanelTablica());
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Izmeniti klasu PanelTablica tako da se brojevi na dijagonali tablice ispisuju crvenom bojom.

PRIMER 5 - KLASA PANELTABLICA

Implementacija klase PanelTablica

Kako bi smo iscrtali tablicu množenja koja se traži u zadatku prvo ćemo koristeći dve petlje ispisati brojeve od 1 do 9 po horizontali i vertikali. Po horizontali je potrebno da se u svakoj iteraciji poveća vrednost x koordinate kako bi smo postigli razmak između cifara, dok se iz istog razloga prilikom crtanja cifara po vertikali povećava y koordinata.

Konačno da bi smo izvršili množenje koristimo dve ugneždene petlje i vršimo množenje brojača "i" i "j" kako bismo dobili odgovarajuće vrednosti koje će se iscrtati u tablici množenja.

Kada se završi jedan ciklus iteracija petlje sa j brojačem potrebno je da se y koordinata poveća za vrednost jednog razmaka, a da se x koordinata restartuje na početnu vrednost u redu (u ovom slučaju 60) kako bi se novi red iscrtavao počevši od početka reda.

Za ispisivanje teksta koristimo metod drawString. Metod zahteva da mu se prosledi string koji se ispisuje kao i pozicija na kojoj se ispisuje. Pre ispisivanja stringa podešavamo boju teksta sa setColor i font kojim se tekst ispisuje sa setFont. Font je predstavljen klasom Font, konstruktor zahteva naziv fonta, stil fonta i veličinu fonta.

```
package zadatak4;

import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
```

```
import javax.swing.JPanel;

public class PanelTablica extends JPanel {

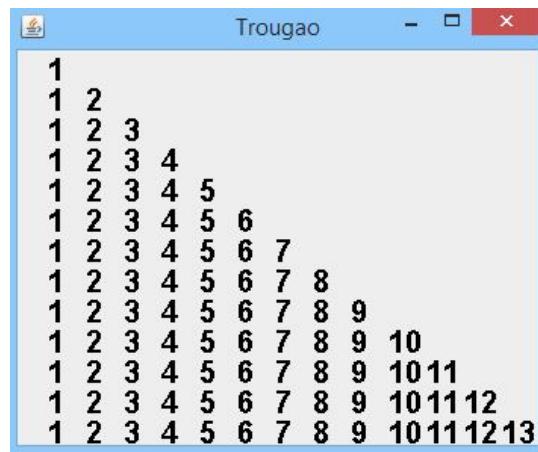
    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        g2.setColor(Color.black);
        g2.setFont(new Font("Arial", 1, 20));
        g2.drawString("Tablica množenja", 80, 20);
        int x = 60;
        for (int i = 1; i < 10; i++) {
            g2.drawString(String.valueOf(i), x, 60);
            x += 25;
        }
        int y = 98;
        for (int i = 1; i < 10; i++) {
            g2.drawString(String.valueOf(i), 30, y);
            y += 25;
        }
        g2.drawRect(45, 70, 240, 230);
        g2.setFont(new Font("Arial", 1, 18));
        int xp = 60;
        int yp = 95;
        for (int i = 1; i <= 9; i++) {
            for (int j = 1; j <= 9; j++) {
                g2.drawString(String.valueOf(i * j), xp, yp);
                xp += 25;
            }
            yp += 25;
            xp = 60;
        }
    }
}
```

PRIMER 6

Iscrtavanje trougla sastavljenog od brojeva

Napraviti grafički interfejs koji će na sebi da ima iscrtan trougao pomoću brojeva kao na sledećoj slici.

Napomena: Veličina trougla, tj. broj redova brojeva bi trebalo da se povećava ukoliko se ekran poveća.



Slika 2.1.8 - Zadatak 5

Broj redova treba da zavisi od visine prozora:

```
int rows = this.getHeight() / 20;
```

Prvi red ima jedan broj, drugi red dva, treći tri, itd. Tako dobijamo trougao. U okviru jednog reda povećavamo x koordinatu broja svaki put, a kada prelazimo u novi red onda povećamo i y, a x resetujemo na početnu vrednost.

Klasa Main:

```
package zadatak5;

import javax.swing.JFrame;

public class Main extends JFrame {

    public Main() {
        setSize(360, 300);
        setTitle("Trougao");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new PanelTrougao());
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Klasa PanelTrougao:

```
package zadatak5;
```

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JPanel;

public class PanelTrogao extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        g2.setColor(Color.black);
        g2.setFont(new Font("Arial", 1, 20));
        int rows = this.getHeight() / 20;
        int x = 20;
        int y = 20;
        for (int i = 1; i <= rows; i++) {
            for (int j = 1; j <= i; j++) {
                g2.drawString(String.valueOf(j), x, y);
                x += 25;
            }
            y += 20;
            x = 20;
        }
    }
}
```

KLASA GRAPHICS (VIDEO)

Metod `paintComponent()` klase `Graphics` kreira sve grafičke komponente koju koristimo pri kreiranju nekog crteža.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADACI 2 - 7

Crtanje teksta, linija, pravougaonika i elipsi - samostalna vežba

Zadatak 2. Opišite metode za crtanje tekstova (stringova), linija, i metode u acrtanje(bojenje pravougaonika, pravougaonika sa zaobljenim uglovima, 3D pravougaonika (paralelopipeda) i elipse,

Zadatak 3. Nacrtajte debelu liniju od tačke (10, 10) do tačke (70, 30). Nacrtajte i nekoliko linija jedne do druge da bi ostvarili efekat debele linije.

Zadatak 4. Nacrtajte i obojite pravougaonik širine 100 i visine 50, sa gornjim levim uglom u tački (10, 10).

Zadatak 5. Nacrtajte i obojite pravougaonik sa zaobljenim uglovima širine 100 i visine 200, sa horizontalnim prečnikom zaobljenja ugla 40, sa vertikalnim prečnikom zaobljenja ugla 20, i sa gornjim levim uglom u tački (10, 10).

Zadatak 6. Nacrtajte i obojite krug sa poluprečnikom 30

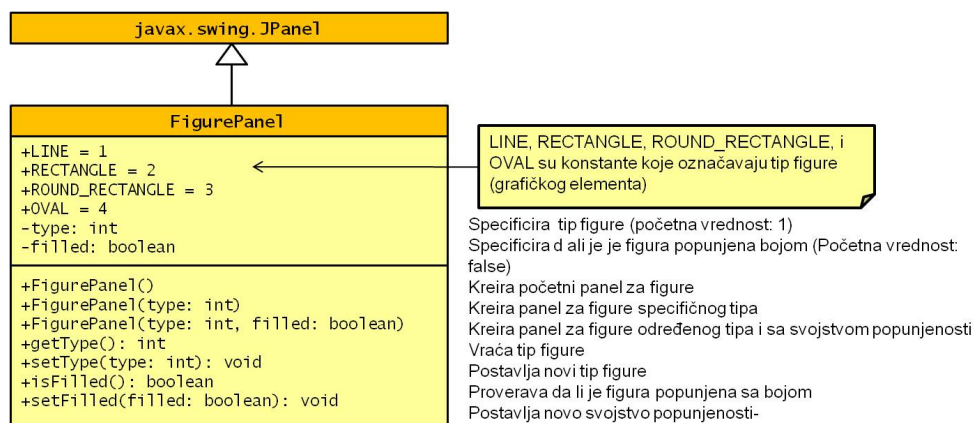
Za **datak 7.** Nacrtajte i obojite elipsu širine 50 i visine 100.

▼ 2.1 Klasa FigurePanel

UML MODEL KLASE FIGUREPANEL

Klasa FigurePanel omogućava prikaz linije, pravougaonika, pravougaonika sa zaobljenim uglovima i elipse.

Klasa **FigurePanel** (slika 1) omogućava korisniku da definiše jedan od osnovnih grafičkih elemenata (linija, pravougaonik, pravougaonik sa zaobljenim uglovima i elipsa) i da odredi da li taj element treba da bude obojen ili ne. Tako definisan element klasa **FigurePanel** onda prikazuje na ekranu.



Slika 2.2.1 Klasa FigurePanel prikazuje osnovne geometrijske elemente na panelu

PRIMER 7 - KORIŠĆENJE KLASE TESTFIGUREPANEL

Klasa TestFigurePanel sadrži main() metod ove studije slučaja. Klasa TestFigurePanel, u svom konstruktoru, kreira objekat FigurePanel za svaku od nacrtanih figura.

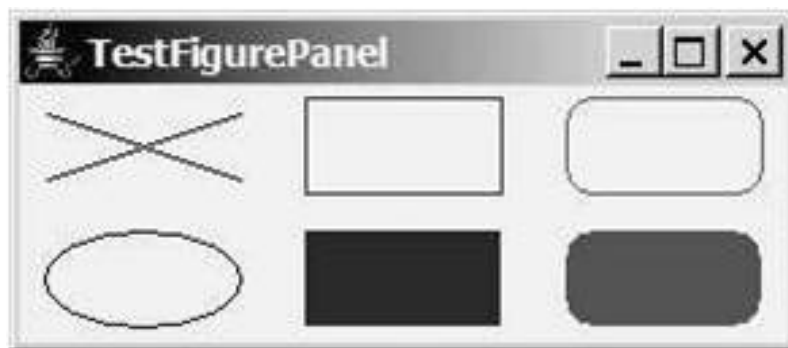
Dole je prikazan listing klase **TestFigurePanel** da bi prikazao šest figura na panelu koje su prikazane na slici 2 .

```
import java.awt.*;
import javax.swing.*;

public class TestFigurePanel extends JFrame {

    public TestFigurePanel() {
        setLayout(new GridLayout(2, 3, 5, 5));
        add(new FigurePanel(FigurePanel.LINE));
        add(new FigurePanel(FigurePanel.RECTANGLE));
        add(new FigurePanel(FigurePanel.ROUND_RECTANGLE));
        add(new FigurePanel(FigurePanel.OVAL));
        add(new FigurePanel(FigurePanel.RECTANGLE, true));
        add(new FigurePanel(FigurePanel.ROUND_RECTANGLE, true));
    }

    public static void main(String[] args) {
        TestFigurePanel frame = new TestFigurePanel();
        frame.setSize(400, 200);
        frame.setTitle("TestFigurePanel");
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```



Slika 2.2.2 Šest figura na panelu

LISTING KLASJE FIGUREPANEL

Klasa FigurePanel crta liniju, pravougaonik, pravougaonik sa zaobljenim temenima i elipsu.

Četiri konstante - LINE, RECTANGLE, ROUND_RECTANGLE, i OVAL definisane su u linijama 6-9. Četiri tipa figura se crtaju u skladu sa svojim tipom (linija 37), a metodom **setColor** postavlja se nova boja za crtanje figura u linijama 39, 44, 53, i 62.

Metod **repaint** (linije 75, 86) je definisan u klasi **Component**. Pozivanjem repaint metoda dovodi do pozivanja metoda **paintComponent**. Metod repaint se poziva da bi se osvežio prikaz na ekranu. Po pravilu, to se dešava kada imate nove stvari za prikazivanje.

Treba obratiti pažnju da se metod **paintComponent** nikad ne treba direktno da poziva. Poziva ga ili JVM pri promeni prostora prikazivanja ili od strane metoda repaint. Metod **paintComponent** se može prekriti da bi se sistemu saopštio kako da oboji prostor prikazivanja, ali se nikad ne sme prekriti metod repaint. Metod **getPreferredSize()** (linije 95-97), definisan u klasi **Component**, je prekriven u klasi **FigurePanel** da bi definisao željenu veličinu za menadžera rasporeda a radi postavljanje rasporeda objekta FigurePanel. Ovo svojstvo se može, ali ne mora, da se uzme u obzir od strane menadžera rasporeda, zavisno od pravila. Dobra je praksa da se prekrije **getPreferredSize()** u podklasi **JPanel** a radi specificiranja

željene veličine, jer je početna širina i visina JPanel definisana sa 0. U tom slučaju, nećete ništa videti pošto je JPanel sa širinom i visinom 0 a nalazi se u kontejneru koju uređuje FlowLayout menadžer rasporeda

```
import java.awt.*;
import javax.swing.JPanel;

public class FigurePanel extends JPanel {
    // Deklaracija konstanti

    public static final int LINE = 1;
    public static final int RECTANGLE = 2;
    public static final int ROUND_RECTANGLE = 3;
    public static final int OVAL = 4;

    private int type = 1;
    private boolean filled = false;

    /**
     * Konstruktor početnog FigurePanel objekta
     */
    public FigurePanel() {
    }

    /**
     * Konstruktor objekta FigurePanel određenog tipa
     */
    public FigurePanel(int type) {
        this.type = type;
    }

    /**
     * Konstruktor objekta FigurePanel određenog tipa i popunjenosti
     */
    public FigurePanel(int type, boolean filled) {
        this.type = type;
        this.filled = filled;
    }

    @Override // Crta figuru na panelu
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
    }
}
```

```
// Dobijanje određene veličine figure na panelu
int width = getWidth();
int height = getHeight();

switch (type) {
    case LINE: // Prikaz dve ukrštene linije
        g.setColor(Color.BLACK);
        g.drawLine(10, 10, width - 10, height - 10);
        g.drawLine(width - 10, 10, 10, height - 10);
        break;
    case RECTANGLE: // Prikaz pravougaonika
        g.setColor(Color.BLUE);
        if (filled) {
            g.fillRect((int) (0.1 * width), (int) (0.1 * height),
                (int) (0.8 * width), (int) (0.8 * height));
        } else {
            g.drawRect((int) (0.1 * width), (int) (0.1 * height),
                (int) (0.8 * width), (int) (0.8 * height));
        }
        break;
    case ROUND_RECTANGLE: // Prikaz pravougaonika sa zaobljenim uglovima
        g.setColor(Color.RED);
        if (filled) {
            g.fillRoundRect((int) (0.1 * width), (int) (0.1 * height),
                (int) (0.8 * width), (int) (0.8 * height), 20, 20);
        } else {
            g.drawRoundRect((int) (0.1 * width), (int) (0.1 * height),
                (int) (0.8 * width), (int) (0.8 * height), 20, 20);
        }
        break;
    case OVAL: // Prikaz elipse
        g.setColor(Color.BLACK);
        if (filled) {
            g.fillOval((int) (0.1 * width), (int) (0.1 * height),
                (int) (0.8 * width), (int) (0.8 * height));
        } else {
            g.drawOval((int) (0.1 * width), (int) (0.1 * height),
                (int) (0.8 * width), (int) (0.8 * height));
        }
    }
}

/**
 * Postavljanje novog tipa figure
 */
public void setType(int type) {
    this.type = type;
    repaint();
}

/**
 * Vraća tip figure
 */
}
```



```

    */
    public int getType() {
        return type;
    }

    /**
     * Podešavanje novog svojstva popunjenosti
     */
    public void setFilled(boolean filled) {
        this.filled = filled;
        repaint();
    }

    /**
     * Provera da li je figura popunjena bojom
     */
    public boolean isFilled() {
        return filled;
    }

    @Override // Definisanje nove veličine
    public Dimension getPreferredSize() {
        return new Dimension(80, 80);
    }
}

```

ZADATAK 8

Korigujte Primer 7

- U razvojnom okruženju NetBeans IDE otvorite prethodni primer "Primer 7";
- Dodajte novi objekat - linije prave znak "+";
- Neka vodoravna linija bude crvena, a vertikalna plava.
- Nakon izmena pokrenite i testirajte program

✓ Poglavlje 3

Crtanje lukova

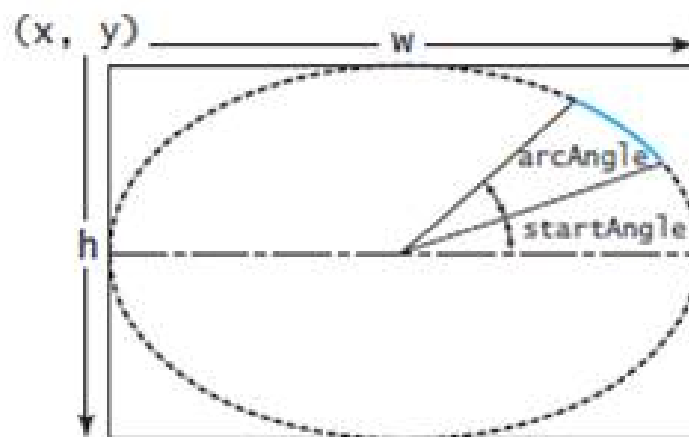
CRTANJE LUKA

Luk je deo elipse ograničene spoljnim pravougaonikom.

Metodi za crtanje ili i popunjavanje bojom luka imaju sledeći potpis::

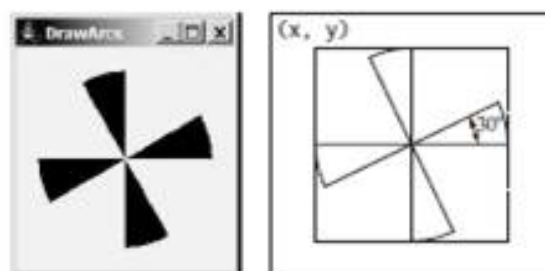
```
drawArc(int x, int y, int w, int h, int startAngle, int arcAngle)
fillArc(int x, int y, int w, int h, int startAngle, int arcAngle)
```

Parametri x , y , w i h su isti kao i kod definisanje elipse, tj. kod metoda **drawOval**. Parametar **startAngle** je početni ugao, a **arcAngle** je ugao obuhvata luka. Uglovi se mere u stepenima u skladu sa uobičajenim matematičkim konvencijama. Slika 1 prikazuje parametre za definisanje luka



Slika 3.1 Metod drawArc crta luka koristeći definisanu elipsu i dva ugla

Listing prikazuje primer crtanja lukova prikazanih na slici 2 .



Slika 3.2 ventilatora koji klasa DrawArcs treba da nacрта

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;

public class DrawArcs extends JFrame {

    public DrawArcs() {
        setTitle("DrawArcs");
        add(new ArcsPanel());
    }

    /**
     * Main metod
     */
    public static void main(String[] args) {
        DrawArcs frame = new DrawArcs();
        frame.setSize(250, 300);
        frame.setLocationRelativeTo(null); // Centar okvira
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

// Klasa za crtanje luka na panelu
class ArcsPanel extends JPanel {

    @Override // crta četiri elise ventilatora
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;
        int radius = (int) (Math.min(getWidth(), getHeight()) * 0.4);

        int x = xCenter - radius;
        int y = yCenter - radius;

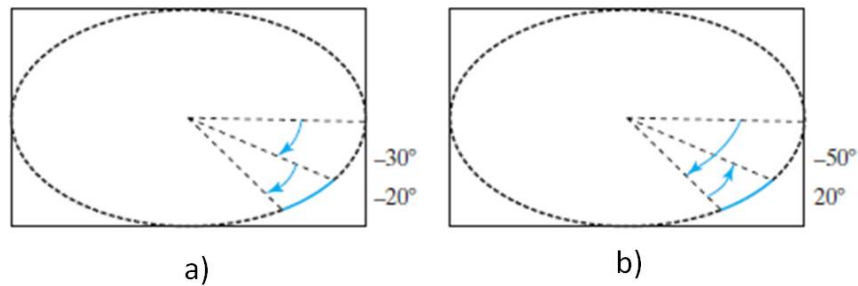
        g.fillArc(x, y, 2 * radius, 2 * radius, 0, 30);
        g.fillArc(x, y, 2 * radius, 2 * radius, 90, 30);
        g.fillArc(x, y, 2 * radius, 2 * radius, 180, 30);
        g.fillArc(x, y, 2 * radius, 2 * radius, 270, 30);
    }
}
```

NEGATIVNI UGLOVI

Uglovi navedeni kao parametri, mogu biti i negativni, ako se kreću u smeru kazaljke na satu

Uglovi navedeni kao parametri, mogu biti i negativni, ako se kreću u smeru kazaljke na satu (slika 3). Sledeća dva primera iskaza o kreiranju luka daju isti rezultat:

```
g.fillArc(x, y, 2 * radius, 2 * radius, -30, -20);
g.fillArc(x, y, 2 * radius, 2 * radius, -50, 20);
```



Slika 3.3 Uglovi mogu biti negativni

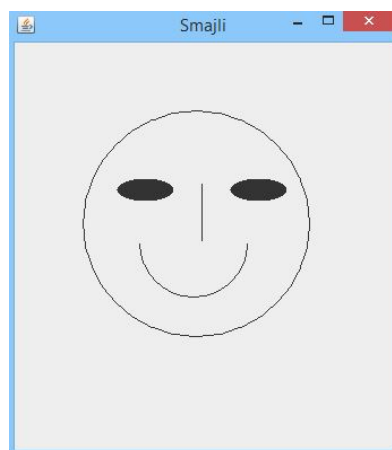
ZADACI ZA SAMOSTALNI RAD

1. Opišite metode za crtanje (bojenje) lukova.
2. Nacrtajte gornju polovinu kruga sa poluprečnikom 50.
3. Popunite bojim gornju polovinu kruga sa poluprečnikom 50 sa crvenom bojom.

PRIMER 8

Iscrtavanje kružnih isečaka

Napraviti grafički interfejs koji će na sebi da ima smajli kao na sledećoj slici:



Slika 3.4 - Zadatak 6

U ovom zadatku kao nov pojam treba pomenuti metodu drawArc koja se koristi za iscrtavanje kružnog isečka.

Od parametara prima početnu x i y koordinatu, dužinu i širinu pravougaonika u koji će isečak biti upisan, kao i broj u stepenima kojim definišemo koji deo kružnog isečka ćemo iscrtati (u ovom slučaju -180 stepeni se odnosi na donju polovinu elipse što predstavlja usta na slici).

Izmeniti klasu PanelSmiley tako da smajli bude tužan :(

Klasa Main:

```
package zadatak6;

import javax.swing.JFrame;

public class Main extends JFrame {

    public Main() {
        setSize(350, 400);
        setTitle("Smajli");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new PanelSmiley());
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

Klasa PanelSmiley:

```
package zadatak6;

import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import javax.swing.JPanel;

public class PanelSmiley extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        Font f = new Font("Helvetica", Font.BOLD, 20);
        g.setFont(f);
        g.drawOval(60, 60, 200, 200);
        g.fillOval(90, 120, 50, 20);
        g.fillOval(190, 120, 50, 20);
        g.drawLine(165, 125, 165, 175);
        g.drawArc(110, 130, 95, 95, 0, -180);
    }
}
```

PRIMER 9

Iscrtavanje i rotacija oblika

Kreirati jednu elipsu koristeći klasu `Ellipse2D`, veličine 200x100 sa počenim kordinatama 50 i 75, a potom rotirajte i crtajte tu elipsu na svakih 45 stepeni do 360. Tako da dobijete slično kao na sledećoj slici.



Slika 3.5 - Zadatak 7

U ovom zadatku je demonstrirano kako je moguće vršiti rotacije oblika koji se iscrtavaju. Dakle, kreirali smo jedan oblik elipse (klasa `Ellipse2D`) i iscrtali ga 4 puta s tim što smo u svakoj iteraciji iscrtavanja rotirali elipsu za 45 stepeni. Za rotaciju elipsu smeštamo u objekat klase `Shape`.

Za ispunjavanje unutrašnjosti elipse koristimo metod **`fill(shape)`**, a za iscrtavanje okvira metod **`draw(shape)`**. Pre iscrtavanja okvira podešavamo debljinu linije sa **`setStroke`**. Boju podešavamo sa **`setPaint`**.

Mogućnost rotiranja nam je dostupna kroz metodu `rotate` u okviru `Graphics2D` koja prima ugao rotacije crteža u radijanima i tačku oko koje se rotira. Za konvertovanje stepeni u radijane koristimo **`Math.toRadians`**.

Klasa Main:

```
package zadatak7;

import javax.swing.JFrame;

public class Main extends JFrame {
    public Main() {
        setSize(300, 300);
        setTitle("Isti krug sa razlicitim uglom");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new PanelCrtanje());
        setLocationRelativeTo(null);
        setVisible(true);
    }

    public static void main(String[] args) {
        new Main();
    }
}
```

```
}
}
```

Klasa PanelCrtanje:

```
package zadatak7;

import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Shape;
import java.awt.geom.Ellipse2D;
import javax.swing.JPanel;

public class PanelCrtanje extends JPanel {

    public PanelCrtanje() {
        setSize(300, 300);
    }

    @Override
    protected void paintComponent(Graphics grphcs) {
        super.paintComponent(grphcs);
        Graphics2D g2d = (Graphics2D) grphcs;
        int x = 50;
        int y = 75;
        int width = 200;
        int height = 100;
        Shape r1 = new Ellipse2D.Float(x, y, width, height);
        for (int angle = 0; angle <= 360; angle += 45) {
            g2d.rotate(Math.toRadians(angle), x + width / 2, y + height / 2);
            g2d.setPaint(Color.RED);
            g2d.fill(r1);
            g2d.setStroke(new BasicStroke(4));
            g2d.setPaint(Color.BLUE);
            g2d.draw(r1);
        }
    }

}
```

PRIMER CRTANJA U JAVI (VIDEO)

Crtanje linija, lukova, pravougaonika i prostornih pralelopipeda

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADACI 9 I 10

Crtanje lukova - samostalno vežbanje

ZADATAK 9:

- U razvojnom okruženju NetBeans IDE otvorite prethodni primer "Primer 8";
- Dodati desno od prikazanog smajlija, još jednog "tužnog";
- Ispod smajlija dodati odgovarajuće labele: "srećan" i "tužan", respektivno.

ZADATAK 10:

- U razvojnom okruženju NetBeans IDE otvorite prethodni primer "Primer 9";
- Ispod slike dodati labelu "ATOM";
- Na labeli podesiti font i boju teksta.
- Nacrtani mali luk kojim je podvučena kreirana labela.

▼ Poglavlje 4

Crtanje poligona i polilinja

KLASA POLYGON

Poligon je dvodimenzionalna površina ograničena proizvoljnim brojem pravih linijskih segmenata

Poligon je dvodimenzionalna površina ograničena proizvodnim brojem pravih linijskim segmentima. Poligon sadrži listu (x, y) parova koordinata, u kome svaki par definiše teme poligona, a dva susedna temena (tačke) se spajaju linijskim segmentima koji zatvaraju poligon.

Da bi ste nacrtali poligon, prvo morate da formirate objekat **Polygon**, upotrebom konstruktora klase **Polygon** (slika 1).

Sledeći programski segment definiše jedan poligon:

```
Polygon polygon = new Polygon();
polygon.addPoint(40, 20);
polygon.addPoint(70, 40);
polygon.addPoint(60, 80);
polygon.addPoint(45, 45);
polygon.addPoint(20, 60);
```

Java.awt.Polygon	
+xpoints: int[] +ypoints: int[] +npoints: int	x- koordinate svih tačaka poligona y- koordinate svih tačaka poligona Broj tačaka poligona
+Polygon() +Polygon(xpoints: int[], ypoints: int[], npoints: int) +addPoint(x: int, y: int): void +contains(x: int, y: int): boolean	Konstruktor praznog poligona. Konstruktor poligona sa specificiranim tačkama. Dodaje tačku u poligon Vraća true ako je specificirana tačka (x, y) sadržana u poligonu.

Slika 4.1 Klasa Polygon sadrži metode za crtanje poligona

Da bi se nacrtao ili obojio poligon, potrebno je upotrebiti sledeće metode klase **Graphics**:

```
drawPolygon(Polygon polygon);
fillPolygon(Polygon polygon);

drawPolygon(int[] xpoints, int[] ypoints, int npoints);
fillPolygon(int[] xpoints, int[] ypoints, int npoints)
```

CRTANJE POLIGONA I POLILINIJA

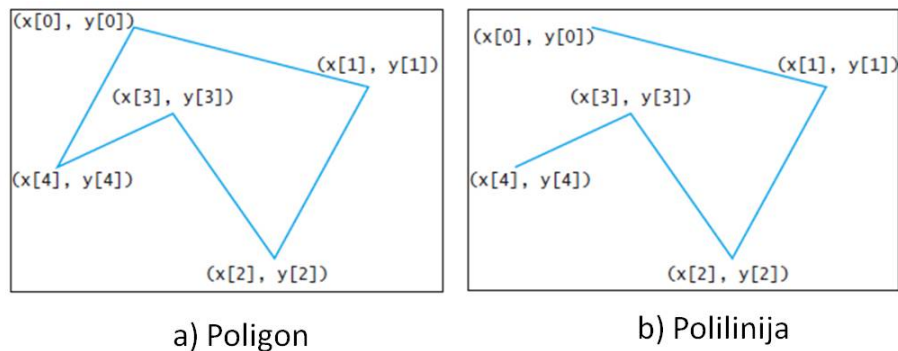
Poligon se može specificirati sa tačkama definisanih nizom x i y koordinata i onda nacrtati sa metodom `drawPolygon()`, a polinija metodom `drawPolyline()`.

Poligon se može specificirati sa tačkama definisanih nizom x i y koordinata i onda nacrtati sa metodom **`drawPolygon()`**:

```
int x[] = {40, 70, 60, 45, 20};  
int y[] = {20, 40, 80, 45, 60};  
g.drawPolygon(x, y, x.length)
```

Poligon se crta spajanjem linijama svih tačaka poligona, kao i poslednje i prve navdene tačke poligona, čime se poligon zatvara (slika 2 a).

Polilinija je skup pravih linijskih segmenata koji spajaju dati skup tačaka, definisanih nizom x i y koordinata (slika 2 b).



Slika 4.2 poligona sa metodom `drawPolygon` i polilinije sa metodom `drawPolyline`

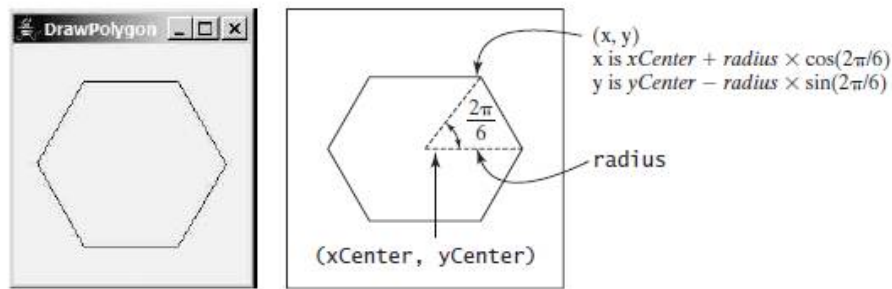
Crtanje poligona sa metodom **`drawPoligon()`**, a polilinija metodom **`drawPolyline()`**.

```
int x[] = {40, 70, 60, 45, 20};  
int y[] = {20, 40, 80, 45, 60};  
g.drawPolyline(x, y, x.length);
```

PRIMER 10 - CRTANJE HEKSAGONA

Heksagon je poligon sa šest jednakih linijskih segmenata.

Listing nam pokazuje primer crtanja heksagona prikazanog na slici 3 ..



Slika 4.3 Dobijen heksagon izvršenjem programa DrawPolygon

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import java.awt.Graphics;
import java.awt.Polygon;

public class DrawPolygon extends JFrame {

    public DrawPolygon() {
        setTitle("DrawPolygon");
        add(new PolygonsPanel());
    }

    /**
     * Main metod
     */
    public static void main(String[] args) {
        DrawPolygon frame = new DrawPolygon();
        frame.setSize(200, 250);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

// Crtanje poligona na panelu
class PolygonsPanel extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;
        int radius = (int) (Math.min(getWidth(), getHeight()) * 0.4);

        // Kreiranje objekta Polygon
        Polygon polygon = new Polygon();

        // Dodavanje tačaka poligona sa datim redosledom
        polygon.addPoint(xCenter + radius, yCenter);
        polygon.addPoint((int) (xCenter + radius
```

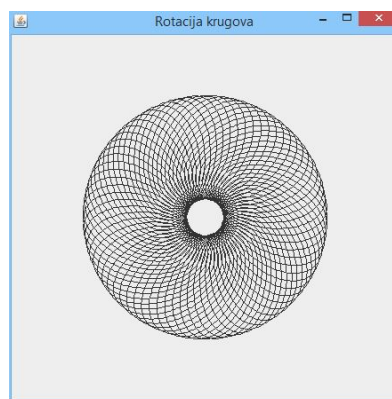
```
        * Math.cos(2 * Math.PI / 6)), (int) (yCenter - radius
        * Math.sin(2 * Math.PI / 6)));
    polygon.addPoint((int) (xCenter + radius
        * Math.cos(2 * 2 * Math.PI / 6)), (int) (yCenter - radius
        * Math.sin(2 * 2 * Math.PI / 6)));
    polygon.addPoint((int) (xCenter + radius
        * Math.cos(3 * 2 * Math.PI / 6)), (int) (yCenter - radius
        * Math.sin(3 * 2 * Math.PI / 6)));
    polygon.addPoint((int) (xCenter + radius
        * Math.cos(4 * 2 * Math.PI / 6)), (int) (yCenter - radius
        * Math.sin(4 * 2 * Math.PI / 6)));
    polygon.addPoint((int) (xCenter + radius
        * Math.cos(5 * 2 * Math.PI / 6)), (int) (yCenter - radius
        * Math.sin(5 * 2 * Math.PI / 6)));

    // Crtanje poligona
    g.drawPolygon(polygon);
}
}
```

PRIMER 11

Rotacija oblika pomoću klase AffineTransform

Kreirati jednu elipsu koristeći klasu Eclipse2D i Shape veličine 80,130 sa početnom tačkom 0,0. Rotirati elipsu i crtati je od 0 do 360 stepeni sa povećanjem od 5 stepeni tako da centar transformacije bude na sredini ekrana.



Slika 4.4 - Zadatak 8

U ovom zadatku se demonstrira rad sa klasom **AffineTransform** koja nam pruža mogućnosti da vršimo **transformaciju** pojedinačnih objekata koji nasleđuju klasu **Shape**. Prvo pomeramo elipsu tako da gornje levo teme pravougaonika koji je ograničava bude u centru prozora. Koristimo **AffineTransform.getTranslateInstance(x,y)**. Tako dobijamo objekat transformacije koji predstavlja translaciju. Zatim na tu translaciju dodajemo i rotaciju za broj stepeni određen sa *i*. Rotacija se vrši metodom **rotate** oko centra prozora. Na kraju iscrtavamo novi oblik dobijen opisanim transformacijama pomoću **draw** i **createTransformedShape**.

Klasa Main:

```
package zadatak8;

import javax.swing.JFrame;

public class Main extends JFrame {
    public Main() {
        setSize(450, 450);
        setTitle("Rotacija krugova");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setContentPane(new PanelCrtanje());
        setLocationRelativeTo(null);
        setVisible(true);
    }
    public static void main(String[] args) {
        new Main();
    }
}
```

Klasa PanelCrtanje:

```
package zadatak8;

import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.geom.AffineTransform;
import java.awt.geom.Ellipse2D;
import javax.swing.JPanel;

public class PanelCrtanje extends JPanel {

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        Graphics2D g2 = (Graphics2D) g;
        Ellipse2D e = new Ellipse2D.Double(0, 0, 80, 130);
        for (double i = 0; i < 360; i += 5) {
            AffineTransform at = AffineTransform.getTranslateInstance(getWidth() /
2, getHeight() / 2);
            at.rotate(Math.toRadians(i));
            g2.draw(at.createTransformedShape(e));
        }
    }
}
```

ZADACI 11 I 12

Vežba crtanja poligona

Zadatak 11 - Nacrtajte poligon koji povezuje sledeće tačke: (20, 40), (30, 50), (40, 90), (90,10), (10, 30).

Zadatak 12 - Kreirajte **Polygon** objekat i dodajte tačke: (20, 40), (30, 50), (40, 90), (90, 10), (10, 30) u ovom redosledu. Popunite poligon crvenom bojom. Nacrtajte poliliniju koja povezuje sve ove tačke sa žutom bojom

CRTANJE SA JAVOM (VIDEO)

Video koristi različite grafičke komponente za kreiranje crteža.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 5

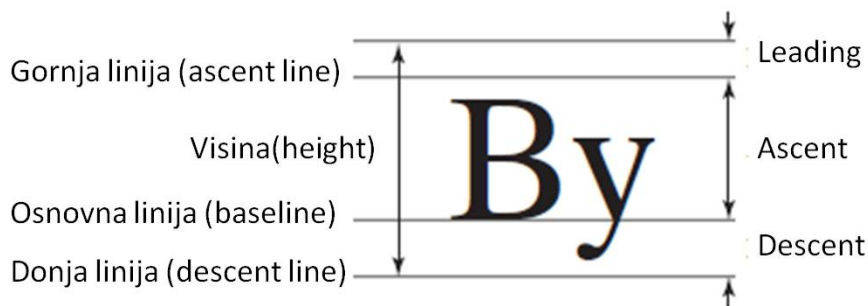
Centriranje teksta upotrebom klase FontMetrics

PRIMENA KLASSE FONTMETRIKS

FontMetrics je apstraktna klasa. Da bi se dobio objekat FontMetrics za određeni font, upotrebljava se metod `getFontMetrics()` koji je definisan u klasi `Graphic`

Možete prikazati tekst na bilo kom položaju na panelu. Ako želite da ga centrirate u odnosu na panel, onda treba da koristite klasu **FontMetrics** da bi premerili širinu i visinu teksta određenog fonta. Klasa **FontMetrics** može da meri sledeće atribute datog fonta (slika 1):

- Leding (Leading)- prostor između linija teksta
- Ascent - rastojanje od osnovne linije do gornje linije pisma
- Descent - rastojanje od donje linije do donje linije pisma
- Visina (Height) - zbir ledinga, ascenta i descenta



Slika 5.1 Osnovne karakteristike pisma, tj. fonta

FontMetrics je apstraktna klasa. Da bi se dobio objekat **FontMetrics** za određeni font, upotrebljava se metod **`getFontMetrics()`** koji je definisan u klasi **Graphics**:

```
public FontMetrics getFontMetrics(Font font)
```

Kada se dobije **FontMetrics** objekat za određeni font, onda se iz njega dobijaju sve veličine tog fonta, tj. njegova metrika. Da bi se dobila metrika tekućeg fonta, koristi se **`getMetrics(Font font)`** i isti metod, ali bez argumenata:

```
public FontMetrics getFontMetrics()
```

Možete koristiti sledeće metode klase **FontMetrics** da bi dobili atribut fonta širinu teksta kada se ispiše određeni font:

```
public int getAscent() // Vraća ascent
public int getDescent() // Vraća descent
public int getLeading() // Vraća leding
public int getHeight() // Vraća visinu
public int stringWidth(String str) // vraća širinu teksta
```

PRIMER 12 - KORIŠĆENJE KLASSE FONTMETRICS

Postavljanje tekst u sredinu panela za poruke

Na slici 2 je primer prikaza poruke na sredini panela, koji se dobija programom čiji se listing ovde prikazuje.



Slika 5.2 Dobijena poruka na sredini panela korišćenjem programa TestCenterMessage

```
import javax.swing.*;
import java.awt.*;

public class TestCenterMessage extends JFrame {

    public TestCenterMessage() {
        CenterMessage messagePanel = new CenterMessage();
        add(messagePanel);
        messagePanel.setBackground(Color.WHITE);
        messagePanel.setFont(new Font("Californian FB", Font.BOLD, 30));
    }

    /**
     * Main metod
     */
    public static void main(String[] args) {
        TestCenterMessage frame = new TestCenterMessage();
        frame.setSize(300, 150);
        frame.setTitle("CenterMessage");
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```



```

        frame.setVisible(true);
    }
}

class CenterMessage extends JPanel {

    @Override
    /**
     * Bojenje poruke
     */
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Dobijanje metrike (mere) fonta za tekući font
        FontMetrics fm = g.getFontMetrics();

        // Određivanje centralnog položaja za prikaz teksta
        int stringWidth = fm.stringWidth("Welcome to Java");
        int stringAscent = fm.getAscent();

        // Određivanje položaja prvog levog karaktera na osnovnoj liniji
        int xCoordinate = getWidth() / 2 - stringWidth / 2;
        int yCoordinate = getHeight() / 2 + stringAscent / 2;

        g.drawString("Welcome to Java", xCoordinate, yCoordinate);
    }
}

```

UNOŠENJE TEKSTOVA (VIDEO)

Video pokazuje ubacivanje poruka u vaš GUI

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADATAK 13

Samostalno vežbanje korišćenja klase FontMetrics.

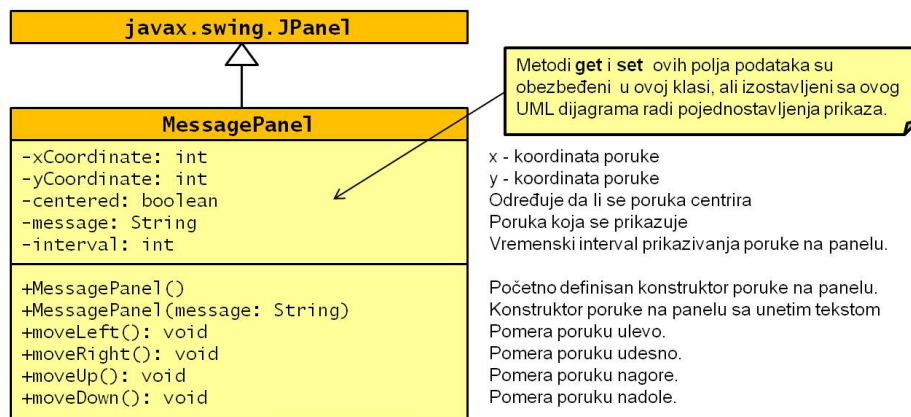
ZADATAK 13:

- U razvojnom okruženju NetBeans IDE otvorite prethodni primer "Primer 12";
- Ispod unetog teksta "Welcome to Java", na identičan način, a manjim fontom dodati nov tekst "By (vaše ime i prezime)";
- Pokrenuti izmenjeni program i prikazati rezultate izvršavanja.

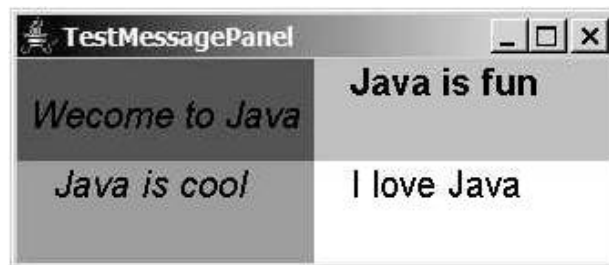
PRIMER 13 - KLASA MESSAGEPANEL

Ova studija slučaja razvija korisnu klasu koja prikazuje poruku na panelu. Klasa omogućava korisniku da postavi poruku u centar panela i da pomera poruku u određenim intervalima.

Na slici 1 je prikazan UML dijagram klase **MessagePanel** koja se koristi za prikaz četiri poruka na panelu, Na slici 2 prikazan je panel sa četiri poruke dobijen programim čiji je listing ovde prikazan.



Slika 5.3 UML dijagram klase MessagePanel



Slika 5.4 sa četiri poruke dobijen programom TestMessagePanel.java

```

import java.awt.*;
import javax.swing.*;

public class TestMessagePanel extends JFrame {

    public TestMessagePanel() {
        MessagePanel messagePanel1 = new MessagePanel("Welcome to Java");
        MessagePanel messagePanel2 = new MessagePanel("Java is fun");
        MessagePanel messagePanel3 = new MessagePanel("Java is cool");
        MessagePanel messagePanel4 = new MessagePanel("I love Java");
        messagePanel1.setFont(new Font("SansSerif", Font.ITALIC, 20));
        messagePanel2.setFont(new Font("Courier", Font.BOLD, 20));
        messagePanel3.setFont(new Font("Times", Font.ITALIC, 20));
        messagePanel4.setFont(new Font("Californian FB", Font.PLAIN, 20));
        messagePanel1.setBackground(Color.RED);
    }
}

```

```

        messagePanel2.setBackground(Color.CYAN);
        messagePanel3.setBackground(Color.GREEN);
        messagePanel4.setBackground(Color.WHITE);
        messagePanel1.setCentered(true);

        setLayout(new GridLayout(2, 2));
        add(messagePanel1);
        add(messagePanel2);
        add(messagePanel3);
        add(messagePanel4);
    }

    public static void main(String[] args) {
        TestMessagePanel frame = new TestMessagePanel();
        frame.setSize(300, 200);
        frame.setTitle("TestMessagePanel");
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

```

LISTING PROGRAMA MESSAGEPANEL.JAVA

Metod `paintComponent()` prikazuje centriranu poruku, ako je svojstvo centriranja (centered) true (linija 91). String objekat `message` definiše početni tekst poruke

Metod **`paintComponent()`** prikazuje centriranu poruku, ako je svojstvo centriranja (centered) true (linija 91). String objekat **`message`** (linija 8) definiše početni tekst poruke „Welcome to Java“. Da nije bilo ove inicijalizacije teksta poruke, sistem bi generisao izuzetak izvršenja **`NullPointerException`** koji se javlja kada kreirate objekat **`MessagePanel`**, upotrebom konstruktora bez argumenta, jer objekat **`message`** bi bio **`null`** u liniji 102.

U klasi **`MessagePanel`** koriste se atributi **`xCoordinate`** i **`yCoordinate`** umesto oznaka `x` i `y`, jer se ove oznake već koriste u klasi **`Component`** i vraćaju se kao upotrebom metoda **`getX()`** i **`getY()`**.

Klasa **`Component`** koristi metode **`setBackground()`**, **`setForeground()`**, i **`setFont()`** radi postavljanja boja i fontova u celoj komponenti. Ako želite da ispišete nekoliko poruka na panelu sa različitim bojama i fontovima, trebalo bi da upotrebite **`setColor`** i **`setFont`** metode klase **`Graphics`**, a da bi postavili boju i font tekućeg crteža, tj. teksta.

Ovde razvijenu klasu **`MessagePanel`** možete uvek upotrebljavati kada želite da prikazete neku poruku na panelu. Zbog mogućih drugih primena, ovde je klasa **`MessagePanel`** razvijena i sa

metodima i atributima koji nisu bili neophodni u analiziranom primeru, jer mogu biti korisni i potrebni u drugim primerima.

```
import java.awt.FontMetrics;
import java.awt.Dimension;
import java.awt.Graphics;
import javax.swing.JPanel;

public class MessagePanel extends JPanel {

    /**
     * Poruka koja se prikazuje
     */
    private String message = "Welcome to Java";

    /**
     * x-koordinata početka poruke
     */
    private int xCoordinate = 20;

    /**
     * y-koordinata početka poruke
     */
    private int yCoordinate = 20;

    /**
     * Određivanje da li se želi prikaz poruke u sredini ekrana
     */
    private boolean centered;

    /**
     * Vremenski interval horizontalnog i vertikalnog pomeranja poruke
     */
    private int interval = 10;

    /**
     * Konstruktor sa početnim vrednostima svojstava
     */
    public MessagePanel() {
    }

    /**
     * Konstruktor panela sa porukom definisane poruke
     */
    public MessagePanel(String message) {
        this.message = message;
    }

    /**
     * Vraćanje poruke
     */
    public String getMessage() {
        return message;
    }

    /**
```

```
* Unos nove poruke
*/
public void setMessage(String message) {
    this.message = message;
    repaint();
}

/**
 * Vraćanje xCoordinator
 */
public int getXCoordinate() {
    return xCoordinate;
}

/**
 * Postavlja novu xCoordinator
 */
public void setXCoordinate(int x) {
    this.xCoordinate = x;
    repaint();
}

/**
 * Vraća yCoordinator
 */
public int getYCoordinate() {
    return yCoordinate;
}

/**
 * Postavlja novu yCoordinator
 */
public void setYCoordinate(int y) {
    this.yCoordinate = y;
    repaint();
}

/**
 * Vraća centered
 */
public boolean isCentered() {
    return centered;
}

/**
 * Postavlja true ili false za centriranje teksta
 */
public void setCentered(boolean centered) {
    this.centered = centered;
    repaint();
}

/**
```

```

    * Vraća interval
    */
    public int getInterval() {
        return interval;
    }

    /**
     * Postavlja novi interval
     */
    public void setInterval(int interval) {
        this.interval = interval;
        repaint();
    }

    @Override
    /**
     * Bojenje poruke
     */
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        if (centered) {
            // Dobija metriku fonta za tekući font
            FontMetrics fm = g.getFontMetrics();

            // Nalazi položaj centra na ekranu
            int stringWidth = fm.stringWidth(message);
            int stringAscent = fm.getAscent();
            // Uzima položaj prvog levog karaktera na osnovnoj liniji
            xCoordinate = getWidth() / 2 - stringWidth / 2;
            yCoordinate = getHeight() / 2 + stringAscent / 2;
        }

        g.drawString(message, xCoordinate, yCoordinate);
    }

    /**
     * Pomera poruku ulevo
     */
    public void moveLeft() {
        xCoordinate -= interval;
        repaint();
    }

    /**
     * Pomera poruku udesno
     */
    public void moveRight() {
        xCoordinate += interval;
        repaint();
    }
}

/**

```

```

    * Pomeru poruku nagore
    */
    public void moveUp() {
        yCoordinate -= interval;
        repaint();
    }

    /**
     * Pomeru poruku nadole
     */
    public void moveDown() {
        yCoordinate += interval;
        repaint();
    }

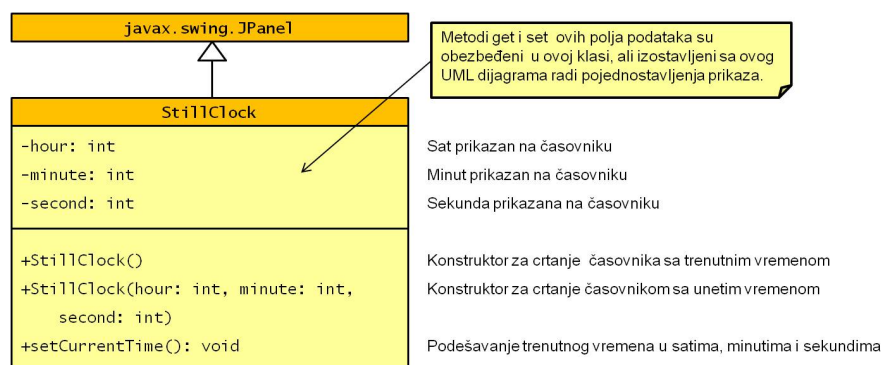
    @Override
    /**
     * Prekriva get metod zbog preferredSize
     */
    public Dimension getPreferredSize() {
        return new Dimension(200, 30);
    }
}

```

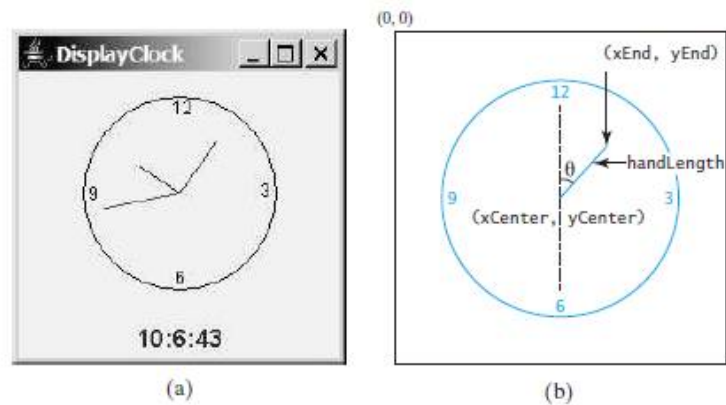
PRIMER 14 - KLASA STILLCLOCK

Ova studija slučaja razvija klasu koja prikazuje analogni sat na panelu

Na slici 1 prikazan je UML dijagram klase **StillClock** koju ovde razvijamao za crtanje analognog časovnika. Ovde se prikazuje listing programa, tj. klasa **DisplayClock** koja prikazuje analogni časovnik, prikazan na slici 2 .



Slika 5.5 Klasa StillClock za crtanje analognog časovnika



Slika 5.6 Izgled dobijeno analognog časovnika izvršenjem programa DisplayClock

```
import java.awt.*;
import javax.swing.*;

public class DisplayClock extends JFrame {

    public DisplayClock() {
        // Kreiranje analognog časovnika sa trenutnim vremenom
        StillClock clock = new StillClock();

        // Prikaz sata, minuta i sekunde na panelu sa porukom
        MessagePanel messagePanel = new MessagePanel(clock.getHour()
            + ":" + clock.getMinute() + ":" + clock.getSecond());
        messagePanel.setCentered(true);
        messagePanel.setForeground(Color.blue);
        messagePanel.setFont(new Font("Courier", Font.BOLD, 16));

        // Dodavanje sata i panela sa porukom u okvir
        add(clock);
        add(messagePanel, BorderLayout.SOUTH);
    }

    public static void main(String[] args) {
        DisplayClock frame = new DisplayClock();
        frame.setTitle("DisplayClock");
        frame.setSize(300, 350);
        frame.setLocationRelativeTo(null); // Centar okvira
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

NAČIN CRTANJA ANALOGNOG SATA

Da bi se nacrtao analogni časovnik, potrebno je nacrtati krug i tri skazaljke, za pokazivanja sata, minute i sekunde

Klasa DisplayClock koristi klasu **MessagePanel**, razvijenu u Studiji slučaja: **Klasa MessagePanel**, kao i klasu **StillClock**, čiji je listing ovde prikazuje.

Da bi se nacrtao analogni časovnik, potrebno je nacrtati krug i tri skazaljke, za pokazivanja sata, minute i sekunde. Da bi se nacrtala skazaljka, potrebno je da definišete dva kraja jedne linije. Kao što je pokazano na slici 30b, jedan kraj te linije je centar časovnika ($xCenter$, $yCenter$), a drugi kraj ($xEnd$, $yEnd$) se određuje

```
xEnd = xCenter + handLength * sin(θ)
yEnd = yCenter - handLength * cos(θ)
```

Kako minut ima 60 sekundi, ugao pomeranja sekundare je: $second \times (2\pi/60)$.

Položaj skazaljke koja pokazuje minute se određuje u zavisnosti i od položaja sekundare, tj. od vrednosti: $minute + second/60$. Kako 60 minuta čine jedan sat, ugao položaja skazaljke koja pokazuje minute se određuje sledećom formulom:

```
(minute + second/60) * (2π/60)
```

Kako jedan krug časovnika odgovara vremenu od 12 sati, to se ugao položaja skazaljke koja pokazuje sate određuje sledećom formulom:

```
(hour + minute/60 + second/(60 * 60)) * (2π/12)
```

Radi pojednostavljenja proračuna uglova položaja skazaljki koje pokazuju sate i minute, možemo izostaviti uticaj sekundi. Sa takvim uprošćenjem, krajnje tačke skazaljki za označavanje sati, minuta i sekundi se mogu izračunati prema sledećim formulama:

```
xSecond = xCenter + secondHandLength * sin(second * (2π/60))
ySecond = yCenter - secondHandLength * cos(second * (2π/60))

xMinute = xCenter + minuteHandLength * sin(minute * (2π/60))
yMinute = yCenter - minuteHandLength * cos(minute * (2π/60))

xHour = xCenter + hourHandLength * sin((hour + minute/60) * (2π/12))
yHour = yCenter - hourHandLength * cos((hour + minute/60) * (2π/12))
```

PRIMER 14 - LISTING KLASSE STILLCLOCK

Program dozvoljava prilagođavanje veličine časovnika kada dođe do promene okvira za prikazivanje časovnika

Program dozvoljava prilagođavanje veličine časovnika kada dođe do promene okvira za prikazivanje časovnika. Uvek kada se promeni veličina okvira, metod **paintComponent** se automatski poziva radi crtanja i bojenja časovnika sa novim dimenzijama. Metod **paintComponent** prikazuje sat u proporcionalnim deimenzijama u odnosu na panel koji ima širinu i svistu određenih sa metodime: **getWidth()** i **getHeight()**, kao što je pokazanu u linijama 60-63 u listingu klase **StillClock**.

```
import java.awt.*;
import javax.swing.*;
import java.util.*;

public class StillClock extends JPanel {

    private int hour;
    private int minute;
    private int second;

    /**
     * Konstruktor sata u trenutnom vremenu
     */
    public StillClock() {
        setCurrentTime();
    }

    /**
     * Konstruktor sata sa određenim satom, minutom i sekundom
     */
    public StillClock(int hour, int minute, int second) {
        this.hour = hour;
        this.minute = minute;
        this.second = second;
    }

    /**
     * Vraća sat
     */
    public int getHour() {
        return hour;
    }

    /**
     * Postavlja novi sat
     */
    public void setHour(int hour) {
        this.hour = hour;
        repaint();
    }

    /**
     * Vraća minute
     */
    public int getMinute() {
        return minute;
    }

    /**
     * Postavlja nove minute
     */
    public void setMinute(int minute) {
        this.minute = minute;
    }
}
```

```

        repaint();
    }

    /**
     * vraća sekund
     */
    public int getSecond() {
        return second;
    }

    /**
     * Postavlja novi sekund
     */
    public void setSecond(int second) {
        this.second = second;
        repaint();
    }

    @Override
    /**
     * Crta sat
     */
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        // Inicijalizacija parametre sata
        int clockRadius
            = (int) (Math.min(getWidth(), getHeight()) * 0.8 * 0.5);
        int xCenter = getWidth() / 2;
        int yCenter = getHeight() / 2;

        // Crtanje kruga
        g.setColor(Color.BLACK);
        g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
            2 * clockRadius, 2 * clockRadius);
        g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
        g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
        g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
        g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);

        // Crtanje sekundare
        int sLength = (int) (clockRadius * 0.8);
        int xSecond = (int) (xCenter + sLength
            * Math.sin(second * (2 * Math.PI / 60)));
        int ySecond = (int) (yCenter - sLength
            * Math.cos(second * (2 * Math.PI / 60)));
        g.setColor(Color.red);
        g.drawLine(xCenter, yCenter, xSecond, ySecond);

        // Crtanje skazaljke sa minutima
        int mLength = (int) (clockRadius * 0.65);
        int xMinute = (int) (xCenter + mLength
            * Math.sin(minute * (2 * Math.PI / 60)));
    }

```

```

        int yMinute = (int) (yCenter - mLength
            * Math.cos(minute * (2 * Math.PI / 60)));
        g.setColor(Color.blue);
        g.drawLine(xCenter, yCenter, xMinute, yMinute);

        // Crtanje skazake sa satima
        int hLength = (int) (clockRadius * 0.5);
        int xHour = (int) (xCenter + hLength
            * Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
        int yHour = (int) (yCenter - hLength
            * Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
        g.setColor(Color.green);
        g.drawLine(xCenter, yCenter, xHour, yHour);
    }

    public void setCurrentTime() {
        // Konstruktor kalendara za trenutni datum i trenutno vreme
        Calendar calendar = new GregorianCalendar();

        // Postavljanje trenutnog vremena u satima, minutima i sekundima
        this.hour = calendar.get(Calendar.HOUR_OF_DAY);
        this.minute = calendar.get(Calendar.MINUTE);
        this.second = calendar.get(Calendar.SECOND);
    }

    @Override
    public Dimension getPreferredSize() {
        return new Dimension(200, 200);
    }
}

```

ZADACI 14 I 15

Sljedeći program bi trebalo da prikaže poruku na panelu, ali se to nije dogodilo. Ispravite greške!

Zadatak 14. Ako se objekat **message** ne inicijalizuje u liniji 8 u listing programa MessagePanel.java, šta će se desiti ako kreirate MessagePanel objekat korišćenjem konstruktora bez argumenata?

Zadatak 15. Sljedeći program bi trebalo da prikaže poruku na panelu, ali se to nije dogodilo. Ispravite greške

```

public class TestDrawMessage extends javax.swing.JFrame {

    public TestDrawMessage() {
        add(new DrawMessage());
    }

    public static void main(String[] args) {

```

```
        javax.swing.JFrame frame = new TestDrawMessage();
        frame.setSize(100, 200);
        frame.setVisible(true);
    }
}

class DrawMessage extends javax.swing.JPanel {

    protected void PaintComponent(java.awt.Graphics g) {
        super.paintComponent(g);
        g.drawString("Welcome to Java", 20, 20);
    }
}
```

▼ Poglavlje 6

Prikazivanje slika

KLASA GRAPHICS

Možete prikazati slike u grafičkom kontekstu

Možete da naučite kako da kreirate ikonu sa slikom i da ubacite u vaš korisnički interfejs, po potrebi. Na primer, sledeća dva iskaza kreiraju ikonu sa slikom i prikazuju je u formi natpisa:

```
ImageIcon imageIcon = new ImageIcon("image/us.gif");
JLabel jlblImage = new JLabel(imageIcon);
```

Ikona slike se prikazuje sa fiksnom dimenzijom slike. Da bi se prikazala sa promenljivom veličinom, potrebno je da koristite **java.awt.Image** klasu. Sa njenom primenom, slika se kreira metodom **getImage()** na sledeći način:

```
Image image = imageIcon.getImage();
```

Korišćenjem natpisa (engl. label) kao prostora za prikazivanje slika, je jednostavno i prigodno, ali tada nemate potpunu kontrolu prikazivanja slike. Fleksibilniji način za prikazivanje slika je da upotrebite metod **drawImage** klase **Graphics** na panelu. Na slici 1 prikazano je četiri verzije drawImage metoda.

Java.awt.Graphics	
+drawImage(image: Image, x: int, y: int, bgcolor: Color, observer: ImageObserver): void	Crta sliku na specifikiranom položaju.(x, y) je gornji levi ugao prostora za unos grafike. Pikseli se unose sa bojom bgcolor. Obesrver je objekat na kome se slika prikazuje. Slika se odseca ako je veća od prostora na kome se crta. Isto kao prethodni metod, sem što ne definiše boju pozadije.
+drawImage(image: Image, x: int, y: int, observer: ImageObserver): void	
+drawImage(image: Image, x: int, y: int, width: int, height: int, observer: ImageObserver): void	Crta proporcionalnu verziju slike koja popunjava ceo raspoloživi prostor u specifikiranom pravougaoniku.
+drawImage(image: Image, x: int, y: int, width: int, height: int, bgcolor: Color, observer: ImageObserver): void	Isto kao prethodni metod, sem što obezbeđuje boju pozadine, iza slike koja se crta(prikazuje).

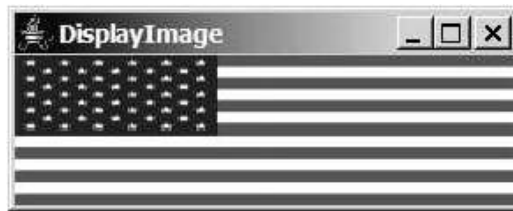
Slika 6.1.1 Metod drawImage omogućava da se objekat Graphics prikaže kao slika na nekoj GUI komponenti

ImageObserver određuje GUI komponentu koja prima informaciju o slici koja se konstruiše, radi prikazivanja na toj GUI komponenti. Da bi se nacrtala slika na Swing komponenti, kao što je **JPanel**, potrebno je da prekrijete metod **paintComponent** da bi saopštili komponenti kako da prikaže vašu sliku na panelu.

PRIMER 15 - LISTING KLASE DISPLAYIMAGE

Metod `drawImage()` prikazuje sliku učitano preko odgovarajuće datoteke, koja popunjava ceo panel

Listing prikazuje kod koji prikazuje sliku preuzetu sa datoteke **image/us.gif**. Datoteka **image/us.gif** (linija 20) je u direktorijumu klase. Objekat `Image` se dobija u liniji 21. Metod **`drawImage()`** prikazuje sliku koja popunjava ceo panel, kao što je prikazano na slici 2



Slika 6.1.2 Prikazana slika izvršenjem programa DisplayImage

asd

```
import java.awt.*;
import javax.swing.*;

public class DisplayImage extends JFrame {

    public DisplayImage() {
        add(new ImagePanel());
    }

    public static void main(String[] args) {
        JFrame frame = new DisplayImage();
        frame.setTitle("DisplayImage");
        frame.setSize(300, 300);
        frame.setLocationRelativeTo(null); // Center the frame
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}

class ImagePanel extends JPanel {

    private ImageIcon imageIcon = new ImageIcon("image/us.gif");
    private Image image = imageIcon.getImage();

    @Override
    /**
     * Draw image on the panel
     */
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);

        if (image != null) {
```

```
        g.drawImage(image, 0, 0, getWidth(), getHeight(), this);  
    }  
}
```

PRIKAZIVANJE SLIKA (VIDEO)

Video koristi ImageIcon klasu za prikazivanje fotografija i crteža na ekranu.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADATAK 16

Prikazivanje slika - samostalno vežbanje

Po uzoru na prethodni primer:

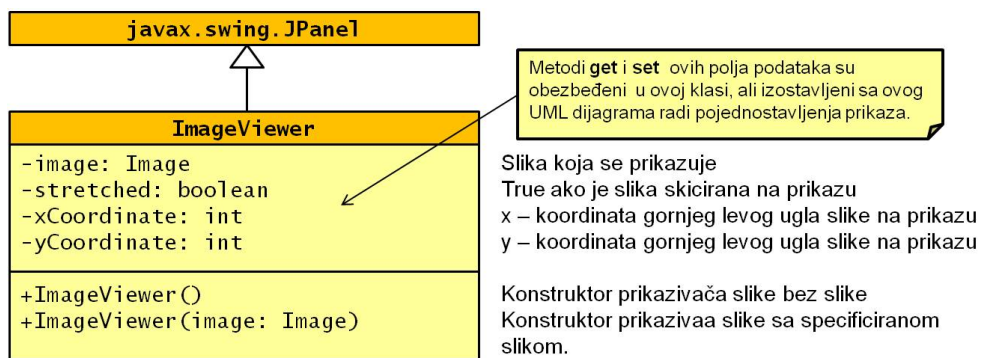
- Izaberite dve slike;
- Postavite slike na panel, jednu ispod druge;
- Ispod svake slike postavite i komentar, koji je opisuje, u formi labela.

▼ 6.1 Klasa ImageViewer

UML MODEL KLASA IMAGEVIEWER

Klasa ImageViewer prikazuje određenu sliku na nekom panelu.

Često se prikazuju slike prilikom programiranja u Javi. Ova studija slučaja razvija višestruko upotrebljivu komponentu pod nazivom **ImageViewer** koja prikazuje određenu sliku na nekom panelu. Slika 1 prikazuje svojstva ove klase, a slika 2 (u narednoj sekciji) listing programa koji prikazuje šest zastava, uz pomoć klase ImageViewer.



Slika 6.2.1 : Klasa ImageViewer za prikazivanje slika na panelu

PRIMER16 - PRIKAZ SLIKA ŠEST ZASTAVA

Metod paintComponent prikazuje sliku na panelu

Za prikazivanje slika možete koristiti i Swing komponente, kao što su **JLabel** i **JBUTTON**, ali slike na njima onda nisu promenljive, tj. nisu fleksibilne jer se ne prilagođavaju veličini prostora u kome se prikazuju. S druge strane, primenom klase **ImageViewer**, slike postaju fleksibilne, tj. prilagođavaju se prostoru koji im stoji na raspolaganju.

Na slici 2 prikazane su šest slika (šest zastava) dobijene primenom klase SixFlags koja koristi klasu **ImageViewer**, čiji je listing ovde prikazan. Metod **paintComponent** (linije 27-36) prikazuje sliku na panelu. Linija 30 osigurava da slika nije **null** pre prikazivanja. Linija 31 proverava da li je slika prilagodljiva ili nije



Slika 6.2.2 SixFlags

Slike zastava možete zameniti i drugim slikama.

```

import javax.swing.*;
import java.awt.*;

public class SixFlags extends JFrame {

    public SixFlags() {
        Image image1 = new ImageIcon("image/us.gif").getImage();
        Image image2 = new ImageIcon("image/ca.gif").getImage();
        Image image3 = new ImageIcon("image/india.gif").getImage();
    }
  
```

```

        Image image4 = new ImageIcon("image/uk.gif").getImage();
        Image image5 = new ImageIcon("image/china.gif").getImage();
        Image image6 = new ImageIcon("image/norway.gif").getImage();

        setLayout(new GridLayout(2, 0, 5, 5));
        add(new ImageViewer(image1));
        add(new ImageViewer(image2));
        add(new ImageViewer(image3));
        add(new ImageViewer(image4));
        add(new ImageViewer(image5));
        add(new ImageViewer(image6));
    }

    public static void main(String[] args) {
        SixFlags frame = new SixFlags();
        frame.setTitle("SixFlags");
        frame.setSize(400, 320);
        frame.setLocationRelativeTo(null); // Centar okvira
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

```

ZADATAK 17

Proverite vaše znanje.

1. Kako kreirate objekat **Image** iz objekta **ImageIcon**?
2. Kako kreirate objekat **ImageIcon** iz objekta **Image**?
3. Opišite **drawImage** metod u **Graphics** klasi.
4. Objasnite razliku između slika na **JLabel** objektu i na **JPanel** objektu?
5. Koji paket sadrži **ImageIcon** i koji sadrži **Image**?

IMAGEVIEWER (VIDEO)

*Video pokazuje korišćenje **ImageViewer** klase, Video je bez zvuka, ali je titlovan.*

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

▼ Poglavlje 7

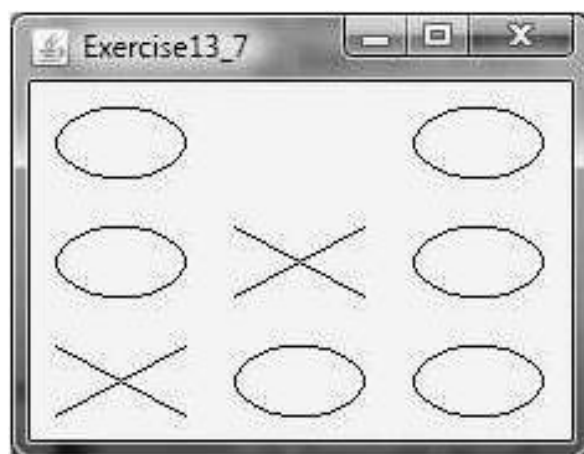
Domaći zadaci

ZADACI 1 I 2

Pokušajte da zadatke 1 i 2 samostalno rešite.

Zadatak 1

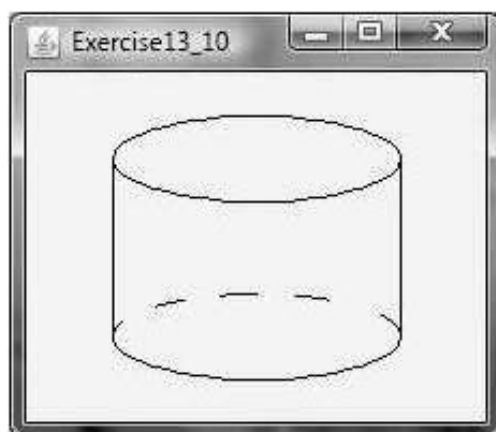
Napraviti program koji iscrtava ono što vidite na sledećoj slici:



Slika 7.1 - Zadatak 1

Zadatak 2

Napraviti program koji iscrtava ono što vidite na sledećoj slici:



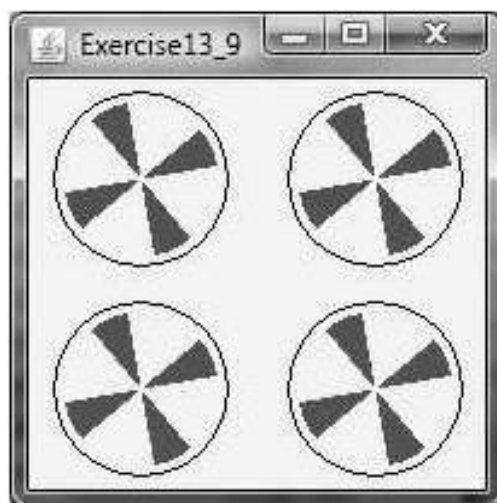
Slika 7.2 - Zadatak 2

ZADACI 3 I 4

Pokušajte da zadatak 3 i 4 samostalno rešite.

Zadatak 3

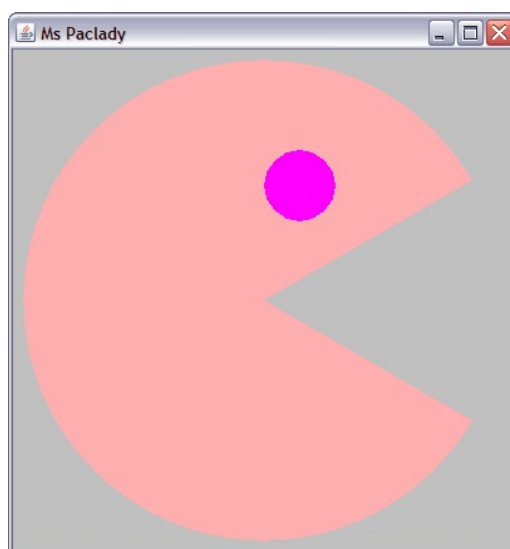
Napraviti program koji iscrtava ono što vidite na sledećoj slici:



Slika 7.3 - Zadatak 3

Zadatak 4

Napraviti program koji iscrtava ono što vidite na sledećoj slici:



Slika 7.4 Zadatak 4

✓ Zaključak

REZIME LEKCIJE

Pouke ove lekcije

1. Svaka komponenta ima svoj sopstveni koordinatni sistem sa koordinatnim početkom (0,0) u gornjem levom uglu prozora. U Javi, x-koordinata raste udesno, a y-koordinata – nadole.
2. Uvek kada komponenta (npr., dugme, natpis, ili panel) se prikaže, JVM automatski kreira **Graphics** objekat za tu komponentu na originalnoj platformi i pruža taj objekat radi pozivanja metoda **paintComponent** koja prikazuje crteže i slike.
3. Obično se koristi **JPanel** objekat kao platno za crtanje. Da bi na njemu crtali, treba da kreirate novu klasu koja proširuje **JPanel** i koja prekriva metod **paintComponent** da bi saopštila panelu crta grafiku.
4. Pozivanjem **super.paintComponent(g)** je neophodno radi osiguranja da prostor prikaza bude čist pre prikaza novog crteža. Korisnik može da zahteva da komponenta bude ponovo prikazana pozivom metoda **repaint()** definisanog u klasi **Component**. Pozivom metoda **repaint()** inicira se poziv i metoda **paintComponent** od strane JVM. Korisnik ne sme nikada da poziva **paintComponent** direktno. Zbog toga, data je zabrana PRISTUPA metoda **paintComponent** (protected).
5. Klasa **Component** ima metode **setBackground**, **setForeground** i **setFont**. Ovim se metodima postavljaju boje i fontovi za komponentu. Ako želite da ispišete nekoliko poruka na panelu sa različitim bojama i fontovima, možete upotrebiti **setColor** i **setFont** metode u klasi **Graphics** da bi podesili boju i font tekućeg crteža.
6. Klasa **FontMetrics** se koristi radi računanja tačne dužine i širine nekog teksta (stringa), što je potrebno pri merenju veličine teksta radi njegovog prikazivanja na željenom mestu.
7. Da bi se prikazala neka slika, mora da prvo kreirate ikonu slike. Onda upotrebite metod **getImage()** klase **ImageIcon** da bi dobili objekat **Image** koji predstavlja sliku i koji crta sliku upotrebom **drawImage** metoda u klasi **java.awt.Graphics**.