Graphical Java applications

Objectives:

- understanding and applying the basic graphics concepts;
- developing stand alone applications;

Implementation of stand-alone applications

Stand-alone Java graphical applications can be implemented using **AWT**, **Swing** or, more recently, **JavaFX** library packages.

A `Frame`-type AWT application is graphically represented by an independent window, which has the appearance dictated by the operating system on which JVM is installed.

Constructors:

```
Frame()
Frame(String title)
```

When a window is built, it has no visible size and is not set to appear on the screen either. The methods `setBounds()` or `setSize()` are used for setting the size and position on the screen and the method `setVisible()` with the parameter put on *true* for the created window to be visible.

When the application is finished, the allocated resources must be released using the method `dispose()`.

```
//window (frame) creation
Frame f = new Frame(”Aplicaţie de tip Frame”);

//positioning
f.setBounds(20, 20, 500, 300);

//making visible
f.setVisible(true);

//{...} working with the window

//finishing the application
f.dispose();
```

Graphics in Java

The methods and classes mentioned below are used.

- Class *Graphics* https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/java/awt/Graphics.html
- Class Graphics2D https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/java/awt/Graphics2D.html

Instances of the mentioned classes are used to gain access to the graphical context attached to a base container (like *Frame*), intermediate container (like *Panel*) or specific components (like *Canvas*).

Elements used in Java graphics

The fonts are managed using the classes *Font, FontMetrics* and *LineMetrics.*

The *Font* class has the following constructor:

```
Font(String name, int style, int size)
```

The colors are managed using the classes *Color* and *ColorSpace*. *RGB, HSB* approaches are available for managing colors and for converting them between the allowed color representations.

Colors can be used
- for drawing (`void setColor(Color color)`)
- as foreground attribute (`void setForeground(Color color)`)
- as background attribute (`void setBackground(Color color)`)

The main methods used in drawing are:

```
public void drawLine(int x0, int y0, int x1, int y1)
public void drawRect(int x, int y, int width, int height)
public void fillRect(int x, int y, int width, int height)
public void drawOval(int x, int y, int width, int height)
public void fillOval(int x, int y, int width, int height)
public void drawArc(int x, int y, int width, int height, int startDegrees, int arcDegrees)
public void fillArc(int x, int y, int width, int height, int startDegrees, int arcDegrees)
public void drawPolygon(int xs[], int ys[], int numPoints)
public void fillPolygon(int xs[], int ys[], int numPoints)
public void drawPolyline(int xs[], int ys[], int numPoints)
public void drawString(String s, int x, int y)
public void drawImage(Image im, int x, int y, ImageObserver observer)
```

Drawing in Java graphics applications is done using the paint (Graphics g) method, defined in the `java.awt.Component` class, having the signature:

```
public void paint(Graphics g)
```

`javax.swing.JComponent` inherits the AWT `Component` class and defines 3 other methods:

```
void paintComponent(Graphics g)
void paintBorder(Graphics g)
void paintChildren(Graphics g)
```

The method `paint()` is called automatically in the following situations:
- when exposing the component
- after de-iconification
- when bringing to the fore the window of an application

Redrawing a component can be controlled by calling the method `repaint()`. The signature of this method in the `Component` class is:

```
void repaint()
```

Obtaining the graphic context in which the drawing/writing is performed can be done with the method `getGraphics()` from the `Component` class, having as signature:

```
Graphics getGraphics()
```

Examples

Ex. 1

```
/* Aplicatia realizeaza desenarea in fereastra a 15 cercuri cu dimensiuni diferite, la
interval de 1 secunda. */
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.WindowListener;
import java.awt.event.WindowEvent;

class MyFrame extends Frame implements WindowListener{

	MyFrame(String title){
		super(title);

		setLayout(new FlowLayout());
		this.addWindowListener(this);

		setSize(300, 300);
		setVisible(true);
	}

	public void windowActivated(WindowEvent arg0) { }

	public void windowClosed(WindowEvent arg0) { }

	public void windowClosing(WindowEvent arg0) {
		System.exit(1);
	}

	public void windowDeactivated(WindowEvent arg0) { }

	public void windowDeiconified(WindowEvent arg0) { }

	public void windowIconified(WindowEvent arg0) { }

	public void windowOpened(WindowEvent arg0) { }

	@Override
	public void paint(Graphics g) {

		int a=150,b=150,c=10,d=10;

		g.setColor(Color.red);
		for(int i=0;i<15;i++){
			try{
				Thread.sleep(1000);
			}
			catch(InterruptedException ex){}

			g.drawOval(a, b, c, d);
			a-=10;
```

```
                        b-=10;
                        c+=8;
                        d+=8;
                  }
            }
}

class Test{
      public static void main(String...strings){
            MyFrame frame = new MyFrame("Circle Animation");
      }
}
```

Ex. 2

```
/* Aplicatia adauga in fereastra AWT o componenta de tip Canvas, in care deseneaza un patrat
si un cerc. Inchiderea ferestrei se realizeaza utilizand clasa WindowAdapter */

import java.awt.Canvas;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.Dimension;

class MyFrame extends Frame {
      MyCanvas canvas;
      MyFrame(String title){
            super(title);
            setLayout(new FlowLayout());
            canvas = new MyCanvas();
            add(canvas);

            this.addWindowListener(new ExitListener());
            setSize(300, 300);
            setVisible(true);
      }
}

class ExitListener extends WindowAdapter {//closing with adapter class
  @Override
  public void windowClosing(WindowEvent event) {
        System.exit(0);
      }
}


class MyCanvas extends Canvas{
      Graphics g;
      int radius = 20;
```

```
	MyCanvas(){
		this.setSize(new Dimension(200, 200));
		this.setBackground(new Color(255, 125, 0));
	}

	@Override
	public void paint(Graphics g) {
		g.setColor(new Color(255, 255, 255));
		int x, y;
		g.drawRect(10, 10, 50, 50);
		g.setColor(new Color(100, 100, 100));
		g.drawOval(10, 10, 50, 50);
	}
}

class Test{
	public static void main(String...strings){
		MyFrame frame = new MyFrame("Painting on a Canvas");
	}
}
```

Ex. 3

```
/* Afisare fonturi disponibile, in cadrul unei imagini create si adaugate intr-o componenta
de tip Canvas */
import java.awt.Canvas;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.Dimension;
import java.awt.image.BufferedImage;
import java.awt.Font;
import java.awt.GraphicsEnvironment;

class MyFrame extends Frame {
	MyCanvas canvas;

	MyFrame(String title){
		super(title);

		setLayout(new FlowLayout());
		canvas = new MyCanvas();
		add(canvas);

		this.addWindowListener(new ExitListener());

		setSize(500, 500);
		setVisible(true);
	}
}

class ExitListener extends WindowAdapter {//closing with adapter class
```

```
        @Override
        public void windowClosing(WindowEvent event) {
              System.exit(0);
            }
      }

class MyCanvas extends Canvas{

      Graphics g;
      BufferedImage bi = null;

      MyCanvas(){
            this.setSize(new Dimension(400, 400));
            this.setBackground(new Color(255, 125, 0));

            bi = new BufferedImage(300, 400, BufferedImage.TYPE_INT_ARGB);

            drawIntoImage();
      }

      public void drawIntoImage(){
            //desenarea in cadrul imaginii
            Graphics g = bi.getGraphics();

            GraphicsEnvironment ge = GraphicsEnvironment.getLocalGraphicsEnvironment();

            Font[] allFonts = ge.getAllFonts();
            for (int i = 0; i < allFonts.length; i++) {
                  Font f = allFonts[i].deriveFont(18.0f);
                  g.setFont(f);
                  g.drawString((i+1) + ". " + f.getFamily(), 10, 20*i+30);
            }
      }

      @Override
      public void paint(Graphics g) {
            g.drawImage(bi, 1, 1, this);
      }
}

class TestImageFont{
      public static void main(String...strings){
            MyFrame frame = new MyFrame("Painting on a Canvas");
      }
}
```

Ex. 4

```
// aplicatie de tip Frame, desenare in Panel (AWT), fara inchiderea ferestrei
import java.awt.Color;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Panel;
```

```
class Panel_Canvas extends Panel {
      private static final long serialVersionUID = 1L;

      public Panel_Canvas( ) {  }
      public void paint(Graphics g) {
            int width = getWidth();
            int height = getHeight();
            g.setColor(Color.magenta);
            g.drawOval(0, 0, width, height);
      }
}

public class AwtPanel{
      public static void main(String args[ ]) {
            Frame frame = new Frame("Oval Sample");
            frame.add(new JPanel_Canvas( ));
            frame.setSize(300, 200);
            frame.setVisible(true);
      }
}
```

Ex.5

```
//aplicatie de tip Frame, desenare in Panel (Swing)
import java.awt.Color;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class JPanel_Canvas extends JPanel {
       private static final long serialVersionUID = 1L;

       public JPanel_Canvas( ) {  }
       public void paintComponent(Graphics g) {
            int width = getWidth();
            int height = getHeight();
            g.setColor(Color.magenta);
            g.drawOval(0, 0, width, height);
       }

       public static void main(String args[ ]) {
            JFrame frame = new JFrame("Oval Sample");
            frame.add(new JPanel_Canvas( ));
            frame.setSize(300, 200);
            frame.setVisible(true);
            //frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       }
}
```

Ex.6

```
/* Meniu simplu AWT */

import java.awt.*;

public class MenuExample
{
      MenuExample( ){
            Frame f= new Frame("Menu and MenuItem Example");
            MenuBar mb=new MenuBar( );
            Menu menu=new Menu("Menu");
            Menu submenu=new Menu("Sub Menu");
            MenuItem i1=new MenuItem("Item 1");
            MenuItem i2=new MenuItem("Item 2");
            MenuItem i3=new MenuItem("Item 3");
            MenuItem i4=new MenuItem("Item 4");
            MenuItem i5=new MenuItem("Item 5");
            menu.add(i1); menu.add(i2); menu.add(i3);
            submenu.add(i4); submenu.add(i5);
            menu.add(submenu);
            mb.add(menu);
            f.setMenuBar(mb);
            f.setSize(400,400);
            f.setLayout(null);
            f.setVisible(true); }

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

Ex.7

```
/* Meniu simplu Swing */
import javax.swing.JFrame;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

public class MenuExample
{
      MenuExample( ){
            JFrame f= new JFrame("Menu and MenuItem Example");
            JMenuBar mb=new JMenuBar( );
            JMenu menu=new JMenu("Menu");
            JMenu submenu=new JMenu("Sub Menu");
            JMenuItem i1=new JMenuItem("Item 1");
            JMenuItem i2=new JMenuItem("Item 2");
            JMenuItem i3=new JMenuItem("Item 3");
            JMenuItem i4=new JMenuItem("Item 4");
            JMenuItem i5=new JMenuItem("Item 5");
            menu.add(i1); menu.add(i2); menu.add(i3);
            submenu.add(i4); submenu.add(i5);
            menu.add(submenu);
            mb.add(menu);
            f.setJMenuBar(mb);
            f.setSize(400,400);
            f.setLayout(null);
            f.setVisible(true); }
      public static void main(String args[ ]){
            new MenuExample( ); }
      }
```

Individual work

Complexity *

1. Write a Frame application that will display in a panel your name and group. The text will be blue and centered both on the horizontal and vertical, given the dimension of the panel.

2. Write a Frame application that will draw, inside a Canvas component, a circle tangent to smaller side of the canvas.

3. Write a Frame application that displays an image stored in a local file. Below the image display the filename with a predefined Font. The text is drawn using a color composed of 3 predefined components (red, green, blue).

Complexity **

4. Write a Frame application that displays a TV test card. The card will contain at least 10 levels of grey and the basic colors red, green, blue, yellow, cyan and magenta. The card will cover the entire surface of the component that displays it (Canvas, etc.).

5. Write a Frame application that draws a circle colored in all the possible colors. The starting color is red and the color transitions must be made as smooth as possible.

Complexity ***

6. Write a Frame application that draws a schematic car. Use different colors for different car parts (doors, wheels, etc.).

7. Implement a Frame application that fills a visual container with circles having randomly generated radii. The circles are tangent and have random colors.

8. Write a Java application that draws the trajectory of an object ”thrown” horizontally with a predefined velocity. The trajectory is limited by the visual container’s lower bound. If the trajectory intersects the left or right container sides, the object will bounce in the opposite direction. Display the object’s final vertical velocity.