KMBB COLLEGE OF ENGINEERING AND TECHNOLOGY

PAINT DRAWING IN JAVA

PRESENTED BY:-

IPSITA BISWAL

REGD NO.:-1901329031

Content:-

OVERVIEW:-

- JAVA is a general-purpose,concurrent,class-based,opensource,highlevel,generalpurpose,generalized programming language.

- It is intended to let application developers **"WRITE ONCE,RUN ANYWHERE",**meaning that code that runs on one platform does not need to be recompiled to run on another.
- Java applications are typically compiled to byte code (class file) that can run on any Java virtual machine (JVM) regardless of computer architecture. Java is, as of 2012, one of the most popular programming languages in use, particularly for client-server web applications, with a reported 10 million users [1][2]. Java was originally developed by James Gosling at Sun Microsystems (which has since merged into Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform.

OBJECT-ORIENTED PROGRAMMING LANGUAGE:-

- Java supports object-oriented programming techniques that are based on a hierarchy of classes and well-defined and cooperating objects [4].

- **Classes:** It is the collection of data member and member function . A class is a structure that defines the data and the methods to work on that data. When you write programs in Java, all program data is wrapped in a class, whether it is a class you write or a class you use from the Java API libraries. Classes in the Java API libraries define a set of objects that share a common structure and behavior.

- **Objects:** An instance is a synonym for object. A newly created instance has data members and methods as defined by the class for that instance.

- **Well-Defined Boundaries and Cooperation:** Class definitions must allow objects to cooperate during execution.
- **Inheritance and Polymorphism:** One object-oriented concept that helps objects work together is inheritance. Inheritance defines relationships among classes in an object-oriented language by help of extend keyword.

  Polymorhism divided in to three types for helps objects work together by help of implements.

The Basic GUI (graphical user interface) Application:-

- There are two basic types of GUI program in Java [5]: stand-alone applications and (online) applets. An applet is a program that runs in a rectangular area on a Web page. Applets are generally small programs, meant to do fairly simple things, although there is nothing to stop them from being very complex.
- **JFrame and JPanel:** In a Java GUI program, each GUI component in the interface is represented by an object in the program.
- A **JFrame** is an independent window that can, for example, act as the main window of an application. One of the most important things to understand is that a JFrame object comes with many of the behaviors of windows already programmed in.
- **JPanel** is another of the fundamental classes in Swing. The basic JPanel is, again, just a blank rectangle. There are two ways to make a useful JPanel : The first is to add other components to the panel; the second is to draw something in the panel.

Graphics and Painting:-

- The physical structure of a GUI is built of components. The term component refers to a visual element in a GUI, including buttons, menus, text-input boxes, scroll bars, check boxes, and so on. In Java, GUI components are represented by objects belonging to subclasses of the class java.awt.Component. In a graphical system, a windowing toolkit is usually responsible for providing a framework to make it relatively painless for a graphical user interface (GUI) to render the right bits to the screen at the right time. Both the AWT (abstract windowing toolkit) and Swing provide such a framework. But the APIs that implement it are not well understood by some developers -- a problem that has led to programs not performing as well as they could.
- In order to use graphics in Java programs, there are a number of libraries we need to import. For the sake of what will be covered in these notes, you need the following statements:
- **In order to use graphics in Java programs, there are a number of libraries we need to import. For the sake of what will be covered in these notes, you need the following statements:**----------import javax.swing.JFrame;/ import javax.swing.Jpanel/import java.awt.Graphics;/import java.awt.geom.*/import java.awt.Color;/import javax.swing.JLabel;/
- import javax.swing.JLabel;/ import java.awt.event.WindowAdapter;/ import java.awt.event.WindowEvent;/import javax.swing.ImageIcon;/import java.awt.BorderLayout;/ import java.awt.event.*;/import java.awt.image.BufferedImage; import java.awt.image.MemoryImageSource; import java.awt.image.PixelGrabber;/import java.io.File;/import java.io.IOException;/import javax.imageio.*;

Coordinates

The Java 2D™ API maintains two coordinate spaces:

- User space – The space in which graphics primitives are specified
- Device space – The coordinate system of an output device such as a screen, window, or a printer.
- The screen of a computer is a grid of little squares called pixels. The color of each pixel can be set individually, and drawing on the screen just means setting the colors of individual pixels.

Colors

RGB: Of course, there are many more colors we might want. We can specify other colors using the **RGB model**, which specifies a color with a red value, a green value, and a blue value (each called **channels**). For example, the red value is how much red we want in our color. In different graphics system, the individual color values might be represented in different ways, but in Java, we use 8 bits to each color value. This gives us 256 (2^8) choices for each of the R, G, and B values, and choices range from 0 to 255.

RED VALUE:-0(NO RED)-255(ALL RED)

GREEN VALUE:-0(NO GREEN)-255(ALL GREEN)

BLUE VALUE:-0(NO BLUE)-255(ALL BLUE)

SHAPES:

- The Graphics class includes a large number of instance methods for drawing various shapes, such as lines, rectangles, and ovals. The shapes are specified using the (x,y) coordinate system.

- **drawString(String str, int x, int y):** Draws the text given by the string str. The string is drawn using the current color and font of the graphics context. x specifies the position of the left end of the string. y is the y-coordinate of the baseline of the string. The baseline is a horizontal line on which the characters rest. Some parts of the characters, such as the tail on a y or g, extend below the baseline.

- **drawLine(int x1, int y1, int x2, int y2):** Draws a line from the point (x1,y1) to the point (x2,y2).

- **drawRect(int x, int y, int width, int height):** Draws the outline of a rectangle. The upper left corner is at (x,y), and the width and height of the rectangle are as specified. If width equals height, then the rectangle is a square. If the width or the height is negative, then nothing is drawn.

- **drawOval(int x, int y, int width, int height)** Draws the outline of an oval. The oval is one that just fits inside the rectangle specified by x, y, width, and height. If width equals height, the oval is a circle.

- **drawRoundRect(int x, int y, int width, int height, int xdiam, int ydiam):** Draws the outline of a rectangle with rounded corners. The basic rectangle is specified by x, y, width, and height, but the corners are rounded. The degree of rounding is given by xdiam and ydiam. The corners are arcs of an ellipse with horizontal diameter xdiam and vertical diameter ydiam. A typical value for xdiam and ydiam is 16, but the value used should really depend on how big the rectangle is. Etc.

GRAPHICS 2D:-

- **public void paintComponent(Graphics g) {**
- **super.paintComponent(g);**
- **Graphics2D g2;**
- **g2 = (Graphics2D)g;**
- **. // Draw on the component using g2.**
- **}**

- In our example:
  - **public void draw(Graphics2D ga){**
    - **ga.setColor(color);**
    - **if (!IsFillColor)**
      - **ga.drawArc(getX1(), getY1(), getwidth(), getheight(), 0, 360); else**
      - **ga.fillArc(getX1(), getY1(), getwidth(), getheight(), 0, 360);**
      - **}**

- . In AWT, there are two kinds of painting operations: ***system-triggered painting***, and ***application-triggered painting.***
- **System-triggered Painting:** In a system-triggered painting operation, the system requests a component to render its contents, usually for one of the following reasons.

  - The component is first made visible on the screen & The component is resized.
  - **App-triggered Painting:** In an application-triggered painting operation, the component decides it needs to update its contents because its internal state has changed.

- **Here is a simple example of a paint callback which renders a filled circle in the bounds of a component:**

```
    public void paint(Graphics g) {
// Dynamically calculate
size information Dimension size = getSize();
// diameter
    int d = Math.min(size.width, size.height);
    int x = (size.width - d)/2;
    int y = (size.height - d)/2;
// draw circle (color already set to foreground) g.fillOval(x, y, d, d); g.setColor(Color.black); g.drawOval(x, y,
d, d);}
```

example of a mouse listener that uses repaint() to trigger updates on a theoretical button component when the mouse is pressed and released

```
MouseListener l = new MouseAdapter() { public void mousePressed(MouseEvent e) {
MyButton b =      (MyButton)e.getSource();
b.setSelected(true);
b.repaint(); }
public void mouseReleased(MouseEvent e) {
MyButton b = (MyButton)e.getSource();
b.setSelected(false);
b.repaint(); } };
```

paint() vs. update(): Why do we make a distinction between "system-triggered" and. "app-triggered" painting? Because AWT treats each of these cases slightly differently for heavyweight

Design and Development our project

- In this section we describe how our project designed and implementation. This is a simple Painter project using the JAVA language. The program uses 10 classes to build the entire structure. Please compile using the JAVA SDK, and run the Main_DrawPaintProject.java as the main program to call the main form interface InterfaceForm.java.
- **Program Ability (Objectives):**
  - Draw Circle, Line, Rectangle, Square, and Oval using FreeHand (move the mouse using your hand to draw any shape and specify the coordinate in JPanel).
  - Undo and Redo process.
- Clear JPanel
- Set Background Color & set Foreground Color.
- Save paint (Panel) to file (*. JPG; *. GIF; *.*)
  - Open paint from file
  - The system enables you to use FreeHand to draw (move the mouse using your hand to draw any shape and specify the coordinate in JPanel) as an easy way to draw the integrated paint, for example, a car , a street , a football stadium , traffic signals and others.

System Framework

- **Interface Model:** Which operation do you want to do that includes as the follows:
  - 1- The **input system** is user chosen operation what is the process that wants to be painted by the user, whether draw a shape on the panel or operations on the drawing located.
  - 2- **Buttons paint**: What are the available operations that you can to do such as Line, Circle, Square, Rectangle, Oval, and so on.
- 3- **Menu Color:** Button and radio group are used to setColor for shape or set Background for the panel

- **Operation Model:** Events are executed requested by the user and show the result of the required on panel or frame.
  1. The events on the buttons or panel such as mouseClicked, MousePressed, MouseReleased, MouseMoved, MouseDragged, and FocusGained, when the user released events the program call the subroutine to achieve operation required.

  1. After completing execute the subroutine show the result on panel.

- **Main_DrawPaintProject.java (*Main Program*)/InterfaceForm.java (Main program interface)/Shape.java/Command.java**

- **Preview (System Interface)**

- Figure 4 shows the main interface of our project and what abilities available (tools) are in our program for drawing, coloring, and save and open image file. Also explain how you can draw an integrated paint from several geometric shapes and different colors, for example, a car, a street.

- <<Interface>> Shape

- public interface Shape {

- public void draw(Graphics2D g);

- }

- <<Interface>> Command

- public interface Command {

- public final static int LINE = 2;

- public final static int CIRCLE = 4;

- public final static int RECTANGLE = 8;

- public final static int SQUARE=12;

- public final static int Oval=16;}

CONCLUSION:-

- It is an introduction to Java and how Java is used to build graphics and what are the tools that can be used to develop graphics and drawing required shapes.
- **References**

1. "Programming Language Popularity". 2009. Retrieved 2009-01-16
2. "TIOBE Programming Community Index". 2009. Retrieved 2009-05-06
3. **http://www.csci.csusb.edu/dick/samples/java.html, 2011,** Thu Aug 25 21:04:36 PDT 2011
4. Monica Pawlan , Essentials of the Java Programming Language A Hands-On Guide
5. David J. Eck , Introduction to Programming Using Java , Version 6.0, June 2011 (Version 6.0.2, with minor corrections, May 2013)
6. Mads Rosendahl, Introduction to graphics programming in Java, February 13, 2009
7. 2D Graphics (The Java™ Tutorials) - Oracle Documentation,
   - **http://docs.oracle.com/javase/tutorial/2d/overview/index.html, Copyright** © 1995, 2013 Oracle
8. **http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html** , 2013
9. **http://docs.oracle.com/javase/7/docs/api/java/awt/Graphics2D.html, 2013**
10. Oracle Technology, http://www.oracle.com/technetwork/java/painting-
    - 140037.html#callback


THANK YOU