Q.1 Write a java program that bounces a blue ball inside a JPanel. The ball should begin moving with a mousePressed event. When the ball hits the edge of the JPanel, it should bounce off the edge and continue in the opposite direction. The ball should be updated using a Runnable.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class BouncingBall extends JPanel implements MouseListener, Runnable {

private int ballX = 100, ballY = 100;

private int ballDiameter = 30;

private int ballVelocityX = 3, ballVelocityY = 3;

private boolean ballMoving = false;

public BouncingBall() {

addMouseListener(this);

setPreferredSize(new Dimension(400, 400));

}

@Override

public void mousePressed(MouseEvent e) {

ballMoving = true;

new Thread(this).start();

}

@Override

protected void paintComponent(Graphics g) {

super.paintComponent(g);

g.setColor(Color.BLUE);

g.fillOval(ballX, ballY, ballDiameter, ballDiameter);

}

@Override

public void run() {

while (ballMoving) {

ballX += ballVelocityX;

ballY += ballVelocityY;

if (ballX <= 0 || ballX >= getWidth() - ballDiameter) {

ballVelocityX = -ballVelocityX;

}

if (ballY <= 0 || ballY >= getHeight() - ballDiameter) {

ballVelocityY = -ballVelocityY;

}

repaint();

try {

Thread.sleep(10);

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

@Override

public void mouseReleased(MouseEvent e) {}

@Override

public void mouseEntered(MouseEvent e) {}

@Override

public void mouseExited(MouseEvent e) {}

@Override

public void mouseClicked(MouseEvent e) {}

public static void main(String[] args) {

JFrame frame = new JFrame("Bouncing Ball");

BouncingBall bouncingBall = new BouncingBall(); frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

frame.add(bouncingBall);

frame.pack();

frame.setVisible(true);

}

}

Q. 2 Create an application of Stopwatch using the concept of Multithreading.

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class StopwatchApp extends JFrame {

private JLabel timeLabel;

private JButton startButton, stopButton, resetButton;

private StopwatchThread stopwatchThread;

private boolean running = false;

private int seconds = 0;

private int minutes = 0;

private int hours = 0;

public StopwatchApp() {

setTitle("Stopwatch");

setSize(300, 200); setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setLocationRelativeTo(null);

timeLabel = new JLabel("00:00:00", SwingConstants.CENTER);

timeLabel.setFont(new Font("Arial", Font.PLAIN, 40));

startButton = new JButton("Start");

stopButton = new JButton("Stop");

resetButton = new JButton("Reset");

startButton.addActionListener(e -> startStopwatch());

stopButton.addActionListener(e -> stopStopwatch());

resetButton.addActionListener(e -> resetStopwatch());

JPanel panel = new JPanel();

panel.setLayout(new GridLayout(2, 2));

panel.add(startButton);

panel.add(stopButton);

panel.add(resetButton);

add(timeLabel, BorderLayout.CENTER);

add(panel, BorderLayout.SOUTH);

}

private void startStopwatch() {

if (!running) {

stopwatchThread = new StopwatchThread();

new Thread(stopwatchThread).start();

running = true;

startButton.setEnabled(false);

}

}

private void stopStopwatch() {

running = false;

startButton.setEnabled(true);

}

private void resetStopwatch() {

running = false;

seconds = 0;

minutes = 0;

hours = 0; timeLabel.setText(String.format("%02d:%02d:%02d", hours, minutes, seconds));

startButton.setEnabled(true);

}

private class StopwatchThread implements Runnable {

@Override

public void run() {

while (running) {

try {

Thread.sleep(1000

seconds++;

if (seconds == 60) {

seconds = 0;

minutes++;

}

if (minutes == 60) {

minutes = 0;

hours++;

}

timeLabel.setText(String.format("%02d:%02d:%02d", hours, minutes, seconds));

} catch (InterruptedException e) {

e.printStackTrace();

}

}

}

}

public static void main(String[] args) {

SwingUtilities.invokeLater(() -> {

StopwatchApp app = new StopwatchApp();

app.setVisible(true);

});

}

}