**Abdul Basit**

**193227**

**BSCS-6C**

**TASK#1**

**CODE**

public class Truck

{

public static void main(String[] args)

{

DrawingPanel panel = new DrawingPanel(650, 650);

panel.setBackground(Color.WHITE);

Graphics g = panel.getGraphics();

redraw(g,10,30,20);

redraw(g,100,45,40);

redraw(g,10,300,2);

redraw(g,100,300,61);

redraw(g,25,300,3);

redraw(g,45,300,4);

redraw(g,10,320,5);

}

static void redraw(Graphics g,int x,int y,int size)

{

// draw the body

g.setColor(Color.BLACK);

g.fillRect(x, y, size\*5, size/2\*5); // x, y, width, ht

// draw the wheels

g.setColor(Color.RED);

g.fillOval(x+size\*5/10, y+size\*20/10, size, size);

g.fillOval(x+size\*35/10, y+size\*20/10, size, size);

// draw the window

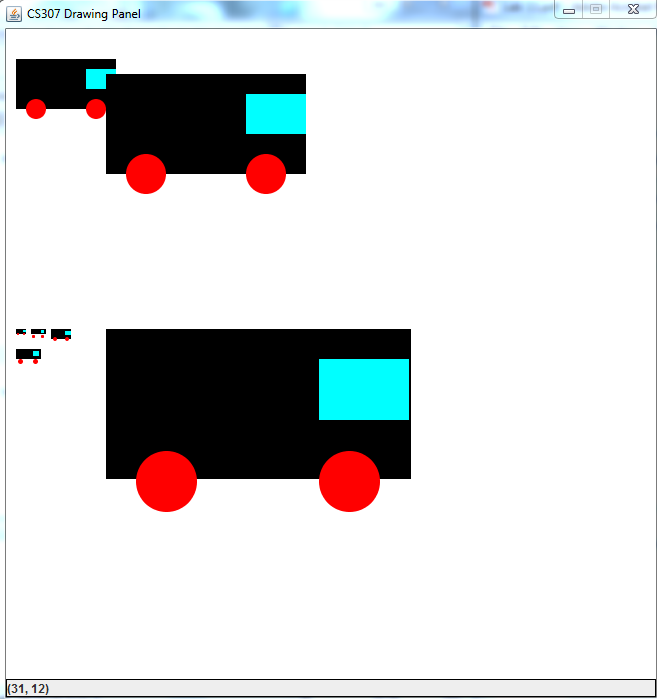
g.setColor(Color.CYAN);

g.fillRect(x+size\*35/10, y+size\*5/10, size/2\*3, size);

}

}

**OUTPUT**



**TASK#2**

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.Graphics;

import java.awt.Graphics2D;

import java.awt.Point;

import java.awt.RenderingHints;

import java.util.Calendar;

import javax.swing.JFrame;

import javax.swing.JPanel;

class Clock extends JFrame {

private static final long serialVersionUID = 1L;

private static final Color BACKGROUND\_COLOR = new Color(132, 112, 255);

public Clock() {

ClockPanel container = new ClockPanel();

add(container, BorderLayout.CENTER);

setBackground(BACKGROUND\_COLOR);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

setResizable(false);

pack();

setVisible(true);

}

public static void main(String[] args) {

new Clock();

}

}

class ClockPanel extends JPanel implements Runnable {

private static final long serialVersionUID = 1L;

Thread t = new Thread(this);

/\*\* The coordinates used to paint the clock hands. \*/

int xSec, ySec, xMin, yMin, xHour, yHour;

/\*\* The size of the clock. \*/

private final int HORIZONTAL\_SIZE = 500;

private final int VERTICAL\_SIZE = 500;

/\*\* The length of the clock hands relative to the clock size. \*/

private final int secondHandLength = HORIZONTAL\_SIZE / 2 - 50;

private final int minuteHandLength = HORIZONTAL\_SIZE / 2 - 70;

private final int hourHandLength = HORIZONTAL\_SIZE / 2 - 100;

/\*\* The distance of the dots from the origin (center of the clock). \*/

private final int DISTANCE\_DOT\_FROM\_ORIGIN = HORIZONTAL\_SIZE / 2 - 40;

private final int DIAMETER\_BIG\_DOT = 8;

private final int DIAMETER\_SMALL\_DOT = 4;

private final static Color GREY\_COLOR = new Color(160,160,160);

public ClockPanel() {

setMinimumSize(new Dimension(HORIZONTAL\_SIZE, VERTICAL\_SIZE));

setMaximumSize(new Dimension(HORIZONTAL\_SIZE, VERTICAL\_SIZE));

setPreferredSize(new Dimension(HORIZONTAL\_SIZE, VERTICAL\_SIZE));

setLayout(null);

t.start();

}

/\*\*

\* At each iteration we recalculate the coordinates of the clock hands,

\* and repaint everything.

\*/

public void run(){

while(true){

try{

//using an getInstance() method that returns the current time object

Calendar instant = Calendar.getInstance();

/\* TODO: Get Seconds from the Calender Class; Replace zero with your code \*/

int currentSecond = instant.get(Calendar.SECOND);

/\* TODO: Get Minutes from the Calender Class; Replace zero with your code \*/

int currentMinute = instant.get(Calendar.MINUTE);

/\* TODO: Get Hours from the Calender Class; Replace zero with your code \*/

int currentHour = instant.get(Calendar.HOUR);

/\*TODO: Calculate the X Coordinate of Seconds Handle using minToLocation method \*/

xSec = (int) minToLocation(currentSecond,secondHandLength).getX();

/\*TODO: Calculate the Y Coordinate of Seconds Handle using minToLocation method \*/

ySec = (int) minToLocation(currentSecond,secondHandLength).getY();

/\*TODO: Calculate the X Coordinate of Minutes Handle using minToLocation method \*/

xMin = (int) minToLocation(currentMinute,secondHandLength).getX();

/\*TODO: Calculate the Y Coordinate of Minutes Handle using minToLocation method \*/

yMin = (int) minToLocation(currentMinute,secondHandLength).getY();

/\*TODO: Calculate the X Coordinate of Hours Handle using minToLocation method \*/

xHour = (int)minToLocation(currentHour \* 5 + getRelativeHour(currentMinute), hourHandLength).getX();

/\*TODO: Calculate the Y Coordinate of Hours Handle using minToLocation method \*/

yHour = (int)minToLocation(currentHour \* 5 + getRelativeHour(currentMinute), hourHandLength).getY();

repaint();

Thread.sleep(500);

} catch(InterruptedException ie){

ie.printStackTrace();

}

}

}

/\*\*

\* Returns how much the hour hand should be ahead

\* according to the minutes value.

\* 04:00, return 0.

\* 04:12, return 1, so that we move the hour handle ahead of one dot.

\* @param min The current minute.

\* @return The relative offset to add to the hour hand.

\*/

private int getRelativeHour(int min) {

return min / 12;

}

protected void paintComponent(Graphics g){

Graphics2D g2 = (Graphics2D)g;

g2.setRenderingHint(RenderingHints.KEY\_ANTIALIASING,

RenderingHints.VALUE\_ANTIALIAS\_ON);

g2.clearRect(0, 0, getWidth(), getHeight());

// Draw the dots

g2.setColor(GREY\_COLOR);

for (int i = 0; i < 60; i++) {

Point dotCoordinates = minToLocation(i, DISTANCE\_DOT\_FROM\_ORIGIN);

g2.setColor((i <= Calendar.getInstance().get(Calendar.SECOND)) ? Color.white : GREY\_COLOR);

if (i % 5 == 0) {

// big dot

/\*TODO: Draw the big dots using the fillOval method \*/

//Using the x and y cordinates of the point

g2.fillOval(dotCoordinates.x , dotCoordinates.y , DIAMETER\_BIG\_DOT, DIAMETER\_BIG\_DOT);

} else {

// small dot

/\*TODO: Draw the small dots using the fillOval method \*/

g2.fillOval(dotCoordinates.x , dotCoordinates.y , DIAMETER\_SMALL\_DOT, DIAMETER\_SMALL\_DOT);

}

}

//Changing the color of each Hand so that it is visible

// Draw the clock hands

g2.setColor(Color.GREEN);

g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xSec, ySec);

/\*TODO: Draw the minute hand here, just like the seconds hand \*/

g2.setColor(Color.GREEN);

g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xMin, yMin);

/\*TODO: Draw the hour hand here, just like the seconds hand \*/

g2.setColor(Color.GREEN);

g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHour, yHour);

}

/\*\*

\* Converts current second/minute/hour to x and y coordinates.

\* @param min The current minute

\* @param radius The radius length

\* @return the coordinates point

\*/

private Point minToLocation(int timeStep, int radius) {

double t = 2 \* Math.PI \* (timeStep-15) / 60;

int x = (int)(HORIZONTAL\_SIZE / 2 + radius \* Math.cos(t));

int y = (int)(VERTICAL\_SIZE / 2 + radius \* Math.sin(t));

return new Point(x, y);

}

}

**OUTPUT**

