**Name: Abdul Ghaffar Kalhoro**

**Reg: no: 194699**

**Class: BSCS-6C (2nd semester)**

**Lab 11: Graphics & Java2D**

**Object Oriented Programming (OOP)**

**Task#1 (Source code)**

**Truck.java**

**package** Lab10;

**import** java.awt.Color;

**import** java.awt.Graphics;

**public** **class** Truck

{

**public** **static** **void** main(String[] args)

{

DrawingPanel panel = **new** DrawingPanel(500, 500);

panel.setBackground(Color.***WHITE***);

Graphics g = panel.getGraphics();

//calling the method drawTruck for drawing truck of different sizes

*drawTruck*(g, 10, 30, 20);

*drawTruck*(g, 100, 45, 40);

*drawTruck*(g, 10, 300, 2);

*drawTruck*(g, 100, 300, 61);

*drawTruck*(g, 25, 300, 3);

*drawTruck*(g, 45, 300, 4);

*drawTruck*(g, 10, 320, 5);

}//end main

**private** **static** **void** drawTruck(Graphics g, **int** Uleftx, **int** Ulefty, **int** size){

//to set the initial width multiply 5 to the total size

**int** constant = size \* 5;

//Drawing the larger rectangle

g.setColor(Color.***BLACK***);

g.fillRect(Uleftx, Ulefty, constant, (**int**)(constant\*0.5)); // x, y, //width, height

// drawing wheels

g.setColor(Color.***RED***);

g.fillOval(Uleftx+(**int**)(size\*0.5), Ulefty + (**int**)((constant\*0.5) - (constant\*0.1)), (**int**)(constant\*0.2),(**int**)(constant\*0.2) );

g.fillOval(Uleftx + (**int**)((size \* 3) + size\*0.5) , Ulefty + (**int**)((constant\*0.5) - (constant\*0.1)), (**int**)(constant\*0.2), (**int**)(constant\*0.2));

// draw the window

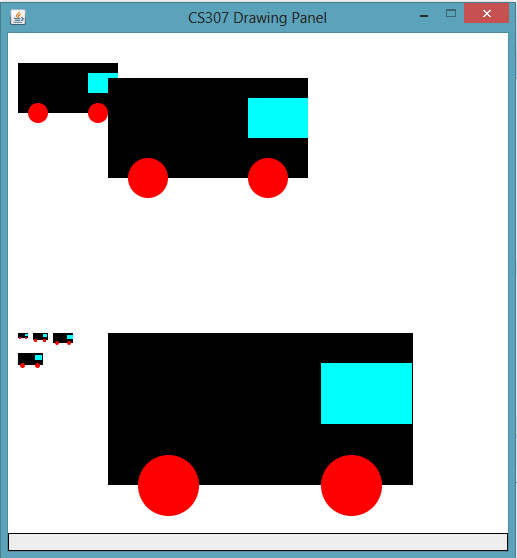
g.setColor(Color.***CYAN***);

g.fillRect(Uleftx+(**int**)((size \* 3) + size\*0.5), Ulefty+(**int**)(size\*0.5), (**int**)(constant\*0.3), (**int**)(constant\*0.2));

}//end method drawTruck

}//end Class Truck

**OUTPUT**

****

**Task#2**

**Clock.java**

**package** Lab11;

/\*\*

\* Paints an analog clock synchronized with the system clock.

\*/

**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(24, 116, 205);

**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** xHandSec, yHandSec, xHandMin, yHandMin, xHandHour, yHandHour;

/\*\* 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**{

Calendar piece = Calendar.*getInstance*();

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

**int** presentSecond = piece.get(Calendar.***SECOND***);

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

**int** presentMinute = piece.get(Calendar.***MINUTE***);

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

**int** presentHour = piece.get(Calendar.***HOUR***);

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

xHandSec = (**int**) minToLocation(presentSecond,secondHandLength).getX();

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

yHandSec = (**int**) minToLocation(presentSecond,secondHandLength).getY();

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

xHandMin = (**int**) minToLocation(presentMinute,secondHandLength).getX();

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

yHandMin = (**int**) minToLocation(presentMinute,secondHandLength).getY();

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

xHandHour = (**int**)minToLocation(presentHour \* 5 + getRelativeHour(presentMinute), hourHandLength).getX();

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

yHandHour = (**int**)minToLocation(presentHour \* 5 + getRelativeHour(presentMinute), 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.***RED***);

g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHandSec, yHandSec);

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

g2.setColor(Color.***GREEN***);

g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHandMin, yHandMin);

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

g2.setColor(Color.***BLACK***);

g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHandHour, yHandHour);

}

/\*\*

\* 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

