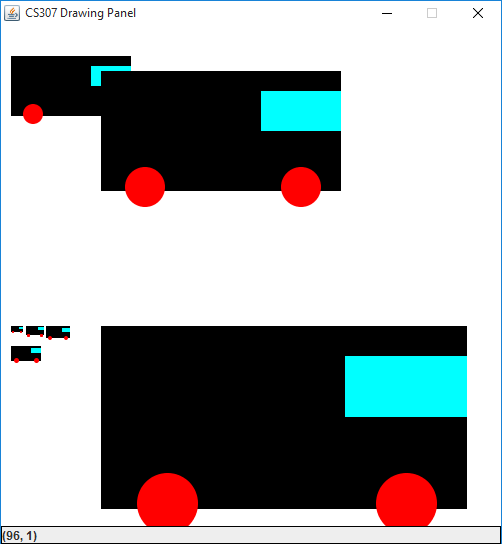
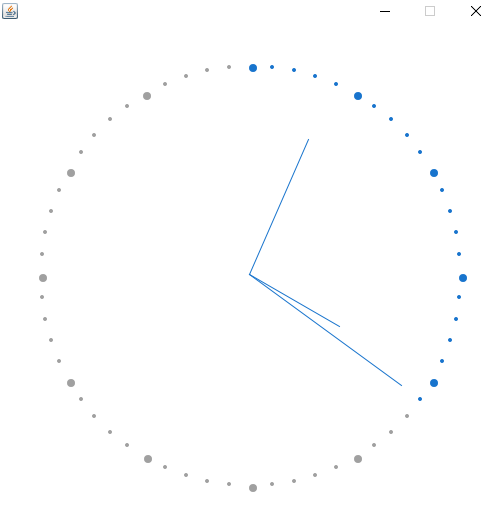
# Task 1



public static void drawTruck(Graphics g, int x, int y, int size) {  
 // recall the x and y indicate the upper left  
 // corner of the rectangle or oval bounding box  
 // draw the body  
 int tyreSize = size;  
 int height = tyreSize \* 3;  
 int width = tyreSize \* 6;  
 g.setColor(Color.BLACK);  
 g.fillRect(x, y, width, height); // x, y, width, ht  
  
 // draw the wheels  
 g.setColor(Color.RED);  
 g.fillOval(x + width / 10 , (y + height) - height / 5, tyreSize, tyreSize);  
 g.fillOval(x + width - tyreSize -tyreSize/2, (y + height) - height / 5, tyreSize, tyreSize);  
  
 // draw the window  
 g.setColor(Color.CYAN);  
 g.fillRect(x + width - tyreSize \* 2, y + tyreSize/2, tyreSize \* 2, tyreSize);  
 }

# Task 2



package jan.khan;  
/\*\*  
 \* 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);  
 }  
  
   
}  
  
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{  
 /\* **TODO:** Get Seconds from the Calender Class; Replace zero with your code \*/  
 Calendar cal = Calendar.getInstance();  
   
 int currentSecond = cal.getTime().getSeconds();  
 /\* **TODO:** Get Minutes from the Calender Class; Replace zero with your code \*/  
 int currentMinute = cal.getTime().getMinutes();  
 /\* **TODO:** Get Hours from the Calender Class; Replace zero with your code \*/  
 int currentHour = cal.getTime().getHours();  
  
   
 Point psec = minToLocation(currentSecond,this.DISTANCE\_DOT\_FROM\_ORIGIN\*9/10);  
 Point pmin = minToLocation(currentMinute,this.DISTANCE\_DOT\_FROM\_ORIGIN\*7/10);  
 Point phour = minToLocation(currentHour\*5,this.DISTANCE\_DOT\_FROM\_ORIGIN\*1/2);  
   
 /\***TODO:** Calculate the X Coordinate of Seconds Handle using minToLocation method \*/  
 xHandSec = psec.x;  
 /\***TODO:** Calculate the Y Coordinate of Seconds Handle using minToLocation method \*/  
 yHandSec = psec.y;  
 /\***TODO:** Calculate the X Coordinate of Minutes Handle using minToLocation method \*/  
 xHandMin = pmin.x;  
 /\***TODO:** Calculate the Y Coordinate of Minutes Handle using minToLocation method \*/  
 yHandMin = pmin.y;  
 /\***TODO:** Calculate the X Coordinate of Hours Handle using minToLocation method \*/  
 xHandHour = phour.x;  
 /\***TODO:** Calculate the Y Coordinate of Hours Handle using minToLocation method \*/  
 yHandHour = phour.y;  
  
 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  
   
 g.fillOval(dotCoordinates.x,dotCoordinates.y,this.DIAMETER\_BIG\_DOT,this.DIAMETER\_BIG\_DOT);  
 /\***TODO:** Draw the big dots using the fillOval method \*/  
 } else {  
 // small dot  
 g.fillOval(dotCoordinates.x,dotCoordinates.y,this.DIAMETER\_SMALL\_DOT,this.DIAMETER\_SMALL\_DOT);  
 /\***TODO:** Draw the small dots using the fillOval method \*/  
 }  
 }  
  
 // Draw the clock hands  
 g2.setColor(new Color(24, 116, 205));  
 g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHandSec, yHandSec);  
 g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHandMin, yHandMin);  
 g2.drawLine(HORIZONTAL\_SIZE / 2, VERTICAL\_SIZE / 2, xHandHour, yHandHour);  
  
 /\***TODO:** Draw the minute hand here, just like the seconds hand \*/  
 /\***TODO:** Draw the hour hand here, just like the seconds hand \*/  
 }  
  
 /\*\*  
 \* 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);  
 }