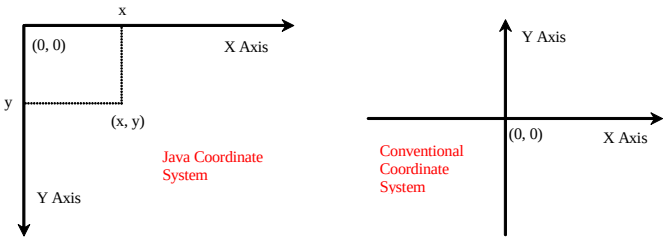
แบบฝึกหัดปฏิบัติการคาบที่ 10: Graphics

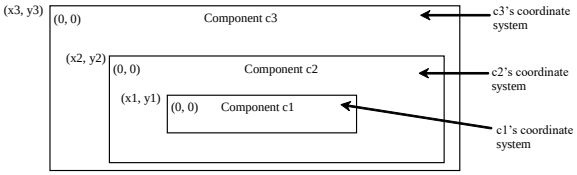
คำสั่ง

1.ให้ศึกษาหลักการต่อไปนี้

1.1 **ระบบ coordinate ของจาวา** ระบบ Coordinate ที่ใช้ในการอ้างอิงตำแหน่งของ object ที่แสดงใน window หนึ่งๆ สามารถแสดงได้ดังรูปต่อไปนี้



คอมโพเน้นต์แต่ละตัวจะมีระบบพิกัดอ้างอิงกับตัวคอมโพเน้นต์เองและจะมีระบบพิกัดที่อ้างอิงกับตัวคอมโพเน้นต์อื่น ดังแสดงตัวอย่างด้านล่าง



คลาส Graphics เราสามารถเขียนข้อความ วาดเส้นตรง วาดสี่เหลี่ยม วาดสามเหลี่ยม โพลีกอนโดยการใช้เมธอดต่าง ๆ ที่อยู่ภายในคลาส Graphics เมธอดต่าง ๆ ที่จะสามารถเลือกใช้ได้แสดงในคลาสไดอะแกรมต่อไปนี้

java.awt.Graphics	
+setColor(color: Color): void	Sets a new color for subsequent drawings.
+setFont(font: Font): void	Sets a new font for subsequent drwings.
+drawString(s: String, x: int, y: int): void	Draws a string starting at point (x, y).
+drawLine(x1: int, y1: int, x2: int, y2: int): void	Draws a line from (x1, y1) to (x2, y2).
+drawRect(x: int, y: int, w: int, h: int): void	Draws a rectangle with specified upper-left corner point at (x, y) and width w and height h.
+fillRect(x: int, y: int, w: int, h: int): void	Draws a filled rectangle with specified upper-left corner point at (x, y) and width w and height h.
+drawRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a round-cornered rectangle with specified arc width aw and arc height ah.
+fillRoundRect(x: int, y: int, w: int, h: int, aw: int, ah: int): void	Draws a filled round-cornered rectangle with specified arc width aw and arc height ah.
+draw3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a 3-D rectangle raised above the surface or sunk into the surface.
+fill3DRect(x: int, y: int, w: int, h: int, raised: boolean): void	Draws a filled 3-D rectangle raised above the surface or sunk into the surface.
+drawOval(x: int, y: int, w: int, h: int): void	Draws an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillOval(x: int, y: int, w: int, h: int): void	Draws a filled oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws an arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+fillArc(x: int, y: int, w: int, h: int, startAngle: int, arcAngle: int): void	Draws a filled arc conceived as part of an oval bounded by the rectangle specified by the parameters x, y, w, and h.
+drawPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a closed polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
+fillPolygon(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a filled polygon defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.
+drawPolygon(g: Polygon): void	Draws a closed polygon defined by a Polygon object.
+fillPolygon(g: Polygon): void	Draws a filled polygon defined by a Polygon object.
+drawPolyline(xPoints: int[], yPoints: int[], nPoints: int): void	Draws a polyline defined by arrays of x and y coordinates. Each pair of (x[i], y[i]) coordinates is a point.

เมธอด paintComponent เมื่อต้องการวาดรูปวาดต่าง ๆ บนคอมโพเน้นต์ จำเป็นต้องกำหนดให้คลาสที่จะใช้วาดภาพสืบทอดคลาส JPanel และโอเวอร์ไรด์เมธอด paintComponent เพื่อแสดงรายละเอียดถึงสิ่งที่จำเป็นต้องการวาด

ตัวอย่าง การโอเวอร์ไรด์เมธอด paintComponent

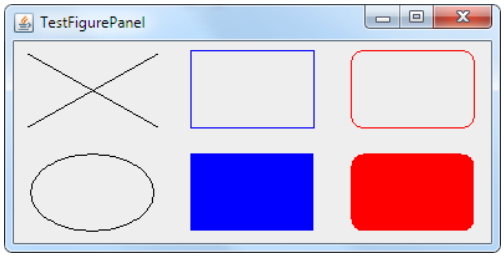
```
import javax.swing.*;
import java.awt.Graphics;

public class TestPaintComponent extends JFrame {
  public TestPaintComponent() {
    add(new NewPanel());
  }
  public static void main(String[] args) {
    TestPaintComponent frame = new TestPaintComponent();
    frame.setTitle("TestPaintComponent");
    frame.setSize(200, 100);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setVisible(true);
  }
}
class NewPanel extends JPanel {
  @Override
  protected void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawLine(0, 0, 50, 50);
    g.drawString("Banner", 0, 40);
  }
}
```

Drawing Lines, Rectangles and Ovals เมธอดที่ใช้ในการวาดเส้น สี่เหลี่ยม และวงกลม มีดังนี้

Method	Description
public void drawLine(int x1, int y1, int x2, int y2)	วาดเส้นจากจุด (x1, y1) ไปถึงจุด (x2, y2)
public void drawRect(int x, int y, int width, int height)	วาดสี่เหลี่ยม จุด (x,y) คือจุดมุมบนซ้ายของสี่เหลี่ยม
public void fillRect(int x, int y, int width, int height)	วาดสี่เหลี่ยมที่มีการระบายสีภายในสี่เหลี่ยม
public void clearRect(int x, int y, int width, int height)	วาด สี่เหลี่ยมและระบายสีภายในด้วยสีbackground
public void drawRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	วาดสี่เหลี่ยมมุมมน
public void fillRoundRect(int x, int y, int width, int height, int arcWidth, int arcHeight)	วาดสี่เหลี่ยมมุมมนและระบายสีภายในด้วย
public void draw3DRect(int x, int y, int width, int height, boolean b)	วาดสี่เหลี่ยมในระบบ 3 มิติ หาก b มีค่าเป็น true สี่เหลี่ยมจะนูนขึ้น แต่หาก b มีค่าเป็น false สี่เหลี่ยมจะบุ๋มลง
public void fill3Drect(int x, int y, int width, int height,boolean b)	วาดสี่เหลี่ยมในระบบ 3 มิติ และระบายสีภายในด้วย
public void drawOval (int x, int y, int width, int height)	วาดวงกลมหรือวงรีภายในโครงสี่เหลี่ยมที่กำ หนดใน parameter ทั้งสี่ โดยเส้นรอบวงจะสัมผัสกับจุดกึ่งกลางด้านแต่ละด้านของสี่เหลี่ยม
public void fillOval(int x, int y, int width, int height)	วาดวงกลมหรือวงรี และระบายสีภายในด้วย

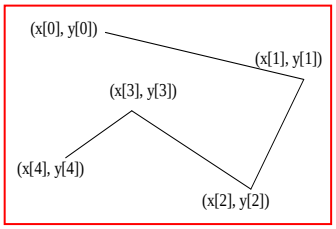
ตัวอย่าง



การวาด Polygons and Polylines Polygons คือ รูปหลายเหลี่ยม ส่วน polylines คือ การวาดเส้นหลายเส้นเชื่อมต่อกัน เช่นต้องการวาดรูปที่มีพิกัดดังนี้

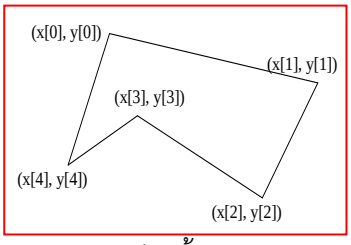
int[] x = {40, 70, 60, 45, 20};

int[] y = {20, 40, 80, 45, 60};



เราสามารถใช้เมธอดต่อไปนี้สำหรับวาดภาพโพลีไลน์

g.drawPolyline(x, y, x.length);



เราสามารถใช้เมธอดต่อไปนี้สำหรับวาดภาพโพลีกอน

g.drawPolygon(x, y, x.length);

method ที่ใช้ในการวาด polygons และ polylines มีดังนี้

Method	Description
public void drawPolygon(int xPoints[ ], int yPoints[ ], int points)	xPoints และ yPoints เป็น array ที่เก็บค่า coordinate ของแต่ละจุดที่จะทำ การวาด polygon method นี้จะวาด polygon ปิด แม้ว่าจุดสุดท้ายกับจุดแรกจะไม่ใช่จุดเดียวกัน argument ตัวสุดท้าย คือ จำนวนจุด
public void drawPolyline(int xPoints[ ], int yPoints[], int points)	เป็นการวาดเส้นเชื่อมจุดแต่ละจุดที่ระบุใน xPointsและ yPoints หากจุดสุดท้ายกับจุดแรกไม่ใช่จุดเดียวกัน ก็จะไม่มีการลากจุดเชื่อมระหว่างจุดแรกกับจุดสุดท้าย
public void drawPolygon (Polygon p)	วาดรูป polygon จาก object p
public void fillPolygon(int xPoints[ ], int yPoints[ ],int points);	วาดรูป polygon และระบายสีภายในรูปด้วย
public Polygon()	เป็นการสร้าง Polygon object แต่ยังไม่มีการระบุจุดต่างๆ ของ polygon
public Polygon(int xValues[ ], int yValues[ ], int numberOfPoints)	เป็นการสร้าง Polygon object พร้อมระบุจุดต่างๆ ของ polygon ด้วย numberOfPoints เป็นจำ นวนด้านของ polygon นั้น

1.2. จากโปรแกรมต่อไปนี้

```
import javax.swing.*;
public class Test extends JApplet {
   public void init() {
       add(new JLabel("OK"));
   }
   public static void main(String[] args) {
       Test applet = new Test();
   }
}
```

ผลลัพธ์จากโปรแกรมเมื่อรันผ่านเว็บบราวเซอร์จะแสดง Label ที่มีข้อความ “OK” โดยลำดับการทำงานของโปรแกรม จะเรียก Constructor แบบไม่มีอาร์กิวเม้นต์ ของคลาสเพื่อทำหน้าที่กำหนดค่าเริ่มต้นเป็นลำดับแรก เมื่อวัตถุถูกสร้างจากคอนสตรักเตอร์เรียบร้อย method init() จะถูกเรียกเป็นลำดับถัดไปเพื่อกำหนดค่าเริ่มต้นและอ่านค่าพารามิเตอร์จากเว็บ

บราวเซอร์เพื่อใช้ในโปรแกรม หลังจากนั้นจะทำในส่วนของ start() -> stop() ->destroy() ต่อไป โดยหากผู้ใช้ต้องการทำงานที่ต้องการสามารถ Overide method เหล่านี้ได้ กรณีเรียกในลักษณะ Application โปรแกรมจะเรียก main() เป็นอันดับแรกซึ่งจากตัวอย่างใน main มีการสร้าง applet แต่ยังไม่แสดงอะไรที่หน้าจอ หากต้องการให้มีปุ่มที่หน้าจอ หลังจากสร้าง object ของ applet เรียบร้อยให้ทำการเรียก method init() เพื่อทำงานต่อไป โดยใช้คำสั่ง applet.init()

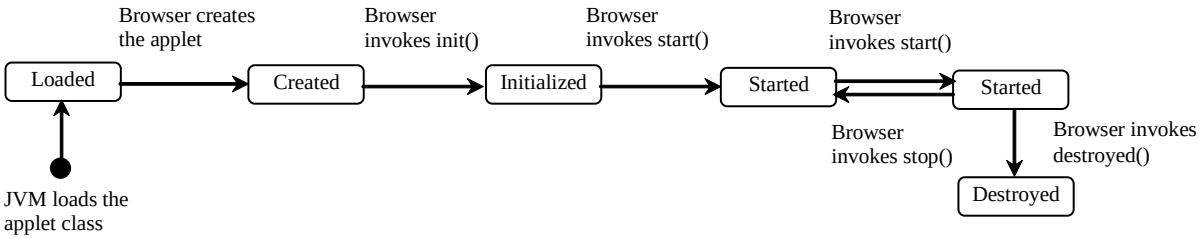
1.3 อธิบายผลลัพธ์ และลำดับขั้นตอนการทำงานของ Applet ต่อไปนี้ method ใดถูกเรียกเป็นอันดับแรก พร้อมทั้งเขียนวงจรการทำงานของ Applet

```
import javax.swing.*;

public class Test extends JApplet {
  public Test() {
      System.out.println("Default constructor is invoked");
  }

  public void init() {
      System.out.println("Init method is invoked");
  }
}
```

ผลลัพธ์จากโปรแกรมเมื่อรันผ่านเว็บบราวเซอร์จะแสดงข้อความ Default constructor is invoked โดยลำดับการทำงานของโปรแกรม จะเรียก Constructor แบบไม่มีอาร์กิวเม้นต์ ของคลาสเพื่อทำหน้าที่กำหนดค่าเริ่มต้นเป็นลำดับแรก เมื่อวัตถุถูกสร้างจากคอนสตรั๊กเตอร์เรียบร้อย method init() จะถูกเรียกเป็นลำดับถัดไปดังนั้นผลลัพธ์ที่เกิดขึ้นคือ Init method is invoked หลังจากนั้นจะทำในส่วนของ start() -> stop() ->destroy() ต่อไป โดยหากผู้ใช้ต้องการทำงานที่ต้องการสามารถ overide method เหล่านี้ได้ วงจรการทำงานของ Applet สามารถเขียนได้ดังนี้



2.ให้ศึกษาและทดลองพิมพ์ตัวอย่างการสร้าง Clock Animation จากตัวอย่างต่อไปนี้

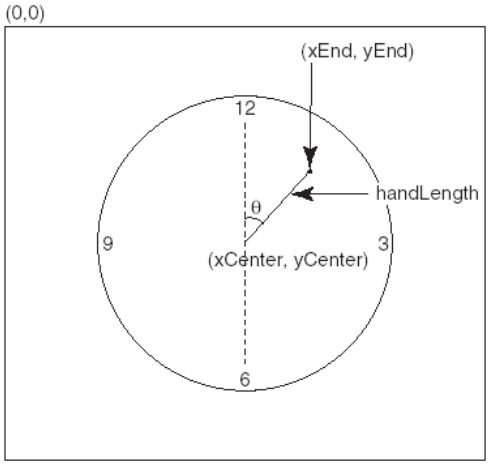
```
1 import java.awt.event.*;
2 import javax.swing.*;
3 import java.awt.*;
4 import java.util.*;
5 import javax.swing.Timer;
6
7 public class ClockAnimation extends JFrame {
8   private StillClock clock = new StillClock();
9
10  public ClockAnimation() {
11    add(clock);
```

```
12
13      // Create a timer with delay 1000 ms
14      Timer timer = new Timer(1000, new TimerListener());
15      timer.start();
16    }
17
18    private class TimerListener implements ActionListener {
19      @Override /** Handle the action event */
20      public void actionPerformed(ActionEvent e) {
21        // Set new time and repaint the clock to display current time
22        clock.setCurrentTime();
23        clock.repaint();
24      }
25    }
26
27    /** Main method */
28    public static void main(String[] args) {
29      JFrame frame = new ClockAnimation();
30      frame.setTitle("ClockAnimation");
31      frame.setSize(200, 200);
32      frame.setLocationRelativeTo(null); // Center the frame
33      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
34      frame.setVisible(true);
35    }
36  }
37
38
 39   class StillClock extends JPanel {
40    private int hour;
41    private int minute;
42    private int second;
43
44    /** Construct a default clock with the current time*/
45    public StillClock() {
46      setCurrentTime();
47    }
48
49    /** Construct a clock with specified hour, minute, and second */
50    public StillClock(int hour, int minute, int second) {
51      this.hour = hour;
52      this.minute = minute;
53      this.second = second;
54    }
55
56    /** Return hour */
57    public int getHour() {
58      return hour;
59    }
60
61    /** Set a new hour */
62    public void setHour(int hour) {
63      this.hour = hour;
64      repaint();
65    }
66
67    /** Return minute */
68    public int getMinute() {
69      return minute;
70    }
71
72    /** Set a new minute */
73    public void setMinute(int minute) {
74      this.minute = minute;
```

```
75      repaint();
76    }
77
78    /** Return second */
79    public int getSecond() {
80      return second;
81    }
82
83    /** Set a new second */
84    public void setSecond(int second) {
85      this.second = second;
86      repaint();
87    }
88
89
90  @Override /** Draw the clock */
91    protected void paintComponent(Graphics g) {
92      super.paintComponent(g);
93
94      // Initialize clock parameters
95      int clockRadius =
96        (int)(Math.min(getWidth(), getHeight()) * 0.8 * 0.5);
97      int xCenter = getWidth() / 2;
98      int yCenter = getHeight() / 2;
99
100     // Draw circle
101     g.setColor(Color.black);
102     g.drawOval(xCenter - clockRadius, yCenter - clockRadius,
103       2 * clockRadius, 2 * clockRadius);
104     g.drawString("12", xCenter - 5, yCenter - clockRadius + 12);
105     g.drawString("9", xCenter - clockRadius + 3, yCenter + 5);
106     g.drawString("3", xCenter + clockRadius - 10, yCenter + 3);
107     g.drawString("6", xCenter - 3, yCenter + clockRadius - 3);
108
109    // Draw second hand
110     int sLength = (int)(clockRadius * 0.8);
111     int xSecond = (int)(xCenter + sLength *
112       Math.sin(second * (2 * Math.PI / 60)));
113     int ySecond = (int)(yCenter - sLength *
114       Math.cos(second * (2 * Math.PI / 60)));
115     g.setColor(Color.red);
116     g.drawLine(xCenter, yCenter, xSecond, ySecond);
117
118
119 // Draw minute hand
120     int mLength = (int)(clockRadius * 0.65);
121     int xMinute = (int)(xCenter + mLength *
122       Math.sin(minute * (2 * Math.PI / 60)));
123     int yMinute = (int)(yCenter - mLength *
124       Math.cos(minute * (2 * Math.PI / 60)));
125     g.setColor(Color.blue);
126     g.drawLine(xCenter, yCenter, xMinute, yMinute);
127
128     // Draw hour hand
129     int hLength = (int)(clockRadius * 0.5);
130     int xHour = (int)(xCenter + hLength *
131       Math.sin((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
132     int yHour = (int)(yCenter - hLength *
133       Math.cos((hour % 12 + minute / 60.0) * (2 * Math.PI / 12)));
134     g.setColor(Color.green);
135     g.drawLine(xCenter, yCenter, xHour, yHour);
136   }
137
```

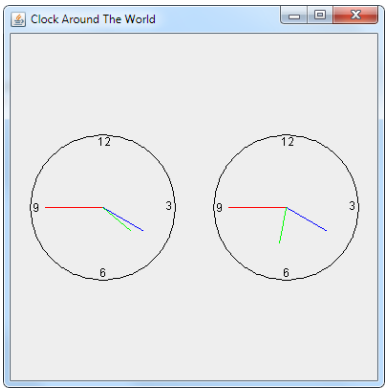
```
138   public void setCurrentTime() {
139     // Construct a calendar for the current date and time
140     Calendar calendar = new GregorianCalendar();
141
142     // Set current hour, minute and second
143     this.hour = calendar.get(Calendar.HOUR_OF_DAY);
144     this.minute = calendar.get(Calendar.MINUTE);
145     this.second = calendar.get(Calendar.SECOND);
146   }
147
148   @Override
149   public Dimension getPreferredSize() {
150     return new Dimension(200, 200);
151   }
152 }
```

2.1 อธิบายหลักการของการสร้างเข็มนาฬิกาแต่ละอันมีหลักการสร้างอย่างไร



2.2 อธิบายขั้นตอนของการเพิ่มเหตุการณ์ที่ทำให้นาฬิกาเดินได้

2.3 สร้างนาฬิกาตามเวลาของประเทศไทยและตามเวลาของประเทศญี่ปุ่น ดังรูปต่อไปนี้



3. ศึกษาและทดลองรันโปรแกรมพร้อมคำอธิบายการทำงานของโปรแกรมแต่ละส่วนต่อไปนี้

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
```

ทำการ Import คลาสและเมธอดต่าง ๆ ที่จำเป็นสำหรับการประมวลผล

```
public class Exercise extends JFrame {
  public Exercise() {
    add(new RaceCar());
  }
```

สร้างคลาส Exercise ที่มีลักษณะเป็นเฟรมโดยการสืบทอดจากคลาส JFrame
กำหนดคอนสตรักเตอร์ของคลาส Exercise ให้สร้างวัตถุใหม่ขึ้นมาจากคลาส RaceCar โดยการเรียกใช้คอนสตรักเตอร์ของ Racecar ด้วยคำสั่ง `new RaceCar()` จากนั้นให้นำวัตถุที่สร้างจากคลาส RaceCar เพิ่มเข้าไปที่คลาส Exercise ที่มีลักษณะเป็น Frame ด้วยคำสั่ง `add(new RaceCar());`

```
  public static void main(String[] args) {
    Exercise frame = new Exercise();
    frame.setTitle("Exercise");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(200, 100);
    frame.setLocationRelativeTo(null); // Center the frame
    frame.setVisible(true);
  }
}
```

`Exercise frame = new Exercise();` ทำการสร้างวัตถุจากคลาส `Exercise`

`frame.setTitle("Exercise");` กำหนดข้อความที่ส่วน Title ของเฟรม เป็น “Excercise”

`frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` กำหนดให้สามารถปิดโปรแกรมโดยการกดปุ่มเครื่องหมายกากบาทบนเฟรมได้

`frame.setSize(200, 100);` กำหนดขนาดความกว้างของเฟรมให้มีขนาด 200 * 100

`frame.setLocationRelativeTo(null);` กำหนดให้เมื่อประมวลผลชุดคำสั่งให้แสดงเฟรมที่ตำแหน่งกลางหน้าจอ

`frame.setVisible(true);` กำหนดให้เฟรมปรากฏที่หน้าจอ

```
class RaceCar extends JPanel {
```

```
  private int xBase = 0;
  private Timer timer = new Timer(10, new Listener());
  public RaceCar() {
    timer.start();
    this.setFocusable(true);
    this.addKeyListener(new KeyAdapter() {
     public void keyPressed(KeyEvent e) {
        if (e.isControlDown() && e.getKeyCode() == 61) {
         if (timer.getDelay() > 5)
          timer.setDelay(timer.getDelay() - 5);
        }
      else if (e.isControlDown() && e.getKeyCode() == 45)
        timer.setDelay(timer.getDelay() + 1);
     }
    });
  }
```

-สร้างคลาส RaceCar ที่มีลักษณะเป็น Panel โดยการสืบทอดจากคลาส JPanel ซึ่งเป็นบริเวณที่จะใช้วาดรถ

-กำหนดตัวแปร xBase มีค่าเป็น 0 ซึ่งเป็นตัวแปรสำหรับเก็บค่าเริ่มต้นของตัวรถ

-สร้างวัตถุจากคลาส Timer และคอยตรวจจับการเกิดเหตุการณ์กับวัตถุ timer โดยการลงทะเบียนการดักจับเหตุการณ์ให้กับ timer โดยใช้คำสั่ง `Timer timer = new Timer(10, new Listener());`

- วัตถุที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์และคอยตอบสนองเหตุการณ์ที่เกิดขึ้นคือวัตถุที่สร้างจากคลาสที่ทำการ Implements อินเตอร์เฟส ActionListener โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด actionPerformed()

-กำหนดคอนสตรักเตอร์ของคลาส RaceCar โดยเมื่อเริ่มสร้างวัตถุจากคลาส RaceCar โปรแกรมจะสั่งให้เวลาจากวัตถุ timer เริ่มเดินโดยการใช้ `timer.start();` ซึ่งเมื่อเวลาเริ่มเดินจะทำให้วัตถุที่สร้างจากคลาส Listener คอยตอบสนองเหตุการณ์ที่เกิดขึ้นโดยการทำการวาดรถที่ตำแหน่งต่าง ๆ ตามเวลาที่เปลี่ยนไป

-กำหนดให้ Panel สามารถตรวจสอบการกดปุ่มจากคีย์บอร์ดได้โดยการลงทะเบียนการดักจับเหตุการณ์ผ่านคลาสที่มีลักษณะเป็น anonymous class โดยการใช้คำสั่ง `addKeyListener()` โดยวัตถุที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์การกดปุ่มและคอยตอบสนองเหตุการณ์ที่เกิดขึ้นคือวัตถุที่สร้างจากคลาส KeyAdapter โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด `keyPressed ( )` เพื่อตรวจสอบเหตุการณ์ของการกดปุ่ม โดยเหตุการณ์ที่เกิดขึ้นกับปุ่มคือ

1. ปุ่ม Ctrl และ ปุ่ม - ตรวจสอบได้จาก `(e.isControlDown() && e.getKeyCode() == 61)` จะทำให้การแสดงผลของรถช้าลง โดยการใช้คำสั่ง `timer.setDelay(timer.getDelay() - 5);`

2. ปุ่ม Ctrl และ ปุ่ม + ตรวจสอบได้จาก `(e.isControlDown() && e.getKeyCode() == 45)` จะทำให้การแสดงผลของรถเร็วขึ้น โดยการใช้คำสั่ง `timer.setDelay(timer.getDelay() + 1);`

```
  class Listener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
     repaint();
    }
  }
```

คลาส Listener เป็นคลาสที่จะทำหน้าที่ตรวจจับการเกิดเหตุการณ์และคอยตอบสนองเหตุการณ์ที่เกิดขึ้นโดยเป็นคลาสที่ Implements อินเตอร์เฟส ActionListener โดยเมื่อทำการสร้างคลาสดังกล่าวขึ้นให้ทำการ override เมธอด actionPerformed() ภายในเมธอด actionPerformed() เมื่อเกิดการเปลี่ยนของเวลาจะทำการเรียกใช้เมธอด repaint() เพื่อทำการวาดตำแหน่งของรถใหม่ โดยการทำงานที่เมธอด paintComponent(Graphics g)

```
  public void paintComponent(Graphics g) {
    super.paintComponent(g);

    int yBase = getHeight();
    if (xBase > getWidth())
      xBase = -20;
    else
      xBase += 1;
```

เมธอด paintComponent(Graphics g) เป็นเมธอดที่หน้าที่ในการวาดรูปบน Panel โดยใช้คำสั่ง super.paintComponent(g); เป็นคำสั่งสำหรับการเคลียร์หน้าจอเก่า และเตรียมพร้อมสำหรับหน้าจอใหม่ ในชุดคำสั่งแรกจะเป็นการเช็คว่าตำแหน่งเริ่มต้นของตัวรถว่าเกินความกว้างของ Panel หรือไม่หาก ไม่เกินจะทำการเพิ่มตำแหน่งของรถไป 1 ตำแหน่ง ในแกน x

```
    g.setColor(Color.BLACK);
    g.fillOval(xBase + 10, yBase - 10, 10, 10);
    g.fillOval(xBase + 30, yBase - 10, 10, 10);
```

วาดล้อรถที่ 1 ที่ตำแหน่งถัดจากตัวรถในแนวแกน x เป็น xBase + 10 และถัดจากตัวรถในแนวแกน y เป็น yBase – 10 โดยความกว้างและความยาวของวงกลมเป็น 10

วาดล้อรถที่ 2 ที่ตำแหน่งถัดจากตัวรถในแนวแกน x เป็น xBase + 30 และถัดจากตัวรถในแนวแกน y เป็น yBase – 10 โดยความกว้างและความยาวของวงกลมเป็น 10

ล้อทั้งสองจะวาดด้วยสีดำจากการใช้คำสั่ง g.setColor(Color.BLACK);

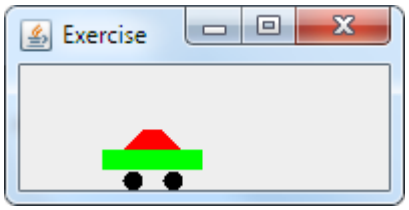
```
    g.setColor(Color.GREEN);
    g.fillRect(xBase, yBase - 20, 50, 10);
```

วาดตัวรถสีเขียวที่มีความกว้างเท่ากับ 50และสูงเท่ากับ 10 ที่ตำแหน่ง xBase, yBase - 20

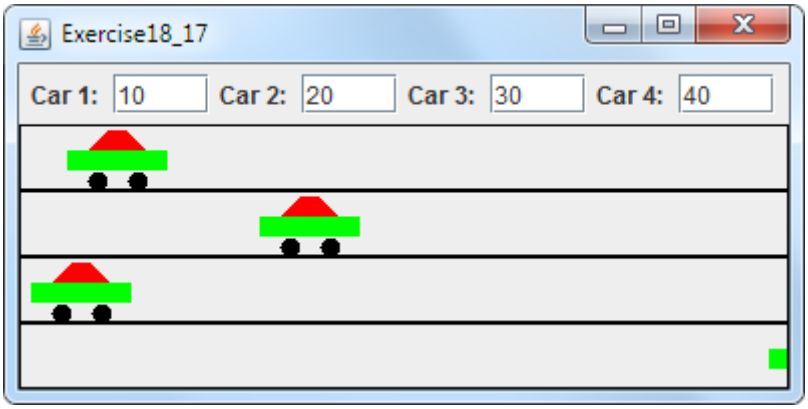
```
    g.setColor(Color.RED);
    Polygon polygon = new Polygon();
    polygon.addPoint(xBase + 10, yBase - 20);
    polygon.addPoint(xBase + 20, yBase - 30);
    polygon.addPoint(xBase + 30, yBase - 30);
    polygon.addPoint(xBase + 40, yBase - 20);
    g.fillPolygon(polygon);
  }
}
```

วาดหลังคารถสีแดงโดยการกำหนดจุดผ่าน Polygon แล้วแสดงผลด้วยการระบายสีบน Polygon ดังกล่าว

ผลลัพธ์ที่ได้จากการรันโปรแกรมทั้งหมดคือ



4. จากคลาส Race car ในปฏิบัติการข้อที่ผ่านมาให้เพิ่มจำนวนรถเป็น 4 คันจำลองให้เป็นสนามแข่งรถ และสามารถกำหนดความเร็วให้รถแต่ละคันได้



5.เขียนโปรแกรมที่แสดงรูปบอลลอนในตำแหน่ง random ใน Panel ให้ใช้ปุ่มลูกศรซ้ายและลูกศรขวาในการเล็งตำแหน่งของปืนให้ตรงกับบอลลูน ใช้ปุ่มลูกศร up-arrow สำหรับยิงลูกกระสุน เมื่อลูกกระสุนถูกบอลลูนให้บอลลูนที่โดนกระสุนหายไป และ random บอลลูนที่ตำแหน่งใหม่ขึ้นมา