Graphics

Compiled by Kanjana Eiamsaard, Version 1.0.2 2022-02-21

Agenda

- กราฟิกส์ & Swing
  - paintComponent()
- Graphics 2D
  - drawLine
  - drawRect
  - drawOval
  - drawArc
  - fillRect
  - fillOval
  - setColor
- การจัดการเมาส์
  - MouseListener
  - MouseMotionListener
  - MouseAdapter
- เกม
  - การโหลดรูป
  - การจัดการแป้นพิมพ์และการเคลื่อนที่
  - ตัวอย่างการสร้างเกมส์

กราฟิกส์

- Swing มีเครื่องมืออำนวยความสะดวก ในการวาดเส้น วาดรูปร่าง โหลดรูปและจัดวางได้ตามต้องการ
- Swing ทำให้เราเขียนโปรแกรมตอบสนองต่อ เมาส์และแป้นพิมพ์ได้
- ทำให้สามารถสร้างเกมหรือโปรแกรมกราฟิกส์ที่มีการตอบโต้กับผู้ใช้ได้
- หากเราวาดภาพบนผืนผ้าใบในความเป็นจริงแล้ว ผ้าใบใน Java ก็คือ **JPanel** โดยอาศัยอ็อบเจกต์ Graphics เป็นเหมือนปากกาหรือพู่กันสำหรับวาด

กลไกการทำงานของกราฟิกส์บน Swing

- คอมโพเนนต์ที่เป็นคอนเทนเนอร์ใน Swing มี 2 ระดับ คือ คอนเทนเนอร์ชั้นบนสุด (top-level container) และ คอนเทนเนอร์ชั้นใน
- คอนเทนเนอร์ชั้นบนสุด ได้แก่ JFrame, JDialog และ JApplet
- คอนเทนเนอร์**ชั้นใน** ได้แก่คอมโพเนนต์ของ Swing ตัวอื่นที่เหลือจากชั้นนอก เช่น

  JPanel , JButton, Jlabel, Jmenu, JTextArea เป็นต้น

  - **มีเมทอด paintComponent()** ที่ถูกเรียกใช้เมื่อมีการวาด
  - เมทอดนี้ถูกเรียกทุกครั้งที่มีเหตุการณ์ที่ต้องวาดตัวเองใหม่ เช่น เปิด เปลี่ยนขนาด ย้าย ย่อ หน้าโปรแกรม เป็น

กลไกการทำงานของกราฟิกส์บน Swing (ต่อ)

- javax.swing.**JComponent** (implements java.io.Serializable)
  - javax.swing.**AbstractButton** (implements java.awt.ItemSelectable, javax.swing.SwingConstants)
    - javax.swing.**JButton** (implements javax.accessibility.Accessible)
    - javax.swing.**JMenuItem** (implements javax.accessibility.Accessible, javax.swing.MenuElement)
      - javax.swing.**JCheckBoxMenuItem** (implements javax.accessibility.Accessible, javax.swing.SwingConstants)
      - javax.swing.**JMenu** (implements javax.accessibility.Accessible, javax.swing.MenuElement)
      - javax.swing.**JRadioButtonMenuItem** (implements javax.accessibility.Accessible)
    - javax.swing.**JToggleButton** (implements javax.accessibility.Accessible)
      - javax.swing.**JCheckBox** (implements javax.accessibility.Accessible)
      - javax.swing.**JRadioButton** (implements javax.accessibility.Accessible)
  - javax.swing.**Box** (implements javax.accessibility.Accessible)
  - javax.swing.**Box.Filler** (implements javax.accessibility.Accessible)
  - javax.swing.**JColorChooser** (implements javax.accessibility.Accessible)
  - javax.swing.**JComboBox**<E> (implements javax.accessibility.Accessible, java.awt.event.ActionListener, java.awt.ItemSelectable, javax.
  - javax.swing.**JFileChooser** (implements javax.accessibility.Accessible)
  - javax.swing.**JInternalFrame** (implements javax.accessibility.Accessible, javax.swing.RootPaneContainer, javax.swing.WindowConstants
  - javax.swing.**JInternalFrame.JDesktopIcon** (implements javax.accessibility.Accessible)
  - javax.swing.**JLabel** (implements javax.accessibility.Accessible, javax.swing.SwingConstants)
    - javax.swing.**DefaultListCellRenderer** (implements javax.swing.ListCellRenderer<E>, java.io.Serializable)
      - javax.swing.**DefaultListCellRenderer.UIResource** (implements javax.swing.plaf.UIResource)
  - javax.swing.**JLayer**<V> (implements javax.accessibility.Accessible, java.beans.PropertyChangeListener, javax.swing.Scrollable)
  - javax.swing.**JLayeredPane** (implements javax.accessibility.Accessible)
    - javax.swing.**JDesktopPane** (implements javax.accessibility.Accessible)
  - javax.swing.**JList**<E> (implements javax.accessibility.Accessible, javax.swing.Scrollable)
  - javax.swing.**JMenuBar** (implements javax.accessibility.Accessible, javax.swing.MenuElement)
  - javax.swing.**JOptionPane** (implements javax.accessibility.Accessible)
  - javax.swing.**JPanel** (implements javax.accessibility.Accessible)

Ref: https://docs.oracle.com/javase/8/docs/api/javax/swing/package-tree.html

คลาส JPanel

- เป็นคลาสที่สำคัญในการสร้างกราฟิกส์ด้วย Swing เพราะเป็นคอมโพเนนต์ว่าง ๆ ที่เราสามารถวาดหรือวางอะไรลงไปก็ได้
- การใช้คลาส JPanel ทำกราฟิกส์จะทำได้โดยการสร้างซับคลาสของ JPanel แล้วโอ **เวอร์ไรด์เมทอด paintComponent()**
- เมทอด paintComponent() จะถูกเรียกได้ใน 2 กรณี
    1. **ระบบ**พิจารณาแล้วเห็นว่าจำเป็นต้องมีการวาดหน้า GUI หรือคอมโพเนนต์บางตัวใหม่ เช่น เปิด ย่อ ย้าย เปลี่ยน หน้าจอ
    2. โปรแกรม(เมอร์) ต้องการให้เกิดการวาดคอมโพเนนต์นั้นใหม่เองโดย **การเรียกเมทอด repaint()**

คลาส Graphics2D

- เมทอด paintComponent() ที่ถูกเรียกเมื่อมีการวาด มีรูปประกาศเต็มคือ

  protected void paintComponent(Graphics g)

- Graphics เป็นคลาสนามธรรม ทำให้อ็อบเจกต์ที่เป็นอาร์กิวเมนต์ของเมทอด paintComponent() เป็นอ็อบเจกต์จากคลาส **Graphics2D** ซึ่งเป็นซับคลาสของ Graphics
- **Graphics2D** เปรียบเสมือนกับ**ปากกา** สามารถสั่ง วาดเส้น รูปร่างต่าง ๆ หรือ ลงสี ได้ **ผ่านการ override method paintComponent()**

ตัวอย่างการวาดเส้น (ต่อ)

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

public class DrawPanel extends JPanel{
    private static final long serialVersionUID = 1L;

    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        int width = getWidth();
        int height = getHeight();
        g.drawLine(0, 0, width, height);
        g.drawLine(0, height, width, 0);
    }
}
```

```java
public class DrawPanelTest {
    public static void main(String[] args) {
        DrawPanel panel = new DrawPanel();
        JFrame app = new JFrame();
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        app.add(panel);
        app.setSize(250, 250);
        app.setVisible(true);
    }
}
```

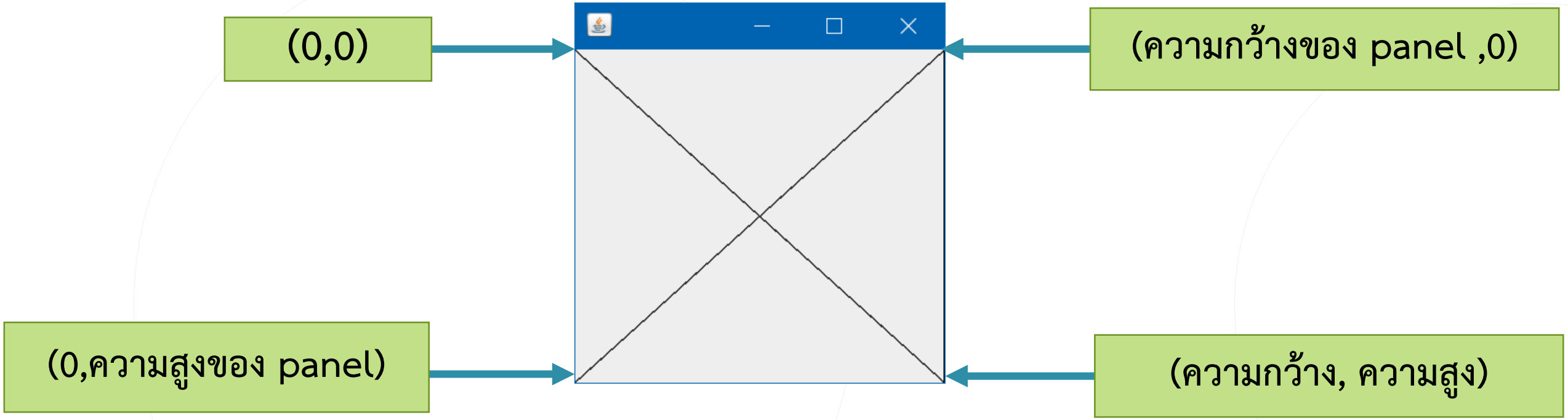
- สร้างซับคลาสของ JPanel
- override method painComponent() โดยเรียกใช้ เมทอดเดียวกันของคลาสแม่ก่อนดำเนินการใด ๆ **เพื่อให้มีการวาดพื้นหลังตามปกติก่อน**
- กำหนด width และ height ตามความกว้างและความสูงของพื้นที่ ๆ วาดได้ของ Panel
- เรียกใช้ method drawLine

วิธีใช้เมทอด drawLine() ของคลาส Graphics

```
void drawLine(int x1, int y1, int x2, int y2)
```

- x1 และ y1 แทนพิกัดของจุด x, y ตั้งต้น
- x2 และ y2 แทนพิกัดของจุด x,y ที่ปลายเส้น

ตัวอย่างการวาดเส้น



- จุดพิกัดตั้งต้นอยู่ตำแหน่ง (0,0) อยู่บนซ้าย ของจอภาพ
- การวาดเส้นอาศัยการกำหนดพิกัดตั้งต้นและพิกัดปลาย

ระบบพิกัดใน Java

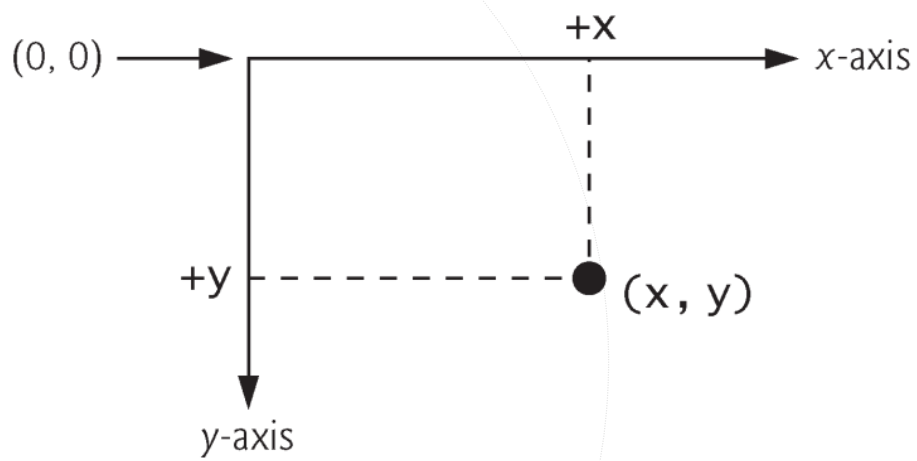


Fig. 4.17 | Java coordinate system. Units are measured in pixels.

Reference P65 – P110

เมทอดของคลาส Graphics ที่น่าสนใจ

- drawRect(int x, int y, int width, int height) วาดเส้นสี่เหลี่ยมโดยมีพิกัดบนซ้ายกำหนดโดย x และ y และขนาดกำหนดโดย width และ height
- fillRect(int x, int y, int width, int height) วาดสี่เหลี่ยมเต็มโดยมีพิกัดบนซ้ายกำหนดโดย x และ y และขนาดกำหนดโดย width และ height
- drawOval(int x, int y, int width, int height) วาดเส้นวงรีในกรอบสี่เหลี่ยมที่ระบุ
- fillOval(int x, int y, int width, int height) วาดวงรีเต็มในกรอบสี่เหลี่ยมที่ระบุ

เมทอดของคลาส Graphics ที่น่าสนใจ (ต่อ)

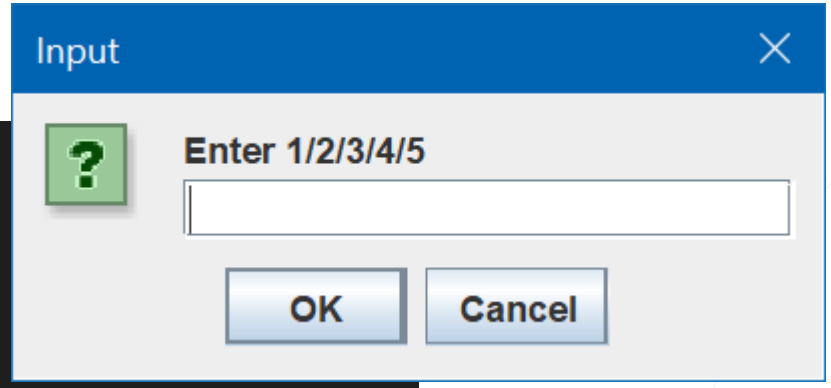
- drawArc (int x, int y, int width, int length, int startAngle, int arcAngle) วงกลม/วงรีที่อยู่ในรูปสี่เหลี่ยม (รูปสี่เหลี่ยมเป็นเพียงจินตนาการ)
- drawLine (int x1, int y1, int x2, int y2) วาดเส้นตรงจากตำแหน่ง (x1,y1) ไปยัง (x2,y2)
- drawPolyLine(int[] xPoints, int[] yPoints, int nPoints) วาดส่วนของเส้นตรงต่อกันโดยอาร์เรย์ของพิกัดเป็นตัวระบุตำแหน่งของจุดเชื่อมต่อ nPoints จุด
- setColor(Color c) กำหนดสีให้เส้นหรือการลงสี

ตัวอย่างการใช้เมทอดของคลาส Graphics (1/9)

```java
import javax.swing.JFrame;
import javax.swing.JOptionPane;

public class ShapesTest {
    public static void main(String[] args) {
        String input = JOptionPane.showInputDialog("Enter 1/2/3/4/5");
        int userChoice = Integer.parseInt(input);

        Shapes panel = new Shapes(userChoice);
        JFrame app = new JFrame();
        app.add(panel);
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        app.setSize(250, 250);
        app.setVisible(true);
    }
}
```

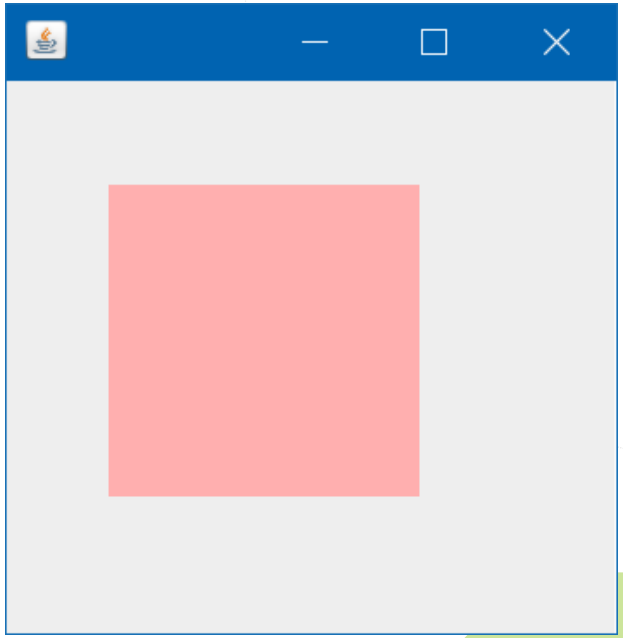


- JOptionPane.showInputDialog ทำให้เกิดหน้าต่างแบบ dialog input

ตัวอย่างการใช้เมทอดของคลาส Graphics (2/9)

```
6   public class Shapes extends JPanel {
7
8       private static final long serialVersionUID = 1L;
9       private int choice;
10
11      public Shapes(int userChoice) {
12          this.choice = userChoice;
13      }
14
15      @Override
16      public void paintComponent(Graphics g) {
17          super.paintComponent(g);
18          switch (this.choice) {
19          case 1:
20              g.setColor(Color.PINK);
21              g.fillRect(40, 40, 120, 120);
22              break;
```

- อย่าลืมเรียกพื้นหลังเดิมมาใช้ด้วย super.paintComponent()
- Case#1 วาดรูปสี่เหลี่ยมทึบด้วยสีชมพู
- เริ่มต้นที่พิกัด (40,40) ขนาด 120 x 120

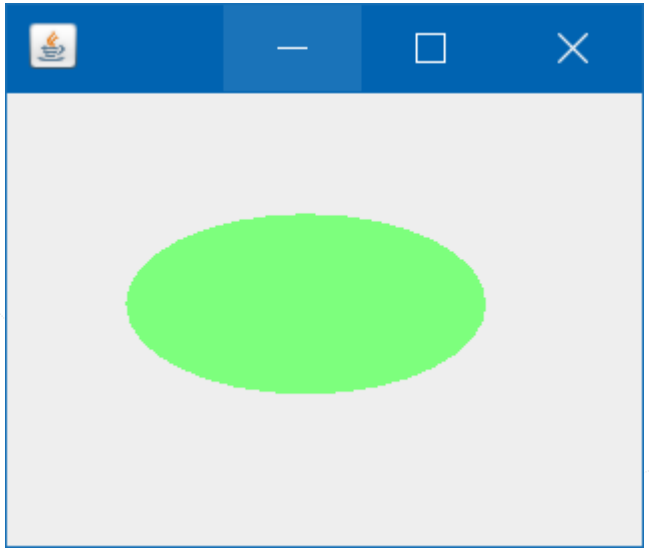


fillRect(int x, int y, int width, int height) วาดสี่เหลี่ยมเต็มโดยมีพิกัดบนซ้ายกำหนดโดย x และ y และขนาดกำหนดโดย width และ height

ตัวอย่างการใช้เมทอดของคลาส Graphics (3/9)

```
case 2:
    g.setColor(new Color(125, 255, 125));
    g.fillOval(40, 40, 120, 60);
    break;
```

fillOval(int x, int y, int width, int height) วาดวงรีเต็มในกรอบสี่เหลี่ยมที่ระบุ

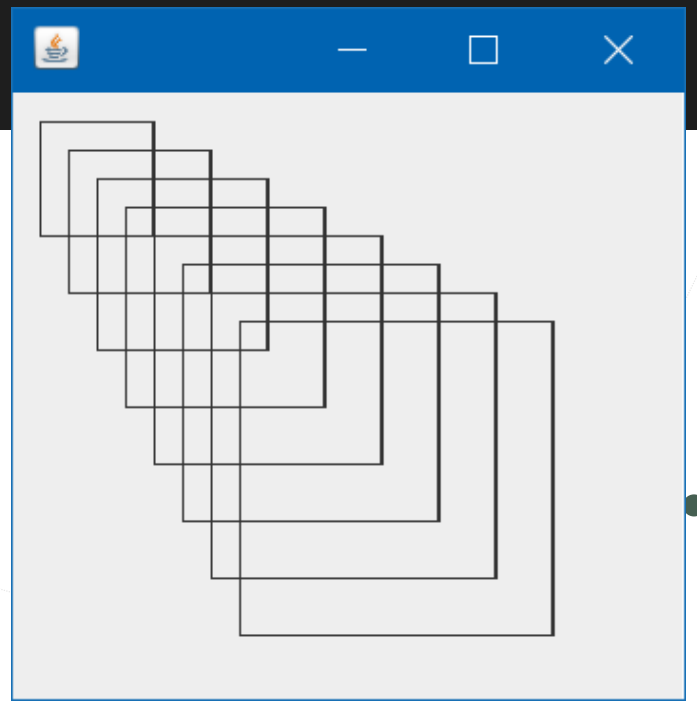


- Case#2 วาดรูปวงรีทึบด้วยสีผสมเองด้วยแม่สี RGB
- เริ่มต้นที่พิกัด (40,40) ขนาด 120 x 60

ตัวอย่างการใช้เมทอดของคลาส Graphics (4/9)

drawRect(int x, int y, int width, int height) วาดเส้นสี่เหลี่ยมโดยมีพิกัดบนซ้าย กำหนดโดย x และ y และขนาดกำหนดโดย width และ height

```
case 3:
    for (int i = 0; i < 8; i++) {
        g.drawRect(10 + i * 10, 10 + i * 10,
        40 + i * 10, 40 + i * 10);
    }
    break;
```

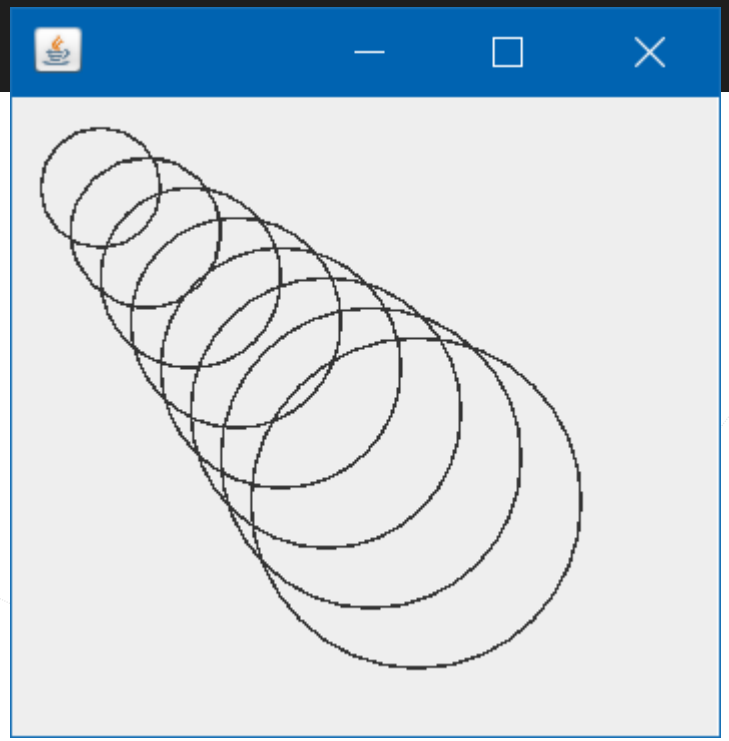


- Case#3 วาดรูปสี่เหลี่ยมซ้อมกัน จำนวน 9 รูป
  - เริ่มต้นที่พิกัด (10,10) ขนาด 40 x 40
  - ตำแหน่งถัดไป จะเลื่อนพิกัดตั้งต้นไปอีก 10 ทั้งแกน x และ y
  - ขนาดจะเพิ่มขึ้นประมาณ 10*i เท่า ทั้งความกว้างและความสูง

ตัวอย่างการใช้เมทอดของคลาส Graphics (5/9)

drawOval(int x, int y, int width, int height) วาดเส้นวงรีในกรอบสี่เหลี่ยมที่ระบุ

```
case 4:
    for (int i = 0; i < 8; i++) {
        g.drawOval(10 + i * 10, 10 + i * 10,
        40 + i * 10, 40 + i * 10);
    }
    break;
```

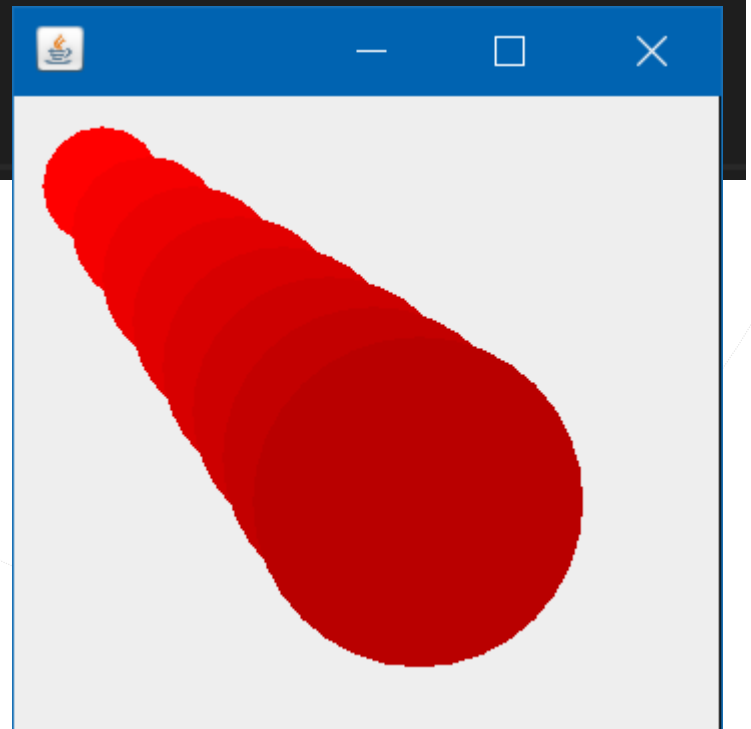


- Case#4 วาดรูปวงกลมซ้อนกัน จำนวน 9 รูป
- เริ่มต้นที่พิกัด (10,10) ขนาด 40 x 40
- ตำแหน่งถัดไป จะเลื่อนพิกัดตั้งต้นไปอีก 10 ทั้งแกน x และ y
- ขนาดจะเพิ่มขึ้นประมาณ 10*i เท่า ทั้งความกว้างและความสูง

ตัวอย่างการใช้เมทอดของคลาส Graphics (6/9)

fillOval(int x, int y, int width, int height) วาดวงรีเต็มในกรอบสี่เหลี่ยมที่ระบุ

```
case 5:
    for (int i = 0; i < 8; i++) {
        g.setColor(new Color(255 - (i * 10), 0, 0));
        g.fillOval(10 + i * 10, 10 + i * 10,
        40 + i * 10, 40 + i * 10);
    }
    break;
}
```



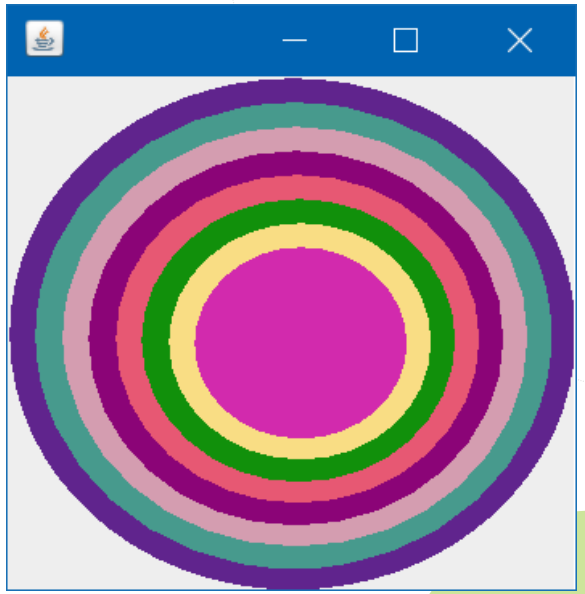
- Case#5 วาดรูปวงกลมทึบซ้อนกัน จำนวน 9 รูป
- เริ่มต้นที่พิกัด (10,10) ขนาด 40 x 40
- ตำแหน่งถัดไป จะเลื่อนพิกัดตั้งต้นไปอีก 10 ทั้งแกน x และ y
- ขนาดจะเพิ่มขึ้นประมาณ 10*i เท่าทั้งความกว้างและความสูง
- สังเกตุการไล่สี

ตัวอย่างการใช้เมทอดของคลาส Graphics (7/9)

fillOval(int x, int y, int width, int height) วาดวงรีเต็มในกรอบสี่เหลี่ยมที่ระบุ

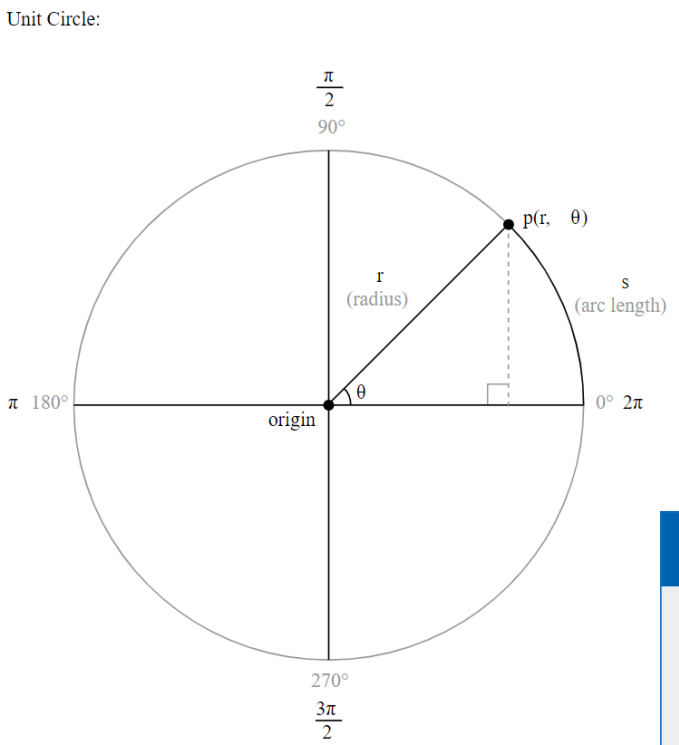
```
case 6:
    int width = getWidth();
    int height = getHeight();
    int gapW = width / 20;
    int gapH = height / 20;
    int xorg = (width / 10) - (gapW * 2);
    int yorg = (height / 10) - (gapH * 2);
    int gapRatioX = (width / 10) - (xorg * 2);
    int gapRatioY = (height / 10) - (yorg * 2);
    for (int i = 0; i < 8; i++) {
        Random rand = new Random();
        float red = rand.nextFloat();
        float green = rand.nextFloat();
        float blue = rand.nextFloat();
        g.setColor(new Color(red,green,blue));
        g.fillOval(xorg+i*gapW,yorg+i*gapH,
                (width-xorg)-i*gapRatioX,
                (height-yorg)-i*gapRatioY);
    }
    break;
```

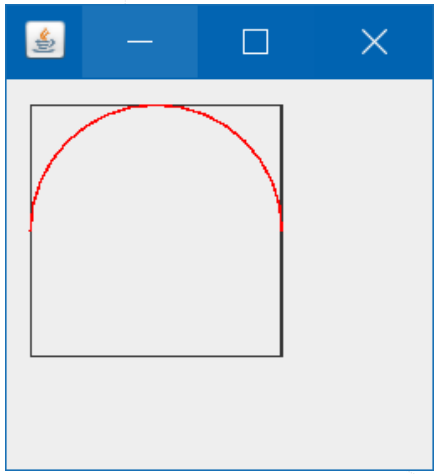
- Case#6 วาดรูปวงกลมซ้อนจากใหญ่ไปเล็ก
- ขนาดใหญ่สุดเท่ากับความกว้างและความสูงของ panel
- ถ้าหน้าจอขนาด 200x200 ตำแหน่งเริ่มต้นคือ (0,0) แต่โปรแกรมนี้รองรับหน้าจอหลายขนาด
- วงกลมชั้นในจะสร้างห่างจากชั้นนอกเท่ากับ 20
- Random สี



ตัวอย่างการใช้เมทอดของคลาส Graphics (8/9)

```
case 7:
g.drawRect(10, 10, 100, 100);
//startAngle = 0 is at 3'O clock
//arcAngle = 180 is at 9'O clock
g.setColor(Color.red);
g.drawArc(10, 10, 100, 100, 0, 180);
break;
```



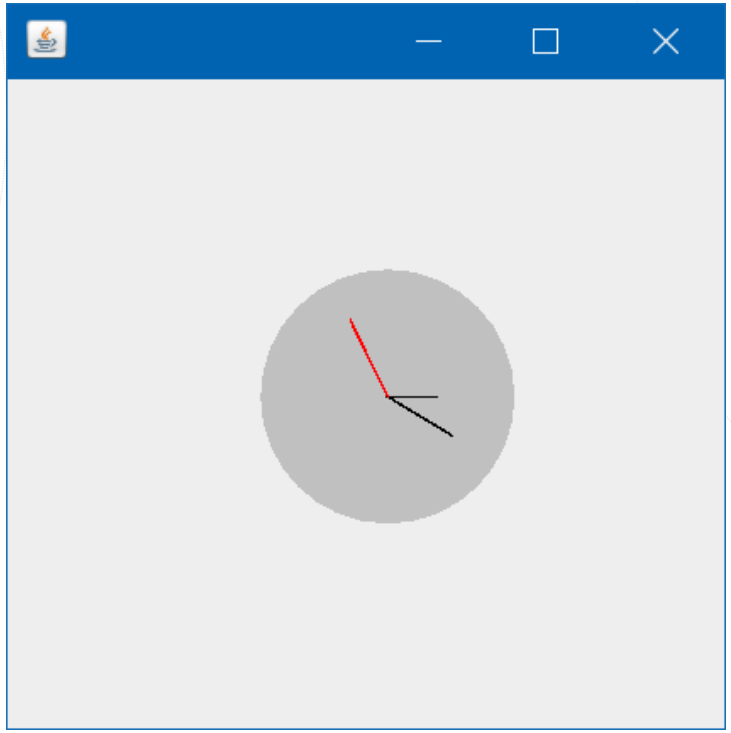


- drawAcr เป็นการวาดรูปวงกลมหรือวงรีภายในรูปสี่เหลี่ยมในจินตนาการ
  - Parameter คู่แรกคือจุดเริ่มต้นในแนวแกน x,y
  - Parameter คู่ที่สองคือ ความกว้างและความสูง
  - Parameter คู่ที่สามคือ องศาในจุดเริ่มต้นและองศาของจุดสุดท้าย

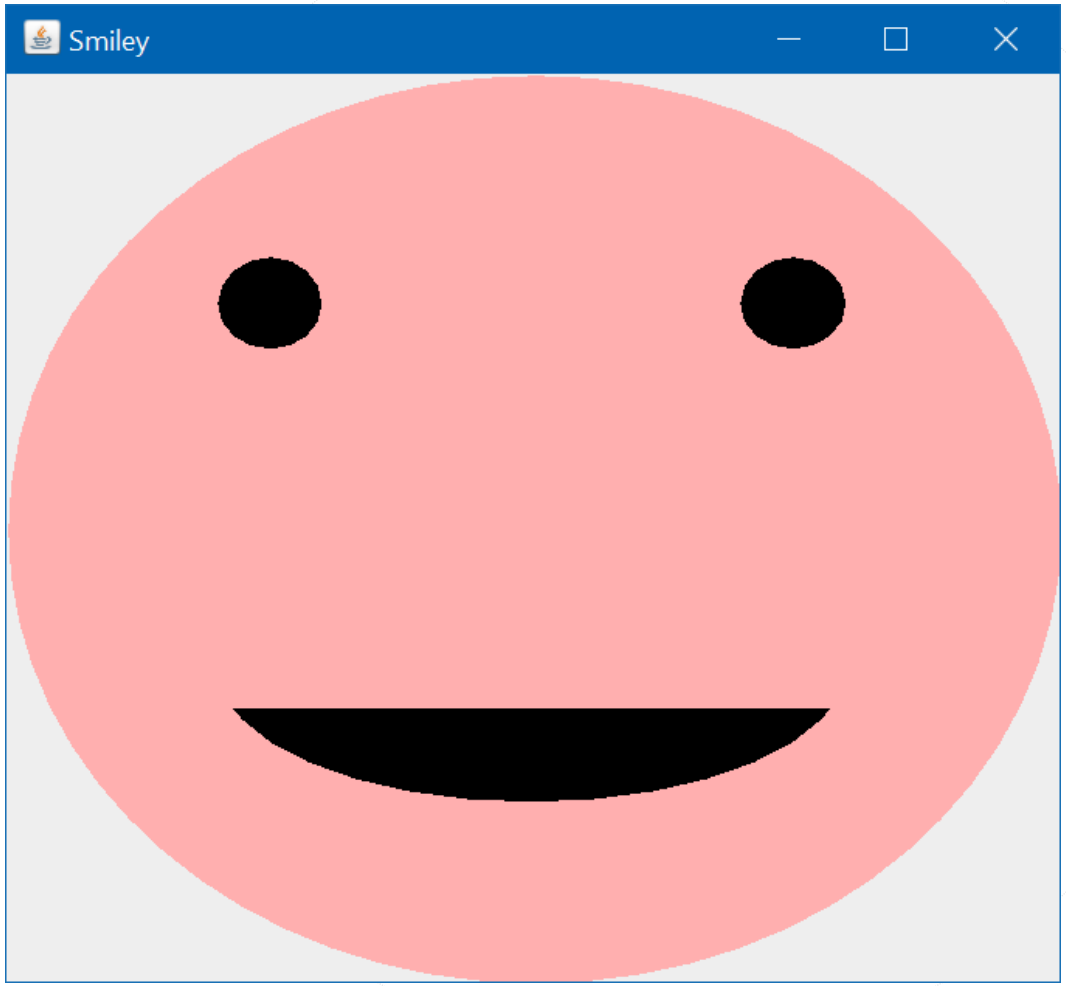
ตัวอย่างการใช้เมทอดของคลาส Graphics (9/9)

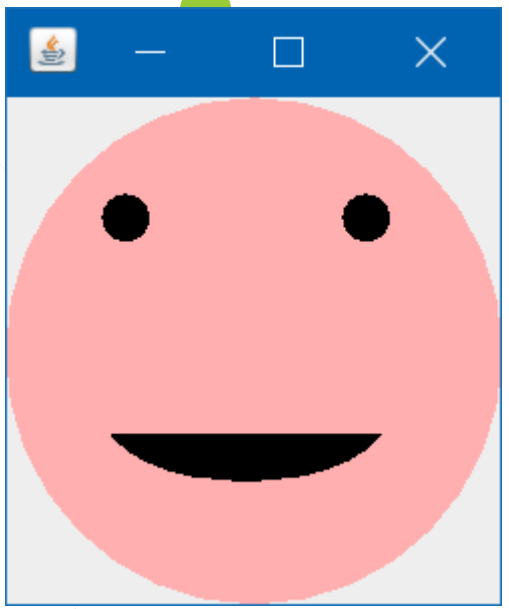
```
case 8:
    g.setColor(Color.lightGray);
    g.fillOval(100, 75, 100, 100);
    g.setColor(Color.BLACK);
    g.drawLine(150, 125, 169, 125);
    g.drawLine(150, 125, 175, 140);
    g.setColor(Color.RED);
    g.drawLine(150,125,135,95);
break;
```

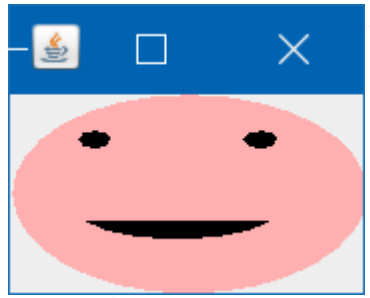
- drawLine (int x1, int y1, int x2, int y2) วาดเส้นตรงจากตำแหน่ง (x1,y1) ไปยัง (x2,y2)



ตัวอย่างหน้ายิ้มยืดหยุ่น







```java
public class Smiley extends JPanel {
    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        // First, draw big pink circle at 0,0 as face
        int width = getWidth();
        int height = getHeight();
        int gapW = width / 20;
        int gapH = height / 20;
        int xorg = (width / 10) - (gapW * 2);
        int yorg = (height / 10) - (gapH * 2);
        int gapRatioX = (width / 10);
        int gapRatioY = (height / 10);
        g.setColor(Color.PINK);
        g.fillOval(xorg, yorg, width, height);

        // Second, draw eyes
        g.setColor(Color.BLACK);
        g.fillOval(xorg + gapRatioX * 2, yorg + gapRatioY * 2, gapRatioX, gapRatioY);
        g.fillOval(xorg + gapRatioX * 7, yorg + gapRatioY * 2, gapRatioX, gapRatioY);

        //Third, draw whole mouth
        g.setColor(Color.black);
        g.fillOval(xorg + gapRatioX * 2, yorg + gapRatioY * 5, gapRatioX*6, gapRatioY*3);

        //Forth, draw rectangle on top of whole mouth
        g.setColor(Color.PINK);
        g.fillRect(xorg + gapRatioX * 2, gapRatioY * 4, gapRatioX*6, gapRatioY*3);
    }
```

ตัวอย่างหน้ายิ้มยืดหยุ่น

การจัดการเมาส์ (MouseEvent)

- MouseListener
- MouseMotionListener
- MouseAdapter

การจัดการเมาส์

- อีเวนต์ที่เกิดจากเมาส์ คือ MouseEvent และ MouseWheelEvent
  - MouseEvent เป็นอีเวนต์ที่เกิดขึ้นตอนคลิกหรือขยับเมาส์
  - MouseWheelEvent เป็นอีเวนต์ที่เกิดขึ้นตอนเลื่อน Scroll wheel
- MouseEvent ทำงานร่วมกับผู้จัดการอีเวนต์ 2 ประเภท คือ
  - MouseListener นำมา implement เพื่อรองรับ การกดปุ่มเมาส์ การเลื่อนเมาส์เข้าหรือออก**จากพื้นที่ของคอมโพเนนต์**
  - MouseMotionListener นำมา implement เพื่อรองรับ การขยับของเมาส์ ทั้งการเลื่อนเมาส์ปกติ และการคลิกปุ่มเมาส์ค้างไว้แล้วเลื่อน

MouseListener methods must be override

- void mouseClicked(MouseEvent e) ถูกเรียกเมื่อมีการคลิก **(กดแล้วปล่อย)** ปุ่มเมาส์
- void mousePressed(MouseEvent e) ถูกเรียกเมื่อมีการกดปุ่มเมาส์
- void mouseReleased(MouseEvent e) ถูกเรียกเมื่อมีการปล่อยปุ่มเมาส์
- void mouseEntered(MouseEvent e) ถูกเรียกเมื่อตัวชี้ของเมาส์ข้ามเข้าไปในพื้นที่ของคอมโพเนนต์
- void mouseExited(MouseEvent e) ถูกเรียกเมื่อตัวชี้ของเมาส์ข้ามออกมาจากพื้นที่ของคอมโพเนนต์

MouseMotionListener methods must be override

- void mouseMoved(MouseEvent e) ถูกเรียกเมื่อมีการขยับเมาส์
- void mouseDragged(MouseEvent e) ถูกเรียกเมื่อมีการขยับเมาส์โดยที่มีการกดปุ่มค้างไว้ด้วย (การลาก)

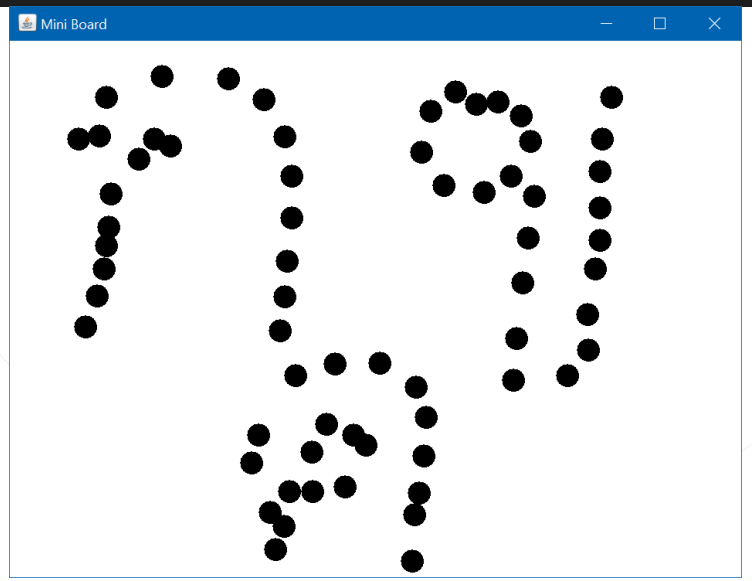
ตัวอย่างการ implement interface MouseListener 1/4

```
@Override
public void mousePressed(MouseEvent e) {
    System.out.println("Pressed at "
    + e.getX() + "," + e.getY());
    xPoints.add(e.getX());
    yPoints.add(e.getY());
    inkLevel = 0.0f;
    repaint();
}
```

- Implement interface MouseListener ใน method mousePressed
  - เก็บตำแหน่ง mouse ในแนวแกน x,y เมื่อมีการกดเมาส์
  - เรียก repaint() เพื่อสั่งให้ paintComponent() ทำงาน

ตัวอย่างการ implement interface MouseListener 2/4

```java
@Override
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    for (int i = 0; i < xPoints.size(); i++) {
        g.setColor(Color.black);
        g.fillOval(xPoints.get(i), yPoints.get(i), 20, 20);
    }
}
```



- เมทอด paintComponent กำหนดสีของพู่กันและวาดวงกลมขนาด 20x20 ในตำแหน่งที่อยู่ในตัวแปร xPoints และ yPoints
- xPoints และ yPoints มีขนาดเท่ากัน ซึ่งขนาดเท่ากับจำนวนครั้งที่คลิก

ตัวอย่างการ implement interface MouseListener 3/4

```
18      addMouseListener(new MouseListener() {
19
20          @Override
21          public void mousePressed(MouseEvent e) {···
29          @Override
30          public void mouseClicked(MouseEvent e) {
31              System.out.println("Clicked at " + e.getX() + "," + e.getY());
32          }
33          @Override
34          public void mouseReleased(MouseEvent e) {
35              System.out.println("Released at " + e.getX() + "," + e.getY());
36          }
37
38          @Override
39          public void mouseEntered(MouseEvent e) {
40              System.out.println("Enter at " + e.getX() + "," + e.getY());
41          }
42
43          @Override
44          public void mouseExited(MouseEvent e) {
45              System.out.println("Exit from " + e.getX() + "," + e.getY());
46          }
47      });
48  }
```

- เมทอดอื่น ๆ ใน interface MouseListener **หากไม่ต้องการให้ทำอะไร ให้ Override โค้ดเปล่าไว้**

ตัวอย่างการ implement interface MouseListener 4/4

```
4   public class DrawingBoardTest{
5       public DrawingBoardTest(){
6           DrawingBoard panel = new DrawingBoard();
7           JFrame app = new JFrame();
8           app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9           app.add(panel);
10          app.setSize(500, 500);
11          app.setVisible(true);
12      }
13      public static void main(String[] args) {
14          SwingUtilities.invokeLater(new Runnable(){
15              @Override
16              public void run(){
17                  new DrawingBoardTest();
18              }
19          });
20      }
21  }
```

- เมทอด main เรียก invokeLater เพื่อส่งต่องานด้าน ในที่เกิดขึ้นใน constructor DrawingBoardTest() ให้กับ event dispatch thread (EDT) เป็นผู้กระทำ
- งานใน DrawingBoardTest() เป็นงานด้าน GUI

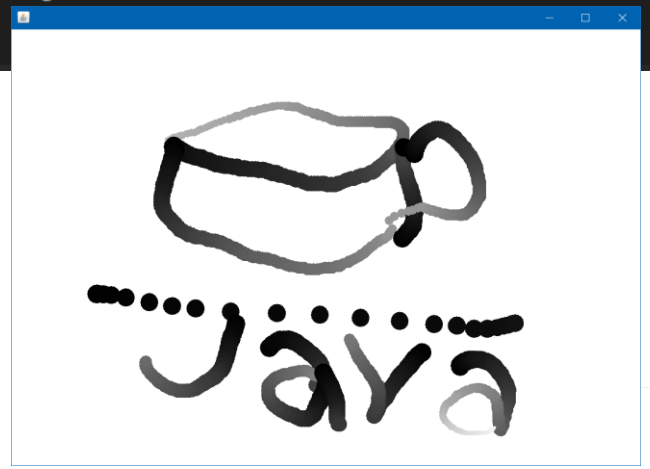
ตัวอย่างการ implement interface MouseMotionListener 1/4

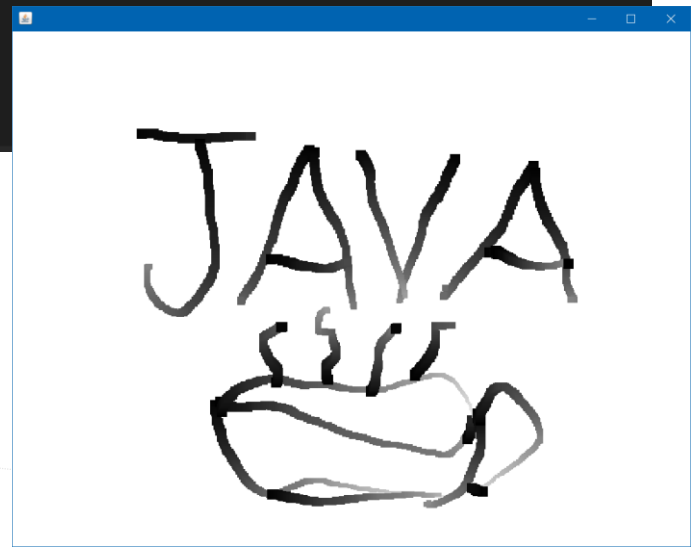
```
18    public DrawingBoard() {
19        super();
20        setBackground(Color.white);
21        addMouseListener(new MouseListener() {···
55        // mouse motion
56        addMouseMotionListener(new MouseMotionListener() {
57            @Override
58            public void mouseMoved(MouseEvent e) {
59
60            }
61
62            @Override
63            public void mouseDragged(MouseEvent e) {
64                xPoints.add(e.getX());
65                yPoints.add(e.getY());
66                /* Add more white until reah to 1.0f */
67                inkLevel += DECRESE_INK_RATIO;
68                if (inkLevel > 1.0f) {
69                    inkLevel = 1.0f;
70                }
71                inkLevels.add(inkLevel);
72                repaint();
73            }
74        });
75    }
```

- จากตัวอย่าง MouseListener ก่อนหน้า ข้อจำกัดอยู่ที่ไม่สามารถวาดภาพด้วยการกดเมาส์ค้างพร้อมกับการเลื่อนได้
- เพิ่มตัวจัดการอีเวนต์ในกลุ่มการเคลื่อนไหวของเมาส์ (MouseMotionListener)
- Implement method mouseDragged
  - เพิ่มสีขาว แต่ไม่ให้เกิน 1.0f

ตัวอย่างการ implement interface MouseMotionListener 2/4

```java
@Override
public void paintComponent(Graphics g) {
    super.paintComponent(g);
    for (int i = 0; i < xPoints.size(); i++) {
        float ink = this.inkLevels.get(i);
        // width should get small when ink turn into white
        int width = (int)(20*(0.2f +
                (1.0f - ink) * 1.0f));
        g.setColor(new Color(ink, ink, ink));
        g.fillOval(xPoints.get(i), yPoints.get(i)
        , width, width);
    }
}
```





- ในเมทอดการวาด เพิ่มความสามารถ
  - การปรับขนาดพู่กัน
  - การไล่เฉดสี
  - กลไกที่ใช้ปรับขนาดและสีเกิดขึ้นที่ เมทอดสำหรับจัดการอีเวนต์ของเมาส์
- ลายเส้นที่แตกเป็นจุด เหตุมาจากอีเวนต์เกิดไม่ทัน เพราะเลื่อนเมาส์เร็ว

ตัวอย่างการใช้คลาส MouseAdapter

```
17      public DrawingBoardAdapter() {
18          super();
19          setBackground(Color.white);
20          addMouseListener(new MouseAdapter() {
21
22              @Override
23              public void mousePressed(MouseEvent e) {
24                  xPoints.add(e.getX());
25                  yPoints.add(e.getY());
26                  inkLevel = 0.0f;
27                  inkLevels.add(inkLevel);
28                  repaint();
29              }
30          });
31          //register mouse motion listener
32          addMouseMotionListener(new MouseAdapter(){
33              @Override
34              public void mouseDragged(MouseEvent e) {
35                  xPoints.add(e.getX());
36                  yPoints.add(e.getY());
37                  /* Add more white until reah to 1.0f */
38                  inkLevel += DECRESE_INK_RATIO;
39                  if (inkLevel > 1.0f) {
40                      inkLevel = 1.0f;
41                  }
42                  inkLevels.add(inkLevel);
43                  repaint();
44              }
45          });
```

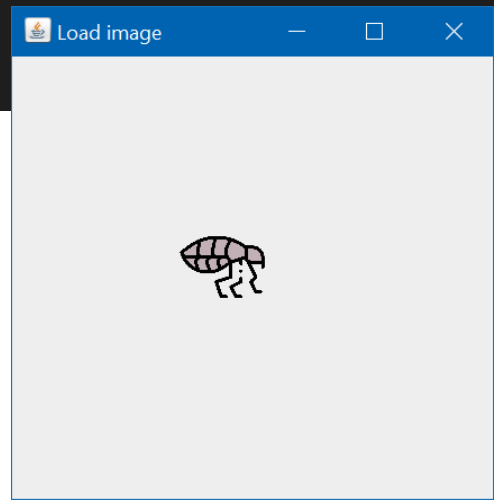
- จากตัวอย่างการจัดการเมาส์ก่อนหน้า พบความไม่สะดวกในการเขียนโปรแกรมคือ “**หากไม่ต้องการให้ทำอะไร ให้ Override โค้ดเปล่าไว้**”
- MouseAdapter เป็นคลาสนามธรรมที่ Implement คลาส MouseListener, MouseWheelListener, MouseMotionListener
- จึงไม่ต้อง override method ที่ไม่
- ถูกเรียกใช้

เกม (Game)

- การโหลดรูป
- การจัดการแป้นพิมพ์และการเคลื่อนที่
- ตัวอย่างการสร้างเกมส์

การโหลดรูป (1/2)

```java
public static void main(String[] args) {
    SwingUtilities.invokeLater(new Runnable(){
        @Override
        public void run(){
            new LoadImage();
        }
    });
}
```



- main thread ส่งการทำงานด้าน GUI ให้กับ EDT ผ่าน SwingUtilities
- การทำงานด้าน GUI ทั้งหมดส่งต่อให้ constructor ของคลาส LoadImage

การโหลดรูป (2/2)

```java
public class LoadImage extends JPanel {
    private static final long serialVersionUID = 1L;
    private BufferedImage img;
    public LoadImage(){
        JFrame app = new JFrame("Load image");
        try {
            img = ImageIO.read(new File("pic/bug.png"));
            if(img == null){
                System.err.println("Unrecognized image type.");
                System.exit(99);
            }
        }catch (IOException e){
            System.err.println(e.toString());
            System.exit(88);
        }
        app.add(this);
        app.setSize(300, 300);
        app.setVisible(true);
        app.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
    @Override
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        g.drawImage(img, 100, 100, 50, 50, null);
    }
```

- นำเข้าไฟล์ผ่านคลาส ImageIO
- สร้างกลไกเพื่อควบคุมข้อผิดพลาดด้วย try...catch
  - สิ่งที่ต้องการไว้ใน try
  - ถ้ามีข้อผิดพลาดให้จับไว้ด้วย catch
- ให้เมทอดการวาด (paintComponent) วาดอ็อบเจกต์ g

ประยุกต์การโหลดรูปและเมาส์อีเวนต์

```
17  public LoadImageMouseIn(){
18      JFrame app = new JFrame("Load image");
19      this.setBackground(Color.PINK);
20      try {
21          img = ImageIO.read(new File("pic/bug.png"));
22          if(img == null){
23              System.err.println("Unrecognized image type.");
24              System.exit(99);
25          }
26      }catch (IOException e){···
30
31      addMouseListener(new MouseAdapter(){
32          @Override
33          public void mouseEntered(MouseEvent e){
34              isShowImg = true;
35              repaint();
36          }
37          public void mouseExited(MouseEvent e){
38              isShowImg = false;
39              repaint();
40          }
41      });
```

```
@Override
public void paintComponent(Graphics g){
    super.paintComponent(g);
    if(isShowImg)
        g.drawImage(img, 100, 100, 50, 50, null)

}
```

- ผลลัพธ์ที่เกิดขึ้นเป็นอย่างไร ?

การจัดการแป้นพิมพ์ (1/)

```
11  public class KeyControl extends JPanel{
12      private static final long serialVersionUID = 1L;
13      private int xPos = 100;
14      private int yPos = 100;
15      private BufferedImage img;
16      public KeyControl(){
17          setBackground(Color.lightGray);
18          setFocusable(true);
19          addKeyListener(new KeyListener(){
20              @Override
21  >           public void keyPressed(KeyEvent e){···
48              @Override
49  >           public void keyReleased(KeyEvent e){···
52
53              @Override
54  >           public void keyTyped(KeyEvent e) {···
58          });
59      }
60  >   private void loadImage(){···
72      @Override
73      public void paintComponent(Graphics g) {
74          super.paintComponent(g);
75          this.loadImage();
76          g.drawImage(this.img, xPos, yPos, 50, 50, null);
77      }
78  }
```

- เมทอด keyControl ทำหน้าที่ กำหนดสิ่งที่จะเกิดขึ้นเมื่อเกิดอีเวนต์ของแป้นพิมพ์
- setFocusable(true) หมายถึง คอมโพเนนต์นี้กำลังถูกจับจ้องอยู่
  - จากตัวอย่าง คอมโพเนนต์ที่ถูกโฟกัส คือ panel
- Interface KeyListener คือคลาสสำหรับจัดการอีเวนต์ที่เกิดจากคีย์บอร์ด ประกอบด้วย abstract method ที่ต้อง implement คือ
  - void keyPressed
  - void keyReleased
  - void keyTyped

การจัดการแป้นพิมพ์ (1/2)

```
addKeyListener(new KeyListener(){
    @Override
    public void keyPressed(KeyEvent e){
        switch(e.getKeyCode()){
            case KeyEvent.VK_LEFT:
            case 65://a
                xPos -= 10;
                repaint();
            break;
            case KeyEvent.VK_RIGHT:
            case 68://d
                xPos += 10;
                repaint();
            break;
            case KeyEvent.VK_UP:
            case 87://w
                yPos -= 10;
                repaint();
            break;
            case KeyEvent.VK_DOWN:
            case 83://s
                yPos += 10;
                repaint();
            break;
            default:
                System.err.println("Press uncontrol key");
            break;
        }
    }
```

- เมทอด keyPressed ถูก implement เพื่อดักจับการกดคีย์ และกำหนดพฤติกรรมให้กับโปรแกรม จากนั้นเรียกเมทอด repaint() เพื่อสั่งให้วาด component อีกครั้ง
- เมทอด getKeyCode รีเทิร์นเลขจำนวนเต็ม ของคีย์ที่กด
- case ติดกันที่ไม่มี break; ให้ความหมายเท่ากับ OR

การจัดการแป้นพิมพ์ (ต่อ)

- keyPressed() คือ เมทอดที่ถูกเรียกเมื่อมีการกดคีย์ใด ๆ บนแป้นพิมพ์
- keyReleased() คือ เมทอดที่ถูกเรียกเมื่อมีการปล่อยคีย์ใด ๆ
- keyTyped() คือ เมทอดที่ถูกเรียกเมื่อมีการปล่อยคีย์ ที่ยกเว้น Control, Shift, Alt, F1
- จากโปรแกรมตัวอย่าง จะเห็นว่า เกิดเมทอดที่ไม่มีโค้ดภายในเมทอด แต่ต้องเขียนไว้ เพราะเป็นข้อกำหนด เพื่อความสะดวกเราสามารถนำ Adapter interface ที่ชื่อว่า “KeyAdapter” มาใช้แทนได้

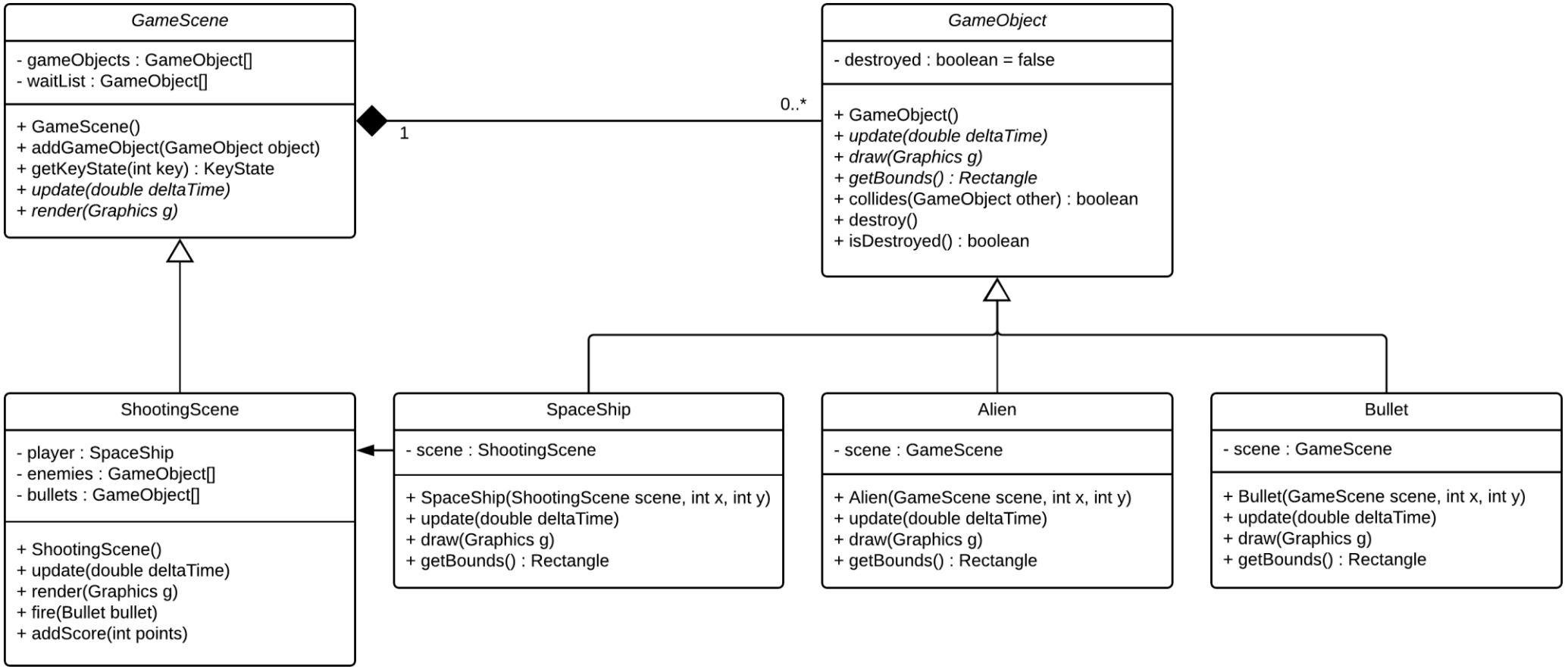
เกม

- โปรแกรมประเภทหนึ่งที่มีหลักการทำงานพื้นฐานคือ **“การประมวลผลเป็นเฟรม”**
  - เฟรม คือ ภาพนิ่งแต่ละภาพที่แสดงต่อเนื่องกันจนดูเหมือนภาพเคลื่อนไหว
- งานที่เกิดขึ้นในแต่ละเฟรม
  1. ตรวจสอบอินพุต
  2. อัปเดตสถานะของวัตถุต่าง ๆ ภายในเกมและข้อมูลของตัวเกม
  3. วาดเฟรม ซึ่งประกอบด้วยฉาก และวัตถุที่เป็นองค์ประกอบของเกม เช่น คะแนน สถานะของตัวละคร
- งานทั้ง 3 ข้อ จะเกิดขึ้นซ้ำ ๆ ตราบเท่าที่โปรแกรมยังทำงาน โดยทิ้งช่วงการเกิดขึ้นของกลุ่มงาน เท่า ๆ กัน (game loop)

เกม (ต่อ)

- จำนวนเฟรมที่เกิดขึ้นภายในหนึ่งหน่วยเวลา มีผลต่อความรู้สึกของผู้เล่น ซึ่งผู้เล่นต้องการให้เกิดเฟรมจำนวนมากในหนึ่งหน่วยเวลา (ลื่นนนน)
  - ปัญหาคือการประมวลผลที่เกิดขึ้นในหนึ่งเฟรมต้องใช้ทรัพยากร
- การควบคุมจำนวนเฟรมต่อหน่วยเวลาแบ่งเป็น 2 รูปแบบ
  - **จำนวนเฟรมต่อวินาทีคงที่** คือ การกำหนดจำนวนเฟรมที่ต้องประมวลผลในหนึ่งวินาที ทำให้การคำนวณการเคลื่อนที่ของวัตถุง่าย แต่ไม่ยืดหยุ่น ใช้ประโยชน์จากทรัพยากรไม่เต็มที่
  - **จำนวนเฟรมต่อวินาทีไม่คงที่** คือ การประมวลผลเฟรมถัดไปในทันทีเมื่อประมวลผลเฟรมก่อนหน้าเสร็จ ทำให้จำนวนเฟรมที่ประมวลผลในหนึ่งวินาทีไม่เท่ากัน การคำนวณการเคลื่อนที่ในเกมอิงกับเวลาที่ผ่านไปนับจากเฟรมก่อนหน้า เรียกว่า “**Delta time**”

ตัวอย่างเกมอวกาศ (SpaceGame)



กำหนดวิธีจัดการฉากหลังของเกม

```
public enum KeyState { UP, DOWN }

private KeyState[] keyStates = new KeyState[5];

private static final int DEFAULT_FRAME_RATE = 50;
private long previousTime;

private ArrayList<GameObject> gameObjects = new ArrayList<>();
private ArrayList<GameObject> waitList = new ArrayList<>();

public GameScene() {
    this(DEFAULT_FRAME_RATE);
}

public GameScene(int frameRate) {
    setBackground(Color.BLACK);
    setFocusable(true);
    setDoubleBuffered(true);
```

- คลาส GameScene เป็นคลาสนามธรรม ที่ใช้เป็นต้นแบบของฉาก ประกอบด้วย
  - การตั้งค่าที่จำเป็นในการเป็นตัวต้นแบบของฉาก เช่น รายการของวัตถุ, จำนวนเฟรมในหนึ่งวินาที, ปุ่มและสถานะของปุ่มที่ใช้ควบคุม เป็นต้น
  - กำหนดให้พาเนลรับโฟกัสได้ (สำหรับการรับอินพุตจากแป้นพิมพ์)
  - เปิดการใช้งาน double buffering (setDoubleBuffered(true)) ซึ่งทำให้พาเนลมีการวาดโดยใช้บัฟเฟอร์คู่ (สลับกันทำงานระหว่างตัวแสดงผลและตัววาดรูป)

กำหนดวิธีจัดการฉากหลังของเกม (ต่อ)

```
addKeyListener(new KeyAdapter() {
    @Override
    public void keyPressed(KeyEvent e) {
        switch (e.getKeyCode()) {
            case KeyEvent.VK_UP:
                keyStates[KEY_UP] = KeyState.DOWN;
                break;
            case KeyEvent.VK_DOWN:
                keyStates[KEY_DOWN] = KeyState.DOWN;
                break;
            case KeyEvent.VK_LEFT:
                keyStates[KEY_LEFT] = KeyState.DOWN;
                break;
            case KeyEvent.VK_RIGHT:
                keyStates[KEY_RIGHT] = KeyState.DOWN;
                break;
            case KeyEvent.VK_SPACE:
                keyStates[KEY_FIRE] = KeyState.DOWN;
                break;
        }
    }

    @Override
    public void keyReleased(KeyEvent e) {···
});
```

- ปรับเพียงสถานะของปุ่มกด แต่ยังไม่ทำให้วัตถุเคลื่อนไหว ซึ่งการเคลื่อนที่ปล่อยให้เป็นหน้าที่ของตัววัตถุเอง

```
@Override
public void keyReleased(KeyEvent e) {
    switch (e.getKeyCode()) {
        case KeyEvent.VK_UP:
            keyStates[KEY_UP] = KeyState.UP;
            break;
        case KeyEvent.VK_DOWN:
            keyStates[KEY_DOWN] = KeyState.UP;
            break;
        case KeyEvent.VK_LEFT:
            keyStates[KEY_LEFT] = KeyState.UP;
            break;
        case KeyEvent.VK_RIGHT:
            keyStates[KEY_RIGHT] = KeyState.UP;
            break;
        case KeyEvent.VK_SPACE:
            keyStates[KEY_FIRE] = KeyState.UP;
            break;
    }
}
```

กำหนดวิธีจัดการฉากหลังของเกม (ต่อ)

- ตั้งเวลาสำหรับเปลี่ยนฉากและวัตถุบนฉาก พร้อมกับเวลาที่บอกได้ว่า เวลาผ่านไปเท่าใด แล้วหลังจากเฟรมก่อนหน้า (delta time)

```
previousTime = System.currentTimeMillis();

Timer timer = new Timer(1000/frameRate, new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        updateAll();
    }
});

timer.start();
```

```
Previous time: 1582591301623
Current time: 1582591301644
Delta time: 0.0209999999716877937
Previous time: 1582591301644
Current time: 1582591301664
Delta time: 0.0199999999552965164
Previous time: 1582591301664
Current time: 1582591301685
Delta time: 0.0209999999716877937
```

```
private void updateAll() {
    long currentTime = System.currentTimeMillis();
    double deltaTime = (currentTime - previousTime) / 1000.0f;
    previousTime = currentTime;

    // Call update() on game scene and all game objects
    update(deltaTime);
    for (GameObject gameObject : gameObjects) {
        gameObject.update(deltaTime);
    }
    // Prune dead game objects
    Iterator<GameObject> it = gameObjects.iterator();
    while (it.hasNext()) {
        GameObject gameObject = it.next();
        if (gameObject.isDestroyed()) {
            it.remove();
        }
    }
    // Add newly queued game objects
    for (GameObject object : waitList) {
        gameObjects.add(object);
    }
    waitList.clear();

    repaint();
}
```

กำหนดวิธีจัดการฉากหลังของเกม (ต่อ)

```
private void updateAll() {
    long currentTime = System.currentTimeMillis();
    double deltaTime = (currentTime - previousTime) / 1000.0f;
    previousTime = currentTime;

    // Call update() on game scene and all game objects
    update(deltaTime);
    for (GameObject gameObject : gameObjects) {
        gameObject.update(deltaTime);
    }
    // Prune dead game objects
    Iterator<GameObject> it = gameObjects.iterator();
    while (it.hasNext()) {
        GameObject gameObject = it.next();
        if (gameObject.isDestroyed()) {
            it.remove();
        }
    }
    // Add newly queued game objects
    for (GameObject object : waitList) {
        gameObjects.add(object);
    }
    waitList.clear();

    repaint();
}
```

- หากวัตถุชนกัน (Alien กับ Bullet) วัตถุสองตัวนั้นจะถูกกำหนดสถานะให้ลบออกในเฟรมถัดไป
- ฉากจึงมีหน้าที่ตรวจสอบว่าวัตถุใดจะไม่ถูกวาดเมื่อมีการสั่งวาดเฟรม
  - for loop ปกติไม่สามารถลบ object ใน ArrayList ที่กำลังถูกจัดการอยู่ได้ จึงต้องใช้ Iterator
- วัตถุที่จะเกิดขึ้นในฉากถัดไปจาก waitList ถูกโหลดใส่ gameObject list

การสร้างฉากโดยอาศัยฉากต้นแบบ (ShootingScene)

```
@Override
public void update(double deltaTime) {
    for (GameObject bullet : bullets) {
        for (GameObject enemy : enemies) {
            if (bullet.collides(enemy)) {
                bullet.destroy();
                enemy.destroy();
                addScore(10);
                respawnCount++;
            }
        }
    }
    while (respawnCount > 0) {
        Alien alien = new Alien(this, 100 +
        random.nextInt(700), 100 + random.nextInt(200));
        enemies.add(alien);
        addGameObject(alien);
        respawnCount--;
    }
}
```

- Override abstract method
  - update
  - render()
  - ต้นแบบระบุให้เรียก update() ก่อน render() เสมอ
- เพิ่มเมทอดที่นอกเหนือจากต้นแบบ
  - fire()
  - addScore()

กำหนดวิธีจัดการวัตถุใด ๆ ในเกม (ต่อ)

```java
public abstract class GameObject {
    private boolean destroyed = false;

    public GameObject() {
        destroyed = false;
    }

    public abstract void update(double deltaTime);

    public abstract void draw(Graphics g);

    public abstract Rectangle getBounds();

    public boolean collides(GameObject other) {
        if (isDestroyed() || other.isDestroyed()) {
            return false;
        } else {
            return getBounds().intersects(other.getBounds());
        }
    }

    public void destroy() {
        destroyed = true;
    }

    public boolean isDestroyed() {
        return destroyed;
    }
}
```

- คลาสนามธรรม GameObject ใช้เป็นต้นแบบของวัตถุทุกประเภทของเกม เช่น SpaceShip, Alien, Bullet
- เมทอด update ใช้สำหรับปรับสถานะของวัตถุ เช่น ตำแหน่งของยานแม่ในเกมถูกกำหนดใน เมทอด update
- เมทอด draw ใช้สำหรับสร้างอ็อบเจกต์ชนิด Graphics เพื่อ add ลง panel เช่น
  - draw() ในคลาส Alien ทำหน้าที่สร้างรูปทรงสี่เหลี่ยมสีน้ำเงิน
- เมทอด collides ทำหน้าที่ตรวจการชนกันของวัตถุ เช่น Bullet กับ Alien ถ้ามีพื้นที่ซ้อนทับ (intersect) กันให้ return true

การสร้างวัตถุจากต้นแบบ

```java
public Alien(GameScene scene, int x, int y) {
    this.scene = scene;
    this.x = x;
    this.y = y;
}
```

```java
public void update(double deltaTime) {
    if (x < 100) {
        speedX = 10;
    } else if (x > 700) {
        speedX = -10;
    }
    if (y < 100) {
        speedY = 10;
    } else if (y > 300) {
        speedY = -10;
    }
    x += speedX;
    y += speedY;
}

@Override
public void draw(Graphics g) {
    g.setColor(Color.BLUE);
    g.fillRect(x, y, 50, 50);
}

@Override
public Rectangle getBounds() {
    return new Rectangle(x, y, 50, 50);
}
```

- ที่ constructor ของวัตถุต้องกำหนดว่าเป็นองค์ประกอบของฉากใด
- Implement abstract method ตามพฤติกรรมของตัววัตถุเอง
  - update ในตัวอย่างคือ ตำแหน่งที่จะนำไปสร้าง Alien ในแต่ละฉาด
  - draw ในตัวอย่างคือ รูปทรงของ Alien

อ้างอิง

- https://nbviewer.jupyter.org/github/Poonna/java-book/