Windows (GUI) & Applet

Vittayasak Rujivorakul

Lecture 11

Windows (GUI) & Applet

Vittayasak Rujivorakul

Lecture 11

Graphics User Interface

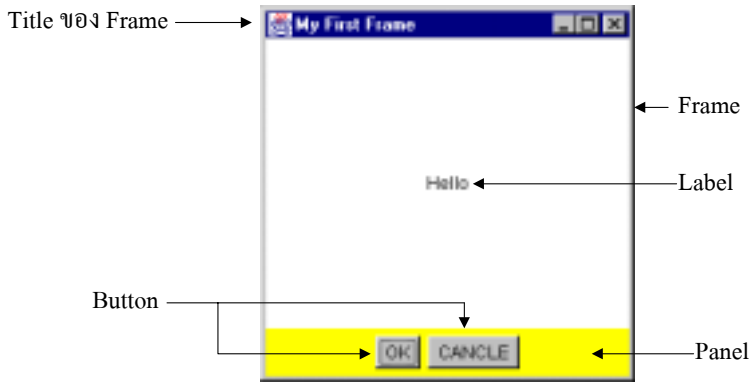
- โปรแกรมสไตล์ที่ใช้กับ **GUI Object** : event-driven programming
- ตัวอย่าง GUI อย่างง่าย (MyFrame.java)
  - สร้าง Window, 2 ปุ่ม คือ OK และ Cancel
  - ถ้ากดปุ่ม OK ขึ้น Label ว่า “You clicked OK”
  - ถ้ากดปุ่ม CANCEL ขึ้น Label ว่า “You clicked CANCEL”

MyFrame.java

```
import java.awt.*;
import java.awt.event.*;
class MyFrame {
   // data members
   private Button okB, cancleB;
   private Label label;
   private Frame fr;
   public static void main(String args[]) {
       MyFrame mF = new MyFrame();
       mF.init();
   }

  // initialize
  public void init() {
       fr = new Frame("My First Frame");
       lable = new Label
                   ("Hello",Label.CENTER);
       fr.add(label,BorderLayout.CENTER);
      okB = new Button("OK");
      cancleB = new Button("CANCLE");
      Panel panel = new Panel();
      panel.setBackground(Color.yellow);
      panel.add(okB);
      panel.add(cancleB);
      fr.add("South",panel);
      fr.pack();
      fr.setVisible(true);
   }
}
```


Components และ Containers



ส่วนประกอบหลักของ GUI

- Container : ที่บรรจุ
- Component : ส่วนประกอบ
- Layout Manager : ตัวจัดการวางตำแหน่งและขนาดของ components

Containers

- Containers มี 2 ชนิด
  - **Windows** (java.awt.Window) : มี 2 ชนิด
    - Frane : เป็น window ที่มี title และ resize comer
    - Dialog : เคลื่อนย้ายได้ แต่เปลี่ยนขนาดไม่ได้
  - **Panels** (java.awt.Panel) : เป็น intermetiate container หรือ container ที่ไม่มีกรอบ ซึ่งต้องวางไว้อยู่ใน container อื่น (เช่น Frame) เพื่อใช้ช่วยในการวาง layout
    - กำหนดพื้นที่สี่เหลี่ยมไว้ในตัว container

การวาง Layout

- โดยให้ Layout Manager เป็นตัวกำหนดตำแหน่งและขนาดของ components ที่อยู่ใน Container
- ทำการควบคุมขนาดหรือตำแหน่งของ components เอง โดยไม่ใช้ Layout Manager (ทำให้ Layout Manager ไม่ทำงาน) :
  - setLayout(Null)
  - จากนั้นใช้ setLocation ( ), setSize หรือ setBounds ในตัว components เพื่อที่จะกำหนดตำแหน่งต่าง ๆ ที่จะวางใน Container เอง

Frame

- **inherit มาจาก Component** โดยเป็น subclass ของ Window
- มี **title** และ **resize** corners
- default Layout Manager คือ **BorderLayout**
- ทำการเพิ่ม components ได้โดยใช้ **add method**
- ใช้ **setLayout method** เพื่อทำการเปลี่ยน default Layout Manager
- **สามารถสร้าง Frame ก่อนจะ set** ให้มองเห็นได้

Panel

- **เป็น container** เตรียมที่ไว้ให้ components คล้ายกันกับ Frame
- แต่จะแสดงให้เห็นได้ **Panel ต้องถูกเพิ่ม (add) ไว้ใน Window หรือ Frame**
- panel แต่ละอัน**สามารถมี Layout Manager เป็นของตนเองได้**
- ทำการเพิ่ม components อื่นๆ ได้โดยใช้ **add method**

Container Layouts

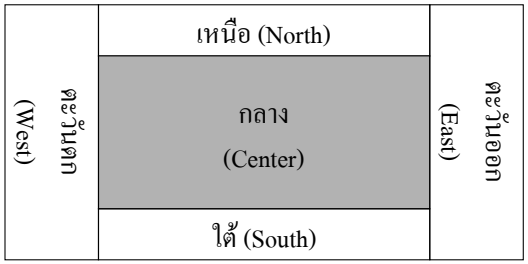
- **Layout มีหลายแบบ เช่น**
  - **Flow Layout:** default ของ Panels และ Applets
  - **Border Layout:** default ของ Windows, Dialogs และ Frames
  - **Grid Layout:** แบ่งหน้าต่างออกเป็นตารางย่อยที่มีขนาดเท่ากัน
  - **Gard Layout:** วาง component ภายในเหมือนสำรับไพ่เปิดได้ทีละแผ่น
  - **GridBag layout:** คล้าย Grid Layout แต่ตารางย่อยแต่ละ cell มีขนาดไม่เท่ากันได้
  - **Box Layout:** จัดวางตำแหน่ง component ไว้หนึ่งแถว หนึ่ง column

FlowLayout

- **จัดเรียง components ในลักษณะบรรทัดต่อบรรทัด** เมื่อเต็มบรรทัดแล้วจึงขึ้นบรรทัดใหม่
- FlowLayout **ไม่เปลี่ยนขนาดของ component** ที่อยู่ในมัน
- สามารถจัดให้เรียงชิด**ซ้าย ขวา** หรือ**ตรงกลาง**ได้
- สามารถ**ระบุช่องไฟ**ระหว่าง components ได้
- **ตำแหน่ง**ของวัตถุอาจมีการเปลี่ยนแปลงได้ แต่**ขนาด**ของวัตถุไม่เปลี่ยนหากมีการขยาย-หดพื้นที่

BorderLayout

- แบ่งพื้นที่ออกเป็น 5 ส่วนย่อย คือ
  - เหนือ, ใต้, ตะวันออก, ตะวันตก วางตามแนวขอบของหน้าต่าง
  - และตรงกลาง ซึ่งจะเป็นพื้นที่ที่เหลือของหน้าต่าง
- components ภายในจะยืด-หดตามขนาดของหน้าต่าง



GridLayout

- จัดแบ่งพื้นที่ออกเป็นตารางย่อยขนาดช่องเท่ากัน

  GridLayout (int row, int column)

- ทำการวาง components โดยการ add
- การวางจะเรียงจากซ้ายไปขวา บนลงล่าง ; components ภายในเปลี่ยนขนาดเป็นสัดส่วนเดียวกับขนาดของหน้าต่าง

GridLayout(4,3)

CardLayout

- **ทำการเพิ่ม components ได้โดยการ add**
  - อาจใช้ add แบบมี constraint

  add (String, Component) หรือ add (Component, Object)

  frame.add (“Hello”, new Label (“How are you?”) ;

  frame.add (“OK”, new Button (“OK”)) ;
- **ทำการพลิกแต่ละ card ได้โดย**

  public void first (Container parent)

  public void next (Container parent)

  public void previous (Container parent)

  public void last (Container parent)

  public void show (Container parent, String name)

การวาง GUI Object ในตำแหน่งที่ระบุแน่นอน

- เราสามารถที่จะระบุในตำแหน่งที่แน่นอนได้ในลักษณะ **absolute positioning** โดย
  - setLayout (null) ; / / เพื่อทำให้ไม่มี layout manager
  - <object>.setBound (<x>, <y>, <width>, <height>) ;

    / / วางตำแหน่งและขนาด

Note : เมื่อ set ให้ไม่มี LayoutManager (to null) แล้ว การ resize window ตำแหน่งของวัตถุจะไม่เปลี่ยน

Absolute Positioning

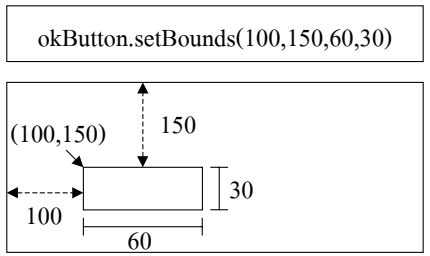
setLayout (null) ; / / no layout manager

okButton.setBounds (100, 150, 60, 30) ;

cancelButton.setBounds (170, 150, 60, 30) ;

add (okButton) ;

add (cancelButton) ;



Components

- Canvas
- Button
- Label
- TextField
- TextArea
- CheckBox
- Choice
- List
- TabbedPane
- JButton
- JLabel
- JComboBox
- JMenu

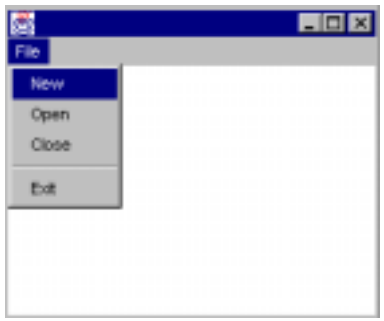
Components กับการวาดภาพ

- **Paint (Graphics)**
  - ทำการวาดภาพ (Graphics) ถูกเรียกโดยอัตโนมัติ ทำการ update ซ่อมส่วนที่อาจขาดหายไป (เช่นเมื่อมีหน้าต่างอื่นบัง, เปิดขึ้นใหม่๗
- **repaint ( )**
  - บอกระบบให้ ทำการ schedule การ update ครั้งต่อไปทันทีที่เป็นไปได้ เพื่อทำการเปลี่ยนภาพบนหน้าต่าง
- **update (Graphics)**
  - ทำการล้างหน้าต่างแล้วเรียก paint เพื่อให้วาดใหม่

การสร้าง Menus

- ลำดับการสร้างและเพิ่ม menus
  - สร้าง MenuBar object แล้วบรรจุใน Frame
  - สร้าง Menu object
  - สร้าง MenuItem object และบรรจุใน Menu object
  - บรรจุ Menu object ลงใน MenuBar object

ตัวอย่าง : MyMenu.java




```
import java.awt.*;
class MyMenu extends Frame{
  MyMenu() {
    MenuBar mb = new MenuBar();
    setMenuBar(mb);
    Menu m = new Menu("File");
    mb.add(m);
    m.add(new MenuItem("New"));
    m.add(new MenuItem("Open"));
    m.add(new MenuItem("Close"));
   m.addSeparator();
   // or use m.add(new MenuItem("-"));
   m.add(new MenuItem("Exit"));
   setSize(150,120);
   setVisible(true);
  }
  public static void main(String args[]){
    new MyMenu();
  }
}
```

การจัดการกับ event

- อาจทำได้ใน 3 ลักษณะคือ
  - โดยประกาศให้ class ทำการ implements Listeners (ถ้ามีหลายตัวใช้ comma คั่นแบ่ง) และ บรรจุ Event Handlers ไว้เป็น methods ของ class
  - โดยการสร้าง anonymous class เพื่อทำการจัดการกับ event
  - ทำการสร้าง class ใหม่ในการจัดการกับ event

Event Handler โดยทำการ implement interface

- **การ Setup Event Listener มี 4 ขั้นตอน**
  - **import Java Event Handling Class โดย**

    import java.awt.event. * ;
  - **เปลี่ยน class declaration** โดยเพิ่ม implement interface ที่ต้องการใช้

    public class MyFrame implement ActionListener
  - **เพิ่ม method ที่ต้องทำการ implement** (I.e., actionPerformed)
  - **register object** (I.e.,botton) ให้เป็น action event listener ของ event source

Event Handler : Anonymous class

- Setup Event Listener มี 3 ขั้นตอน
  - import Java Event Handling Class โดย

    import java.awt.event. * ;
  - ทำการ register object (I.e., frame) ให้เป็น action event listener ของ event source
  - ทำการเพิ่ม method ที่ต้องทำการ implement (I.e., windowClosint) ในลักษณะ Anonymous class

ตัวอย่าง Anonymous Class

- **สร้าง anonymous class** เพื่อเป็น event handle ของการปิด window เมื่อมีการ click ปุ่มปิดของหน้าต่าง

```
Frame frame = new Frame (“My Frame”) ;
frame.addWindowListener ( new WindowAdapter( ) {
        / / handle the window closing event
        public void windowClosing (WindowEvent e) {
                / /  deallocate frame from memory
                frame.dispose ( ) ;
                / /  cal  system for exit
                System.exit (0) ;
        }
}) ;
```

การสร้าง class ใหม่เพื่อจัดการกับ events

- **การ Setup Event Listener** มีขั้นตอนดังนี้
  - Import Java event Handling Class (import java.awt.event. * ;)
  - register object ให้เป็น action event listener ของ event source

    addWindowListener (new Program Terminator ( )) ;
  - สร้าง class สำหรับ handler โดยการ extends event class adapter หรือ implements interface ที่ต้องการใช้ และ implements methods

```
class ProgramTerminator extends WindowAdapter   {
   public void windowClosing (WindowEvent event)   {
            System.exit (0) ;
   }
}
```


การจับคู่ Listener Interface กับ component

- เมื่อ**จับคู่ component กับ event** ที่มันสามารถ support ได้ เราสามารถจะ**ทำ event handler** โดย
  - นำชื่อของ event class เปลี่ยนคำว่า Event เป็น Listener จะได้ Listener interface ที่เราจะต้อง implements
  - implement interface และ methods ที่เราต้องการให้จับ event นั้น ๆ
  - สร้าง object ของ listener class ข้างต้น แล้วทำการ register มันเข้ากับ component โดยเรียก method “add” ตามด้วยชื่อของ listener ข้างต้น

    (เช่น frame.addWindowListener(new WindowListener ( ) { . . .}) ;)

Graphics

- **การวาดภาพใน Java** เกิดใน Graphics object (java.awt.Graphics)
  - Graphics แต่ละอันจะมี coordinate เป็นของตัวเอง
  - method เพื่อใช้ในการวาด string, line, rectangle, circle, polygon, agc และอื่น ๆ

Applet Declaration

- Syntax :

public class <applet subclass> **extend**s Applet

{

<class member declaration>

}

  - **applet subclass** ต้อง declare **เป็น public class**
  - **extends** เป็น reserved word ที่ใช้สำหรับ ความสัมพันธ์แบบ **inheritance**

ตัวอย่างเทมเพลทสำหรับโปรแกรม Applet อย่างง่าย

```
//Comment
import java.applet. * ;
import java.awt. *;
import statements
public class subclass name extends Applet
{
        public void init ( )
        {
                method body
        }
}
```


ตัวอย่าง html file

```
<HTML>
<BODY>
<APPLET CODE = "MyFirstApplet.class"
        HEIGHT = 190 WIDTH = 300 >
< / APPLET>
< / BODY>
< / HTML>
```


Method ใน Applet

Method	หน้าที่
init()	ถูกเรียกเมื่อ applet ถูกสร้างและ loaded เป็นครั้งแรก เพื่อทำการ initialization
start()	ถูกเรียกทุกครั้งเมื่อ applet ปรากฏขึ้นบนหน้าจอของ browser เพื่อทำการเริ่มต้นการทำงาน (esp. หลัง stop() หรือหลังจาก init())
stop	ถูกเรียกทุกครั้งเมื่อเปลี่ยนหน้าจอของ browser หรือหลัง destroy() เพื่อทำการหยุด operations ของ applet
destroy()	ถูกเรียกเมื่อ unloaded จากหน้าจอของ browser เพื่อปล่อย resource ก่อน applet จะยุติการทำงาน
paint(Graphics)	ถูกเรียกเมื่อ browser ต้องทำการ refresh การแสดงจอภาพ applet เช่น เมื่อมีการ restore จอภาพจากการ iconify

การบรรจุ applet ใน JAR file

- ทำการ jar applet โดย

  jar cf jarName.jar *.class

- จากนั้นเขียน html file

```
<HTML>
<BODY>
<APPLET CODE = "MyFrame.class"
        ARCHIVE = "jarName.jar"
        HEIGHT = 190 WIDTH = 300 >
< / APPLET>
< / BODY>
< / HTML>
```

appletviewer

- **Java application ซึ่งสามารถ run applet ได้โดยไม่ต้องอาศัย web browser**
- ทำการ load applet ได้โดย
  - appletviewer MyApple.html หรือ
  - appletviewer MyApplet.java **(ถ้าใน java file นั้นมี tag ของ <APPLET CODE …> เป็น comment อยู่)**

Applet Tag

<applet

[archive = archiveList]	preloaded class ถ้ามีมากกว่า 1 คั่นด้วย ,
code=appletName.class	ชื่อของ class
width=pixels height=pixels	ความกว้างและความสูงของหน้าต่าง applet
[codebase=codebaseURL]	directory ที่เป็นที่อยู่ของ applet ถ้าไม่ระบุใช้ URL
[alt=alterText]	คำที่แสดงในกรณีที่ browser ไม่สามารถ run applet ได้
[name=appletInstancdName]	ชื่อเพื่อใช้อ้างถึงในกรณีมีหลาย applet ในหน้าเดียวกัน
[align=alignment]	จัดวาง applet ตามที่ระบุของหน้ากระดาษ
[vspace=pixels] [hspace=pixels]	ช่องไฟ

>

[<param name=attribute1 value=val1>]	ค่าของ parameter ที่กำหนดให้จากภายนอก
[<param name=attribute2 value=val2>]	เรียกใช้ได้โดย getParameter()
[<param name=….>]	

</applet>

การส่งผ่าน parameter ให้กับ applet

```
import java.applet.*;
import  java.awt.*;
public class DrawStringApplet extends Applet{
   public void paint(Graphics g) {
      String inputFromPage =  getParameter
     ("Message");
      g.drawString(inputFromPage, 50, 25);
   }
}
```

```
<HTML>
<BODY>
<APPLET>
code="DrawStringApplet.class"
width="300"  height="50">
<PARAM name="Message"
          value="Hello world!">
</APPLET>
</BODY>
</HTML>
```


การหาขนาดของ applet

```
import java.applet. * ;
import java.awt. * ;
/ *
        <applet code = SizeApplet.class width = 300 height = 300>
        < / applet>
* /
public class sizeApplet extends Aplet {
        public class SizeApplet extends Applet   {
                Dimension appletSize  =  getSize ( )  ;
                int appletHeight  =  appletSize.height ;
                int appletWidth  =  appletSize.width ;
                g.drawString ("This applet is"  +  appletHeight ;
                        "pixele high by "  + appletWidth,  "pixelsWidth"
                        15, appletHeight / 2) ;
        }
}
```

Applet Security

- **Applet สามารถที่จะ**
  - วาดภาพบน web page
  - **สร้างหน้าต่างใหม่**และวาดภาพในหน้าต่างนั้น
  - เล่น**เสียง** (เช่น wave file)
  - **รับ input** จาก ผู้ใช้ผ่าน keyboard หรือ mouse
  - ทำการติดต่อ **รับ/ส่งข้อมูลไป - กลับ**ยัง server ที่เป็น source ของ applet นั้น
- Note : Java applet windows จะเขียนคำเตือนเมื่อมีการ load applet "Warning : Applet Window" or "Unsigned Java Applet Window"

Applet ไม่สามารถที่จะ

- **ลบหรือเขียนข้อมูล**ลงในดิสก์ของเครื่องที่ load applet มา (host machine)
- **อ่านข้อมูลจาก ดิสก์ของ host** โดยไม่ได้รับอนุญาต ในบาง environments, เช่นใน Netscape ไม่อนุญาตให้อ่าน user's disk เลย
- **อ่านหรือเขียนจาก หน่วยความจำ** การ access หน่วยความจำ จะอยู่ภายใต้ ความดูแลของ **JVM**
- ทำการติดต่อไปยัง server ใด ๆ บน Internet **นอกจากแหล่งที่มันมา**
- ทำการ**เรียก native API โดยตรง** (Though Java API calls may eventually lead back to native API calls)
- นำ **virus** หรือ **trojan horse** เข้ามาใน host system
- ไม่สามารถที่จะ **crash ระบบ** (ในทางปฏิบัติ Java นั้นเองยังไม่เสถียร)

Applet Life Cycle

- browser **อ่าน HTML** page และพบ **<APPLET> tage**
- browser แยก <APPLET> tag เพื่อหา **CODE** และ **CODEBASE parameter**
- browser **download class file** ที่ระบุจาก URL ที่พบครั้งสุดท้าย
- browser **แปลง** raw bytes ใน Java class file
- browser สร้าง **applet object** จาก default constructor ของ class นั้น
- browser เรียก **init ( )** method
- browser เรียก **start ( )** method
- ในขณะ run applet, browser ส่ง **events** เช่น mouse clicks, key presses เพื่อให้ **event handler** จัดการ ซึ่ง Update event อาจจะบอก applet ให้ทำการ repaint ตัวเอง
- browser เรียก **stop ( )** method
- browser เรียก **destroy ( )** method

Applet กับ image

```
/ * <applet code = AppletImage.class width = 100 height = 100>
    <param name = imagefile value="test.gif">
    < / applet>
* /
import java.awt. * ;
import java.applet. * ;
public class AppletImage extends Applet  {
    Image image ;
    public void init ( )  {
        String filename = this.getParameter ("imagefile") ;
        image = this.getImage (getDocumentBase ( ), filename) ;
    }
    public void paint (Graphics g)  {
        g.drawImage (image, 0, 0, this) ;
    }
}
```

Applet กับเสียง

```
/ * <applet code = AppletAudio.class width = 100 height = 100>
    <param name = audiofile value = >
    < / applet >
* /
import java.awt. * ;
import java.applet. * ;
public class AppletAudio extends Applet   {
    AudioClip sound ;
    public void init ( )   {
        String filename = this.getParameter ( “audiofile”) ;
        audio = this. getAudioClip (getDocumentBase ( )  filename) ;
    }
    public void paint (Graphics g)   {
        g.drawString (“Audio Test”, 50, 50) ;
        this.play (sound) ;
    }
}
```


การรวม application กับ applet

```
import  java.applet.*;
import  java.awt.*;
import  java.awt.event.*;
public class MyAppApplet extends Applet {
    Button b1, b2;
    TextField tf = new TextField(20);
    ActionListener al = new
      ActionListener() {
        public void actionPerformed
    (ActionEven e) {
            String name = ((Button)
    e.getSource( )).getLabel( );
            tf.setText(name);
     } };
   public void init( ) {
     b1 = new Button(“B1”);
     b2 = new Button(“B2”);
     b1.addActionListener(al);
     b2.addActionListener(al);
     add(b1); add(b2); add(tf);
   }
   public static void main(String args[]) {
     Applet applet=new MyAppApplet( );
     Frame fr = new Frame(“My
     Application/Applet”);
     fr.addWindowListener(
       new WindowAdapter( ) {
         public void windowClosing
     (WindowEvent e) {
         System.exit(0);
     }
   });
   fr.add(applet);
   fr.setSize(400,100);
   applet.init( );
   applet.start( );
   fr.setVisible(true);
 } }
```