Command-Line Interface (CLI) (基于文本)

↓

Graphical User Interface (GUI) (基于图形)

Abstract Windowing Toolkit (AWT): JDK 1.0 (platform-dependent)

调用的是操作系统中高层的方法

Swing: JDK 1.1 (platform-independent) (light-weight)

完全由Java写成，支持组件更丰富。调用的是操作系统中底层的方法

Java FX: JDK 8

SWT: Eclipse 使用，归 Eclipse 管理

Swing classes (package javax.swing)

容器装组件，容器也可装容器

(因为容器本身也算是组件)

Containers (容器)

top-level container: (顶层容器)

- JFrame: used for the applicant's main window

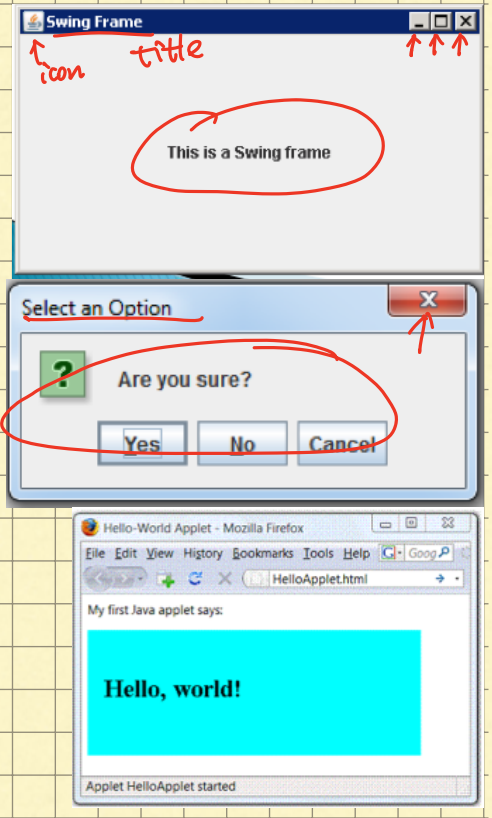
  GUI 以 JFrame 为基础，是屏幕上 window 的对象，能最大、最小化和关闭。

- JDialog: used for secondary pop-up window

  有标题、关闭、内容面

- JApplet: used for the applet's display-area (content-plane) inside a browser's window

Secondary containers can be used to group and layout relevant components.



```java
import javax.swing.JFrame;

public class HelloWorld extends JFrame {
    public HelloWorld() {
        super("Our first Swing program");
    }

    public static void main(String[] args) {
        HelloWorld gui = new HelloWorld();
        gui.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        gui.setSize(800, 600);
        gui.setVisible(true);
    }
}
```

Select a top-level container (mostly JFrame)

①选择顶层容器

→点击X按钮后，窗口关闭，程序结束

②设置点击窗口关闭页面后的行为

Exit the application (process) when the close button is clicked.
Default value HIDE_ON_CLOSE hides the JFrame, but keeps the application running.

Display the JFrame

③显示GUI

若不定义，则当关闭窗口后，程序仍然运行

```java
public class HelloWorld extends JFrame {

    private JLabel label;

    public HelloWorld() {
        super("Our first Swing program");
        setLayout(new FlowLayout());
        label = new JLabel("Hello World");
        label.setFont(new Font("San Serif", Font.PLAIN, 30));
        add(label);
    }

    public static void main(String[] args) { // same as earlier }
}
```

Declaring GUI components as fields makes it easier to interact with the corresponding objects (no need to retrieve references from the container)

Specifying layout (how to position GUI components)

流式布局

设置字体

Creating GUI component (a label here) and add it to the JFrame (actually its content pane)

当设置组件后该如何排放

在给容器内添加一些主要组件的时候，最好在类当中创建一个private field来保存这些组件的引用。

GUI-Based Input/Output 基于GUI的输入/输出

JOptionPane: pop up a dialog box that prompts users for a value or informs them of sth.

静态方法：showInputDialog() //输入　　showMessageDialog() //输出

```java
public static void main(String[] args) {
    String str1 = JOptionPane.showInputDialog("Enter 1st integer");
    String str2 = JOptionPane.showInputDialog("Enter 2nd integer");
    int num1 = Integer.parseInt(str1);
    int num2 = Integer.parseInt(str2);
    int sum = num1 + num2;
    JOptionPane.showMessageDialog(null, num1 + " + " + num2 + " = " + sum);
}
```

prompts for user input

"123" will be read as a string

null will be read

```java
public static void main(String[] args) {
    String str1 = JOptionPane.showInputDialog("Enter 1st integer");
    String str2 = JOptionPane.showInputDialog("Enter 2nd integer");
    int num1 = Integer.parseInt(str1);
    int num2 = Integer.parseInt(str2);
    int sum = num1 + num2;
    JOptionPane.showMessageDialog(null, num1 + " + " + num2 + " = " + sum);
}
```

Static method showMessageDialog()
tells user about something that has happened

Event-driven : GUI Programming

Event Handling (事件处理机制) : delegation event model (委托事件模式)

Event source (事件源) : 与用户交互的GUI组件

Event object (事件对象) : 封装了事件的相关信息的对象

Event listener (事件监听器) : 当某事件发生时, 事件源会提醒该对象

↓

其中的方法会接收到事件相关的一系列信息 (被封装到event object中)

↓

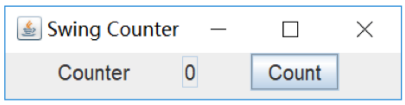
然后处理这些信息并作出反应

计数器:

```
public class SwingCounter extends JFrame {
    private JTextField tfCount;→计数
    private JButton btnCount;→按钮
    private int count = 0;
    public SwingCounter() {
        setLayout(new FlowLayout(FlowLayout.LEFT, 50, 0));
        add(new JLabel("Counter"));→加到frame中,并给名
        tfCount = new JTextField("0");→初始化文本
        tfCount.setEditable(false); add(tfCount);
        btnCount = new JButton("Count"); add(btnCount);→加到frame中
    }
    public static void main(String[] args) { SwingCounter sc = new SwingCounter(); ... }
}
```

文本不可改动

未设置事件监听器



Nothing will happen when we click the button (we have not handled the event yet)

改良

```
public class SwingCounter extends JFrame {
    private JTextField tfCount;
    private JButton btnCount;
    private int count = 0;
    public SwingCounter() {
        setLayout(new FlowLayout(FlowLayout.LEFT, 50, 0));
        add(new JLabel("Counter"));;
        tfCount = new JTextField("0");
        tfCount.setEditable(false); add(tfCount);
        btnCount = new JButton("Count"); add(btnCount);
        btnCount.addActionListener(new ButtonClickListener());
    }
    public class ButtonClickListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent arg0) {
            count++; tfCount.setText(count + "");
        }
    }
    public static void main(String[] args) { … }
}
```

Event source

Event listener

(未显式调用,但实际上是框架frame调用的

Event object will be passed here

Layout Management (布局管理器): LayoutManager (in package java.awt)

FlowLayout (流式布局)

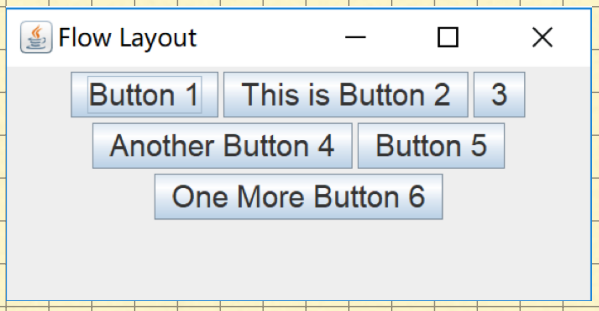
将组件从左向右排行，一行位置不够换下行，默认居中

是 javax.swing.JPanel 中的默认布局管理器

```
public class FlowLayoutDemo extends JFrame {
   private JButton btn1, btn2, btn3, btn4, btn5, btn6;

   public FlowLayoutDemo() {
      super("Flow Layout");
      setLayout(new FlowLayout());
      btn1 = new JButton("Button 1"); add(btn1);
      btn2 = new JButton("This is Button 2"); add(btn2);
      btn3 = new JButton("3"); add(btn3);
      btn4 = new JButton("Another Button 4"); add(btn4);
      btn5 = new JButton("Button 5"); add(btn5);
      btn6 = new JButton("One More Button 6"); add(btn6);
   }

   public static void main(String[] args) { ... }
}
```



GridLayout (网格布局)

将组件按行列排

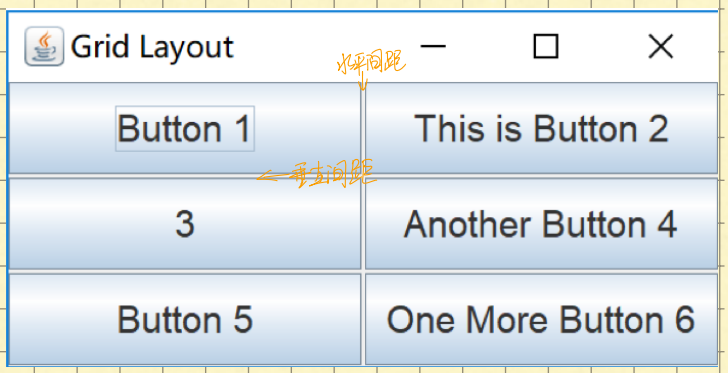
```
public class GridLayoutDemo extends JFrame {
   private JButton btn1, btn2, btn3, btn4, btn5, btn6;

   public GridLayoutDemo() {
      super("Grid Layout");
      setLayout(new GridLayout(3, 2, 3, 3));
      btn1 = new JButton("Button 1"); add(btn1);
      btn2 = new JButton("This is Button 2"); add(btn2);
      btn3 = new JButton("3"); add(btn3);
      btn4 = new JButton("Another Button 4"); add(btn4);
      btn5 = new JButton("Button 5"); add(btn5);
      btn6 = new JButton("One More Button 6"); add(btn6);
   }

   public static void main(String[] args) { ... }
}
```

3 x 2 grid layout (3 rows, 2 columns)
Horizontal and vertical gaps between components: 3 pixels

好 列 水平 垂直间距



BorderLayout (边框布局)

将界面分为东/西/南/北/中五个区域

是 javax.swing.JFrame 中的默认布局管理器

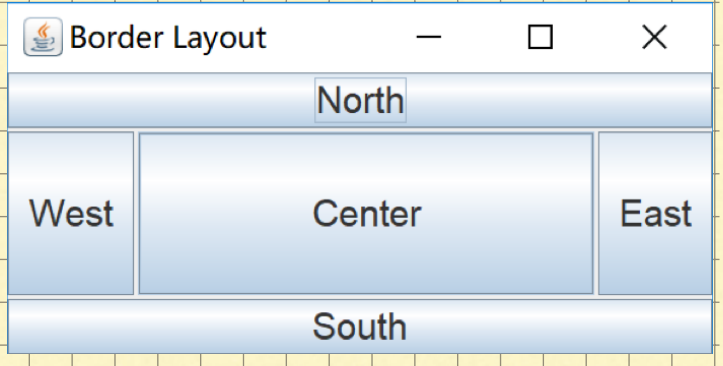
```
public class BorderLayoutDemo extends JFrame {
  private JButton btnNorth, btnSouth, btnCenter, btnEast, btnWest;

  public BorderLayoutDemo() {
    super("Border Layout");
    setLayout(new BorderLayout(3, 3));
    btnNorth = new JButton("North"); add(btnNorth, BorderLayout.NORTH);
    btnSouth = new JButton("South"); add(btnSouth, BorderLayout.SOUTH);
    btnCenter = new JButton("Center"); add(btnCenter, BorderLayout.CENTER);
    btnEast = new JButton("East"); add(btnEast, BorderLayout.EAST);
    btnWest = new JButton("West"); add(btnWest, BorderLayout.WEST);
  }

  public static void main(String[] args) { ... }
}
```

Horizontal and vertical gaps: 3 pixels

水平 垂直间距



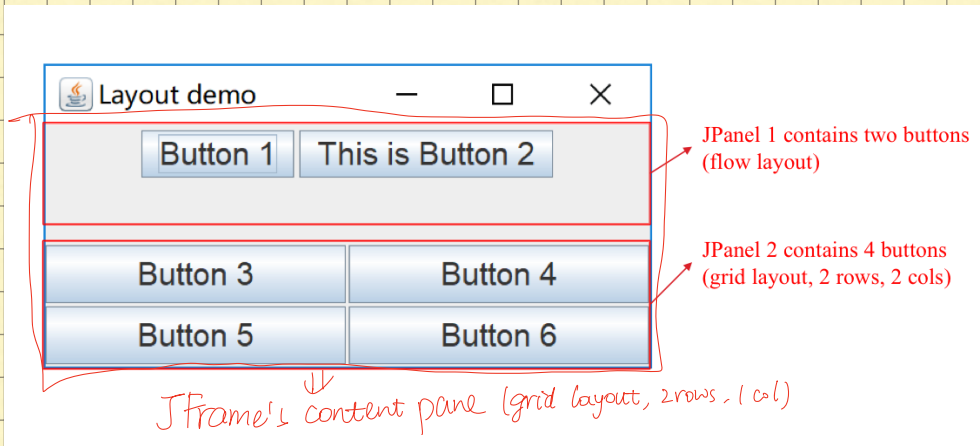
用两个容器布局：

```
public class LayoutDemo extends JFrame {
    private JButton btn1, btn2, btn3, btn4, btn5, btn6;

    public LayoutDemo() {
        super("Layout demo");
        setLayout(new GridLayout(2, 1)); //set the layout of JFrame's content pane
        JPanel panel1 = new JPanel(new FlowLayout());
        JPanel panel2 = new JPanel(new GridLayout(2, 2, 3, 3));
        add(panel1); add(panel2); //add the two JPanels to the JFrame
        btn1 = new JButton("Button 1"); panel1.add(btn1);
        btn2 = new JButton("This is Button 2"); panel1.add(btn2);
        btn3 = new JButton("Button 3"); panel2.add(btn3);
        btn4 = new JButton("Button 4"); panel2.add(btn4);
        btn5 = new JButton("Button 5"); panel2.add(btn5);
        btn6 = new JButton("Button 6"); panel2.add(btn6);
    }
    public static void main(String[] args) {...}
}
```

Create two JPanels and set layout

Group buttons



AWT P6-9.

Frame窗口：

```java
import java.awt.*;

//GUI的第一个界面
public class TestFrame {
    public static void main(String[] args) {

        //Frame，JDK， 看源码!
        Frame frame = new Frame( title: "我的第一个Java图像界面窗口");

        //需要设置可见性  w h
        frame.setVisible(true);

        //设置窗口大小
        frame.setSize( width: 400, height: 400);

        //设置背景颜色  Color
        frame.setBackground(new Color( r: 85,  g: 150,  b: 68));

        //弹出的初始位置
        frame.setLocation( x: 200, y: 200);

        //设置大小固定
        frame.setResizable(false);


    }
}
```

```java

import java.awt.*;

public class TestFrame2 {
    public static void main(String[] args) {
        //展示多个窗口 new
        MyFrame myFrame1 = new MyFrame(100, 100, 200, 200, Color.blue);
        MyFrame myFrame2 = new MyFrame(300, 100, 200, 200, Color.yellow);
        MyFrame myFrame3 = new MyFrame(100, 300, 200, 200, Color.red);
        MyFrame myFrame4 = new MyFrame(300, 300, 200, 200, Color.MAGENTA);
    }
}

class MyFrame extends Frame{
    static int id = 0; //可能存在多个窗口，我们需要一个计数器

    public MyFrame(int x,int y,int w,int h,Color color){
        super("Myframe+"+(++id));
        setBackground(color);
        setBounds(x,y,w,h);
        setVisible(true);
    }

}
```

Panel画板：不能单独存在，需放在frame上.

```java
import java.awt.*;

//Panel 可以看成是一个空间，但是不能单独存在
public class TestPanel {
    public static void main(String[] args) {
        Frame frame = new Frame();
        Panel panel = new Panel();

        //设置布局
        frame.setLayout(null);

        //坐标
        frame.setBounds( x: 300, y: 300, width: 500, height: 500);
        frame.setBackground(new Color( r: 40,  g: 161,  b: 35));

        //panel设置坐标，相对于frame
        panel.setBounds( x: 50, y: 50, width: 400, height: 400);
        panel.setBackground(new Color( r: 193,  g: 15,  b: 60));

        //frame.add(panel)
        frame.add(panel);

        frame.setVisible(true);


    }
}
```

```java
//监听事件，监听窗口关闭事件  System.exit(0)
//适配器模式 :
frame.addWindowListener(new WindowAdapter() {
    //窗口点击关闭的时候需要做的事情
    @Override
    public void windowClosing(WindowEvent e) {
        //结束程序
        System.exit(0);
    }
});
```

布局管理器

```java
public class TestFlowLayout {
    public static void main(String[] args) {
        Frame frame = new Frame();

        //组件-按钮
        Button button1 = new Button("button1");
        Button button2 = new Button("button2");
        Button button3 = new Button("button3");

        //设置为流式布局
        //frame.setLayout(new FlowLayout());
        //frame.setLayout(new FlowLayout(FlowLayout.LEFT));
        frame.setLayout(new FlowLayout(FlowLayout.RIGHT));

        frame.setSize(200,200);

        //把按钮添加上去
        frame.add(button1);
        frame.add(button2);
        frame.add(button3);

        frame.setVisible(true);


    }
}
```

```java
import java.awt.*;

public class TestGridLayout {
    public static void main(String[] args) {
        Frame frame = new Frame("TestGridLayout");

        Button btn1 = new Button("btn1");
        Button btn2 = new Button("btn2");
        Button btn3 = new Button("btn3");
        Button btn4 = new Button("btn4");
        Button btn5 = new Button("btn5");
        Button btn6 = new Button("btn6");

        frame.setLayout(new GridLayout(3,2));

        frame.add(btn1);
        frame.add(btn2);
        frame.add(btn3);
        frame.add(btn4);
        frame.add(btn5);
        frame.add(btn6);

        frame.pack(); //Java函数!
        frame.setVisible(true);

    }
```

自动适配填充位置

```java
public class TestBorderLayout {
    public static void main(String[] args) {
        Frame frame = new Frame("TestBorderLayout");

        Button east = new Button("East");
        Button west = new Button("West");
        Button south = new Button("South");
        Button north = new Button("North");
        Button center = new Button("Center");


        frame.add(east,BorderLayout.EAST);
        frame.add(west,BorderLayout.WEST);
        frame.add(south,BorderLayout.SOUTH);
        frame.add(north,BorderLayout.NORTH);
        frame.add(center,BorderLayout.CENTER);

        frame.setSize(200,200);
        frame.setVisible(true);

    }
}
```

事件监听

```java
public class TestActionEvent {
    public static void main(String[] args) {
        //按下按钮，触发一些事件
        Frame frame = new Frame();
        Button button = new Button();
        //因为，addActionListener()需要一个 ActionListener，所以我们需要构造一个 ActionListener
        MyActionListener myActionListener = new MyActionListener();
        button.addActionListener(myActionListener);
        //文本框

        frame.add(button,BorderLayout.CENTER);
        frame.pack();

        windowClose(frame); //关闭窗口
        frame.setVisible(true);


    }

    //关闭窗体的事件
    private static void windowClose(Frame frame){
        frame.addWindowListener(new WindowAdapter() {
            @Override
            public void windowClosing(WindowEvent e) {
                System.exit( status: 0);
            }
        });
    }
```

```java
//事件监听
class MyActionListener implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        System.out.println("aaa");
    }

}
```

```java
class MyMonitor implements ActionListener{

    @Override
    public void actionPerformed(ActionEvent e) {
        // e.getActionCommand() 获得按钮的信息
        System.out.println("按钮被点击了: msg=> "+e.getActionCommand());
    }

}
```

可多个按钮只写一个监听类，共享一个事件