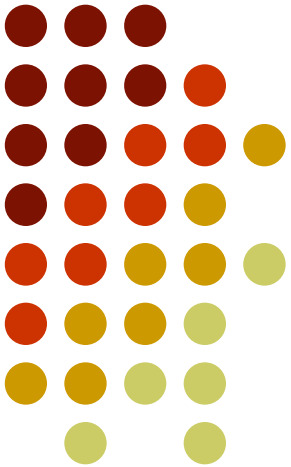
第六章 图形用户界面设计

1. Java的GUI及开发工具
2. 容器和组件
3. GUI的布局
4. 事件处理
5. Java的标准组件



图形用户界面（GUI）

- 字符界面
  - 用字符串、命令行的方式与用户交互
- 图形用户界面(Graphics User Interface)
  - 用直观的图形来表示数据
  - 用直观、方便的GUI标准组件来接收命令
  - GUI组成成分的标准化

使用图形的方式借助菜单、按钮等标准界面元素和鼠标操作，来帮助用户方便地向计算机系统发出命令，启动操作，并将系统运行的结果同样以图形的方式显示给用户。

JAVA 开发工具

- **Eclipse**
- Jdk
- Apache NetBeans11.1 https://netbeans.org/
- Weblogic
- websphere
- jRun
- jboss
- jbuilder
- Ajax

JavaFX

Java GUI工具

- Eclipse插件
  - ✓ **WindowBuilder** https://www.eclipse.org/windowbuilder
  - ✓ GWT developer plugin http://www.gwtproject.org
  - ✓ https://developers.google.com/web-toolkit/download?hl=zh-CN
- http://www.codegear.com/products/jbuilder JBuilder 2007 Editions (http://www.borland.com/)

- file:///D:/JAVA/Sun/docs/index.html

Google收购了Instantiations

- 2010.8 Google收购了Instantiations，Google重新发布了Instantiations的开发工具，并且对所有开发人员免费:

  1.GWT Designer：强大的基于 Eclipse的开发工具，使Java开发者能够使用Google Web Toolkit (GWT)快速创建Ajax用户界面。

  2.CodePro AnalytiX：全面自动化的软件代码质量及安全性分析工具，用于提高软件质量、可靠性和可维护性。

  3.WindowBuilder Pro：Java图形界面设计器，支持Swing, SWT, GWT, RCP, 和 XWT UI框架

  4.WindowTester Pro：使用Java富客户端应用测试GUI交互，支持SWT 和 Swing UI框架

- new release of the Google Plugin for Eclipse:

  http://code.google.com/eclipse/docs/getting_started.html

  To learn more refer to the Google Code Blog:

  http://googlewebtoolkit.blogspot.com/

Tutorials

Create, build and run a GWT application – Create, build, debug and compile a sample application.

Communicating with the server – Add an asynchronous call to a web server using GWT RPC or JSON, serialize Java objects, and handle exceptions

Internationalizing a GWT application – Translate the user interface of a GWT application into another language

Unit testing with JUnit – Add unit tests to a GWT application using JUnit

Deploying to Google App Engine – Deploy a GWT application to App Engine

Developer Guide

What's New in GWT 2.5.1?

Organize Projects – Describes conventions to identify which code is intended to run on the client browser, the server, or both

Compile & Debug – Describes development and production modes

Coding Basics – Describes GWT programming fundamentals

Build User Interfaces – How to work with widgets, panels, the DOM, events, CSS, declarative UI and images. Cell widgets / Editors - 2.1, Cell tables - 2.2

HTML5 Feature Support 2.3 – Describes GWT support for HTML5 features, such as Storage, Canvas, Audio, Video, drag and drop, and so forth.

Security for GWT Applications - How to secure your GWT applications against JavaScript attacks

Security: Safe HTML 2.1 – Describes coding guidelines that prevent a large class of Cross-Site-Scripting (XSS) vulnerabilities

Security: GWT RPC XSRF protection 2.3 – Describes how to prevent Cross-Site Request Forgery (XSRF or CSRF) vulnerabilities in GWT RPCs

MVP Framework 2.1 – Sample app and documentation showing how to use Activities, Places, and the EventBus.

RequestFactory 2.1 – Guide to creating data-oriented services using RequestFactory and EntityProxy classes.

Logging 2.1 – Describes how to log events in client-side code in GWT applications.

Accessibility 2.5 – Describes features that enable screen readers to interpret what is displayed on the screen for a visually impaired user

Internationalization – Describes a flexible set of tools to help you internationalize your applications and libraries

Communicate with a Server – Describes a couple of different ways to communicate with a server via HTTP

Test with JUnit – Describes how to use the JUnit unit testing framework and Emma code coverage tool

Deploy – Describes how to deploy both client- and server-side JavaScript

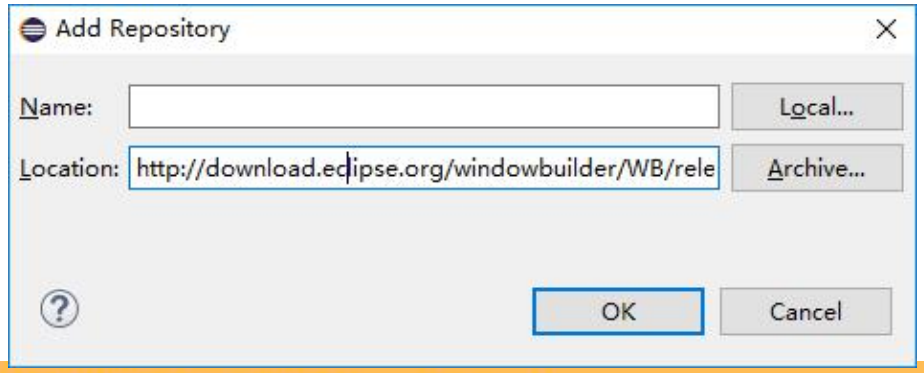
Optimize – Describes how to improve the performance of your application

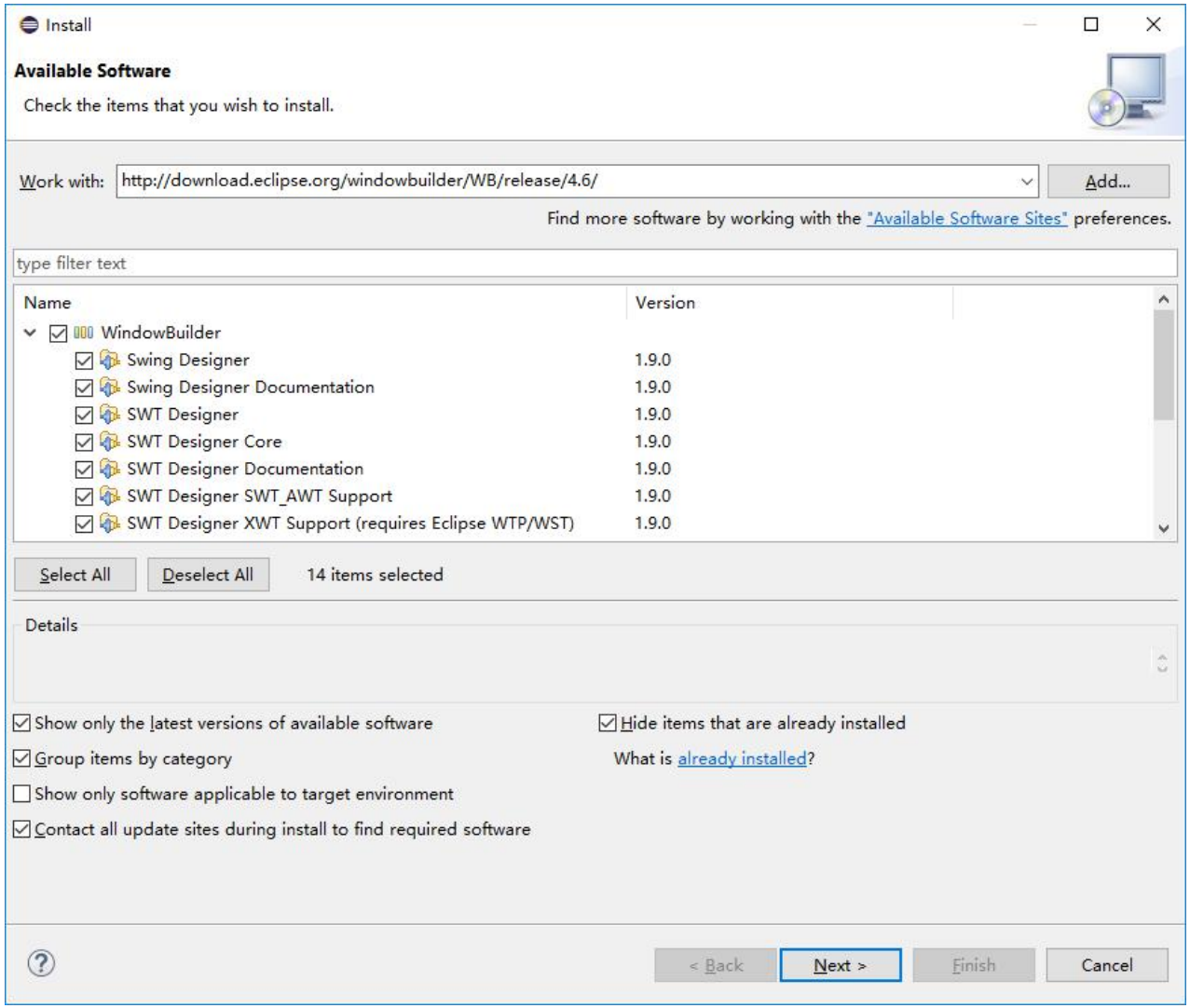
IE9 Support - Tips and Tricks 2.3 - Support for Internet Explorer 9

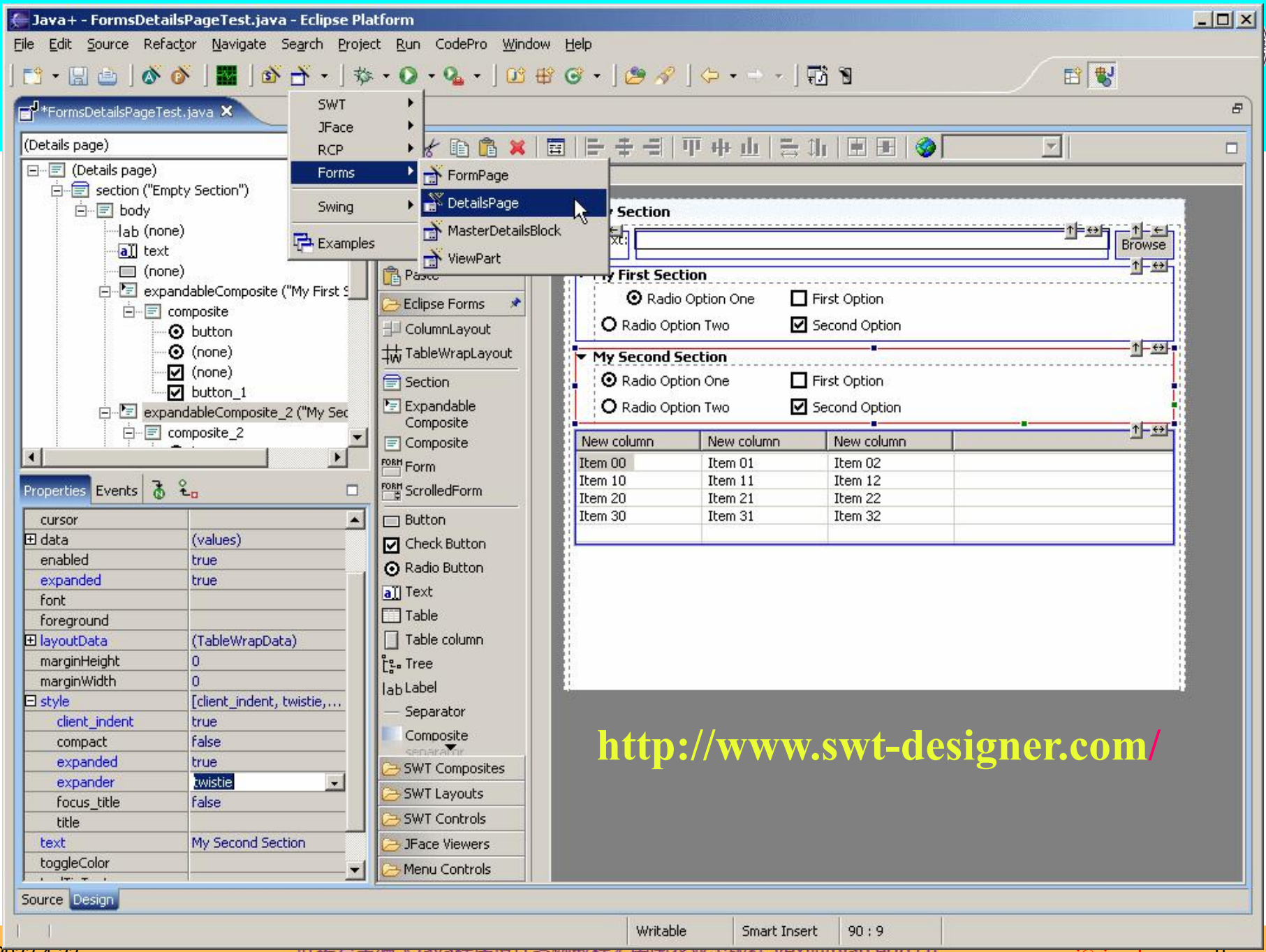
Reference – Provides javadoc for GWT and related libraries and technical details for GWT widgets

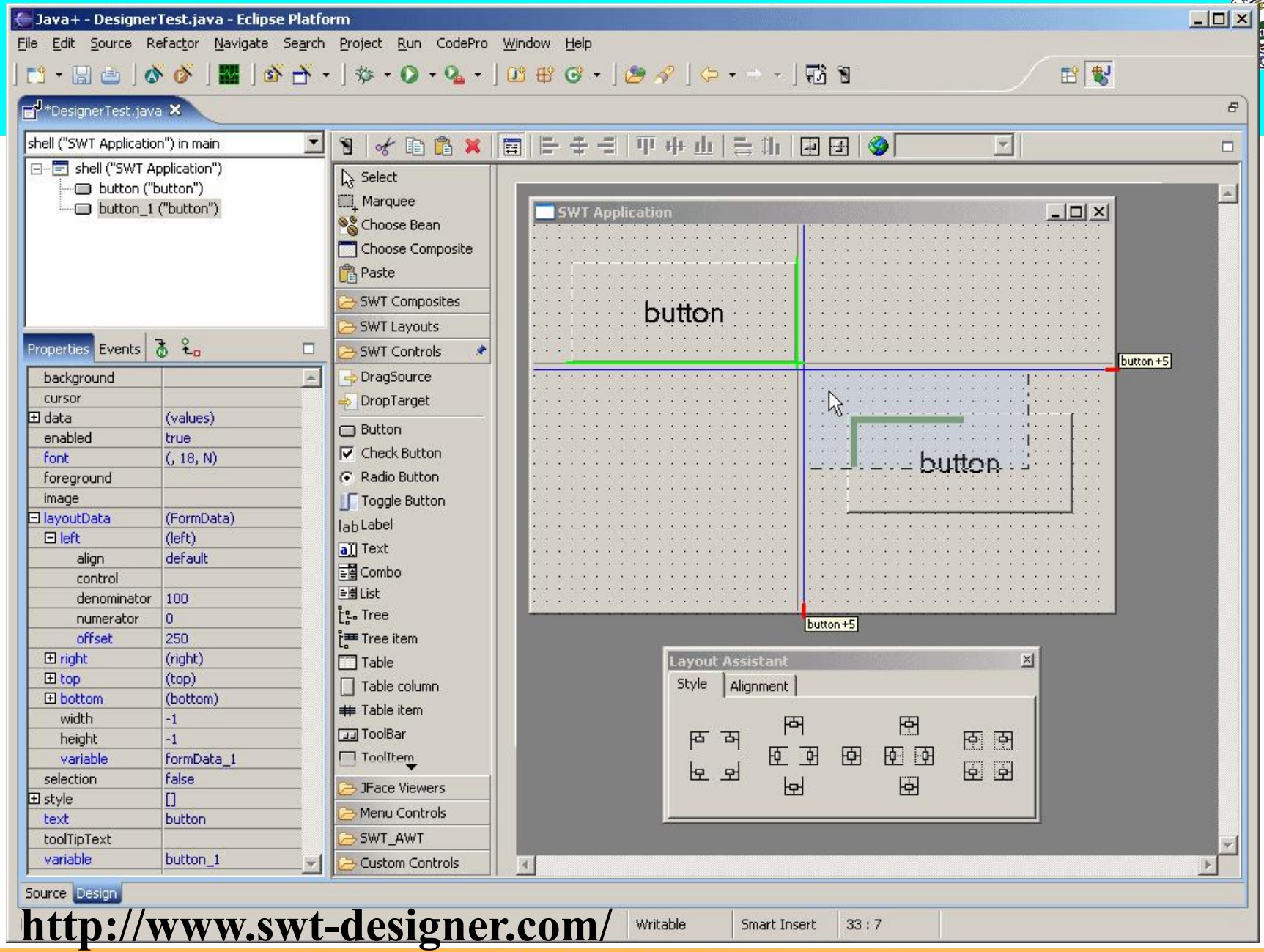
Installing WindowBuilder Pro

- 针对Eclipse版本(neon4.6.3）
  (https://www.eclipse.org/windowbuilder/download.php)
- **To install WindowBuilder Pro Eclipse from this update site**
  1. Select "Help > Install New Software..." in the main menu to open the "Install" dialog
  2. click the "Add..." button in the "Install" dialog,then copy and paste the URL into the "Add Site" dialog and click "OK".
  3. If you add the URL but WindowBuilder Pro Eclipse does not appear in the list of sites, then pull down the "Work with" field and select "All Available Sites".
  4. Check all features to be installed in the "Install" dialog
  5. Click the "Next" button
- **在Location中输入：**
  http://download.eclipse.org/windowbuilder/WB/release/4.6/
  - ✓ 按OK进入下一步。

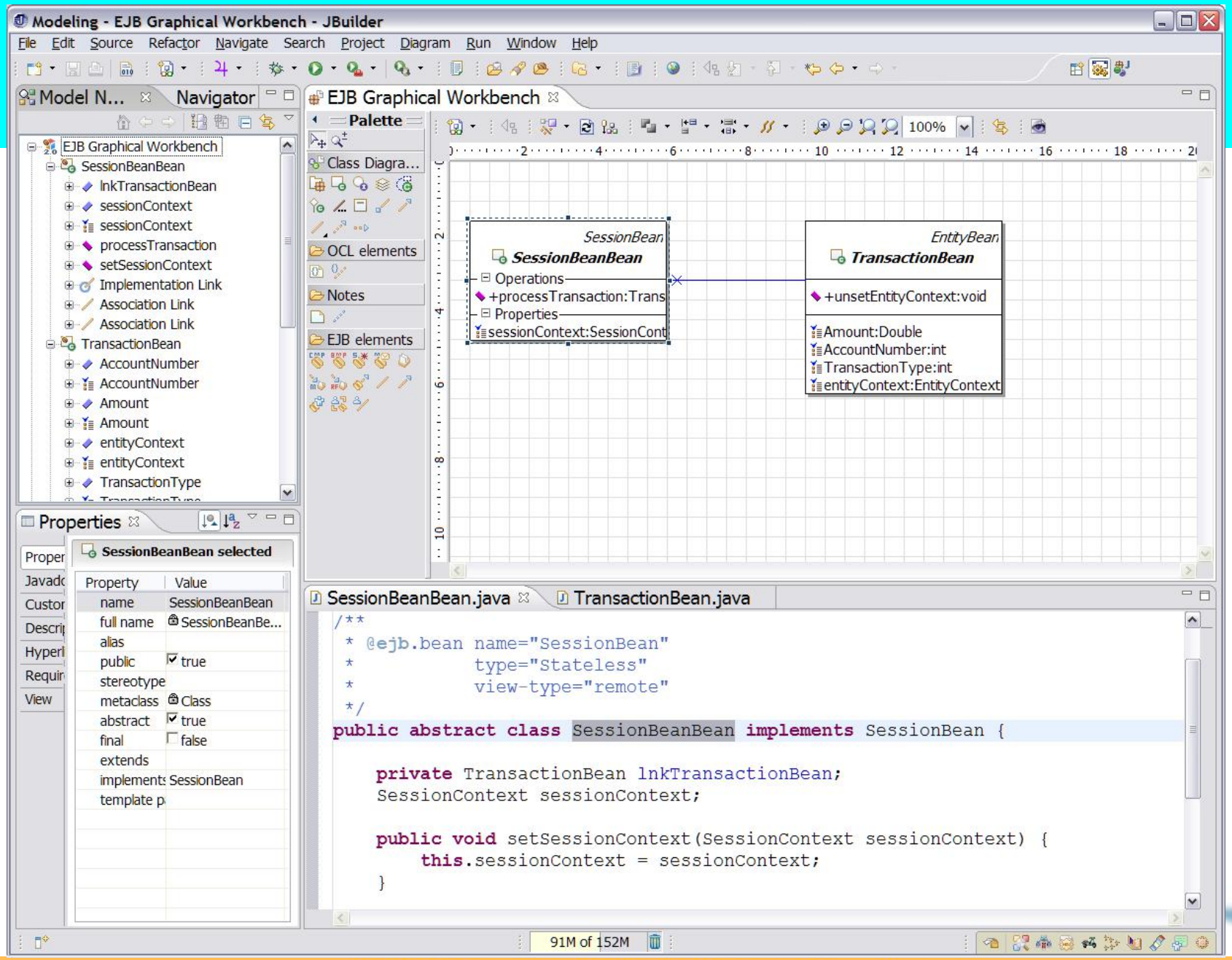



Install
Available Software
Check the items that you wish to install.
Work with:
http://download.eclipse.org/windowbuilder/WB/release/4.6/
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Version
WindowBuilder
Swing Designer
1.9.0
Swing Designer Documentation
1.9.0
SWT Designer
1.9.0
SWT Designer Core
1.9.0
SWT Designer Documentation
1.9.0
SWT Designer SWT_AWT Support
1.9.0
SWT Designer XWT Support (requires Eclipse WTP/WST)
1.9.0
Select All
Deselect All
14 items selected
Details
Show only the latest versions of available software
Hide items that are already installed
Group items by category
What is already installed?
Show only software applicable to target environment
Contact all update sites during install to find required software
< Back
Next >
Finish
Cancel


Java+ - FormsDetailsPageTest.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run CodePro Window Help
*FormsDetailsPageTest.java
(Details page)
(Details page)
section ("Empty Section")
body
lab (none)
text
(none)
expandableComposite ("My First S
composite
button
(none)
(none)
button_1
expandableComposite_2 ("My Sec
composite_2
SWT
JFace
RCP
Forms
Swing
Examples
FormPage
DetailsPage
MasterDetailsBlock
ViewPart
Eclipse Forms
ColumnLayout
TableWrapLayout
Section
Expandable Composite
Composite
Form
ScrolledForm
Button
Check Button
Radio Button
Text
Table
Table column
Tree
Label
Separator
Composite
SWT Composites
SWT Layouts
SWT Controls
JFace Viewers
Menu Controls
Properties
Events
cursor
data (values)
enabled true
expanded true
font
foreground
layoutData (TableWrapData)
marginHeight 0
marginWidth 0
style [client_indent, twistie,...
client_indent true
compact false
expanded true
expander twistie
focus_title false
title
text My Second Section
toggleColor
Browse
My First Section
Radio Option One
First Option
Radio Option Two
Second Option
My Second Section
Radio Option One
First Option
Radio Option Two
Second Option
New column
New column
New column
Item 00
Item 01
Item 02
Item 10
Item 11
Item 12
Item 20
Item 21
Item 22
Item 30
Item 31
Item 32
http://www.swt-designer.com/
Source
Design
Writable
Smart Insert
90 : 9



http://www.swt-designer.com/


Modeling - EJB Graphical Workbench - JBuilder
File Edit Source Refactor Navigate Search Project Diagram Run Window Help
Model N... Navigator
EJB Graphical Workbench
SessionBeanBean
lnkTransactionBean
sessionContext
sessionContext
processTransaction
setSessionContext
Implementation Link
Association Link
Association Link
TransactionBean
AccountNumber
AccountNumber
Amount
Amount
entityContext
entityContext
TransactionType
Palette
Class Diagra...
OCL elements
Notes
EJB elements
100%
SessionBean
SessionBeanBean
Operations
+processTransaction:Trans
Properties
sessionContext:SessionCont
EntityBean
TransactionBean
+unsetEntityContext:void
Amount:Double
AccountNumber:int
TransactionType:int
entityContext:EntityContext
Properties
SessionBeanBean selected
Property Value
name SessionBeanBean
full name SessionBeanBe...
alias
public true
stereotype
metaclass Class
abstract true
final false
extends
implements SessionBean
template p
SessionBeanBean.java
TransactionBean.java
/**
* @ejb.bean name="SessionBean"
* type="Stateless"
* view-type="remote"
*/
public abstract class SessionBeanBean implements SessionBean {
private TransactionBean lnkTransactionBean;
SessionContext sessionContext;
public void setSessionContext(SessionContext sessionContext) {
this.sessionContext = sessionContext;
}
91M of 152M

常用的Java项目开发环境

- JBuilder、VisualAge for Java、Forte for Java, Visual Cafe、Eclipse、NetBeans IDE、JCreator +J2SDK、jdk+记事本、EditPlus+ J2SDK等等。
- 一般开发J2EE项目时都需要安装各公司的应用服务器（中间件）和相应的开发工具

用户界面设计原则

- 控制权在用户
- 界面与操作的风格一致性
- 宽容（容错）性
- 简洁与美观并重

6.1 图形用户界面简介

- 图形用户界面的构成
  - 容器：布局、安排
  - 标准组件
  - 用户自定义成分

Java的图形用户界面编程：

- 为方便图形用户界面的开发，设计了专门的类库 来生成各种标准图形界面元素和处理图形界面的 各种事件，这个用来生成图形界面的类库就是java.awt包。AWT是**abstract window toolkit**抽象窗口工具集的缩写。
- AWT类库中的各种操作被定义在一个窗口中进行，开发人员用AWT开发出的图形用户界面可以适用于所有的平台系统。
- **Java1.2为Java 1.0 AWT添加一个被称为“Swing”的GUI部分。**丰富的、易于使用和理解的Java Beans能经过拖放操作，创建出能使程序员满意的GUI。

AWT包

- Abstract Window Toolkit(抽象窗口工具集)
  - 提供各种构成GUI的标准构件。
  - AWT类库中的各种操作被定义在一个窗口中进行的。
  - 抽取不同软硬件平台中所实现的窗口的公共特性。
  - 依赖于具体平台系统实现：显示效果可能不同。
- 提供与机器无关的基本GUI标准组件
  - 选择类组件：单选按钮、复选框、下拉选单、列表框
  - 文字处理类组件：标签、文本框、编辑框。
  - 命令类组件：按钮、工具栏、菜单等。

从一个简单窗口开始

- 创建GUI应用程序
  - ✓ Frame必不可少

    它是带标题的顶层窗口，是构建应用程序图形界面的基础，它为应用程序实现人机交互提供了对话窗口
- 先来建一个空的窗口应用: Empty.java
  - ✓ 注意：一定要处理关闭窗口的事件

例6.1 Empty.java 产生一个空窗口

```
import java.awt.*;  //must import to use GUI
                    //must extends Frame in an application
public class Empty extends Frame{
                        // 暂时采用Java的事件处理方法，关闭窗口
   public boolean handleEvent(Event evt){
                //根据Event类中参数evt的id值来判断发生事件的种类
          if(evt.id == Event.WINDOW_DESTROY)
              System.exit(0);
   // return的返回值表示将余下的事件处理交给handleEvent( )的父类
   // 处理，从而保证对事件的正常处理
              return super.handleEvent(evt);
   }

   public static void main(String[] args){
          Frame f=new Empty();
          f.setSize(300,200);
          f.setVisible(true);
   }
}
```

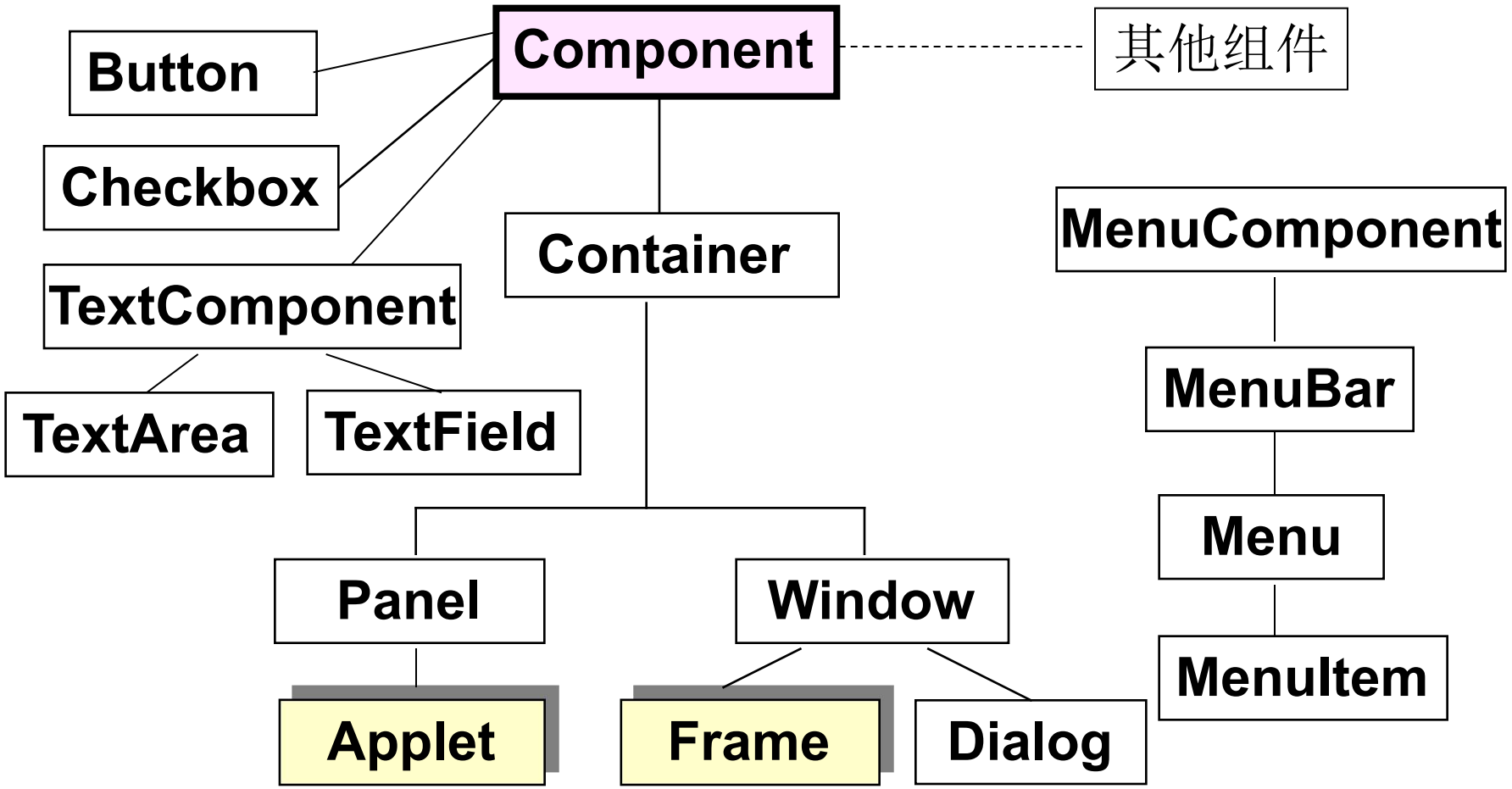
另一例EX1: GridArr.java

GUI标准组件概述

- 构建程序的图形用户界面的主要任务有两个
  - 创建各界面组件并排列成图形用户界面的外观(设计)
  - 定义这些组件对不同事件的影响从而完成图形用户界面功能(编程)
- 组成图形用户界面的成分有三类
  - 容器: 是能够容纳并排列其他组件的对象
  - 组件: 放置在容器中的对象
  - 用户自定义类

在Java中，容器和组件是由AWT包中的对象来代表，这些对象间的层次关系如下图：

AWT类层次关系图


Component
其他组件
Button
Checkbox
TextComponent
TextArea
TextField
Container
Panel
Applet
Window
Frame
Dialog
MenuComponent
MenuBar
Menu
MenuItem

GUI标准组件

- Component:所有组件和容器的抽象父类
  - 显示功能：paint(), update(), repaint()
  - 显示效果控制：字体、颜色、位置、尺寸
  - 图象处理：一般利用Canvas和Container来显示图像
  - 事件处理机制（java 1.1）：
    - addXXXListener( )
    - removeXXXListener( )

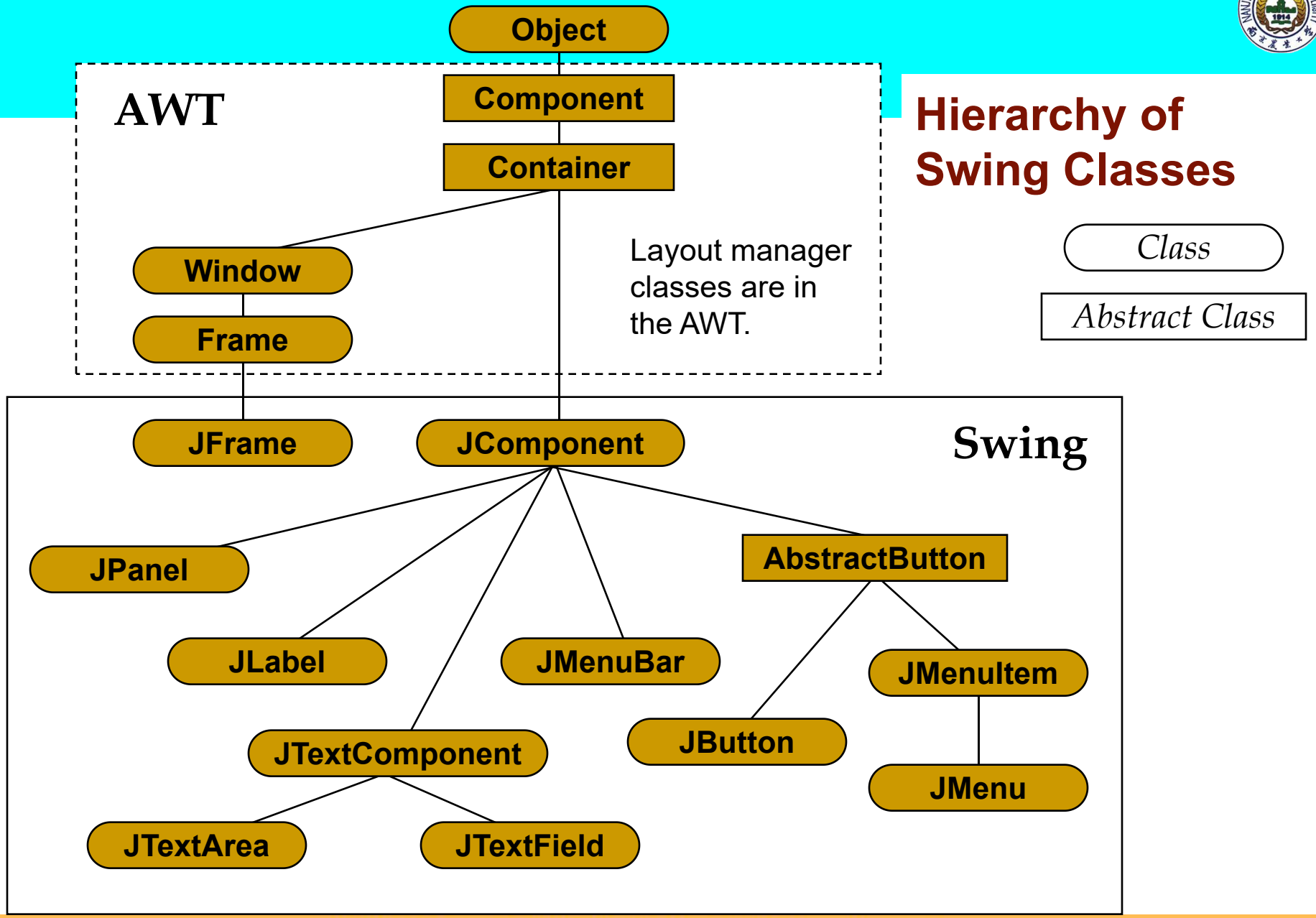
Component类

- 可显示在屏幕上的图形对象，可与用户交互。
- 是所有组件和容器的抽象父类，其中定义了一些每个容器和组件都可能用到的方法。
- add(PopupMenu popup) 在组件上加入一弹出菜单
- addFocusListener(FocusListener l)将发生在本组件上的事件注册给监听者，以进行事件处理。
- setSize(int width, int height) 设置组件尺寸
- repaint(int x,int y,int width, int height) 重画组件
- setFont(Font f) 设置组件字体
- setBackground(Color c) 设置组件背景色
- setVisible(boolean b) 设置组件是否可见

Swing

- Java Swing包中包含250多个类，是组件和支持类的集合。Swing提供了40多个组件，是AWT组件的四倍，除提供替代AWT的组件外，还提供了大量有助于开发图形用户界面的附加组件。对于初学者来说，Swing的类显得过于庞大和复杂。

Hierarchy of Swing Classes



- file:///I:\JavaEx\tutorial\uiswing\index.html
- https://docs.oracle.com/javase/tutorial/uiswing/components/tabbedpane.html
- http://java.sun.com/docs/books/tutorial/
- file:///D:/JavaEx/tutorial/ui/features/components.html

A simple demonstration of a window constructed with Swing


```
import javax.swing.*;
public class FirstSwingDemo  {
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    public static void main(String[] args)    {
        JFrame myWindow = new JFrame(“First Window”);
        myWindow.setSize(WIDTH, HEIGHT);
        JLabel myLabel = new JLabel("Please don't click that button!");
        myWindow.getContentPane().add(myLabel);

        WindowDestroyer myListener = new WindowDestroyer();
        myWindow.addWindowListener(myListener);

        myWindow.setVisible(true);
    } }
```

6.2 容器和组件

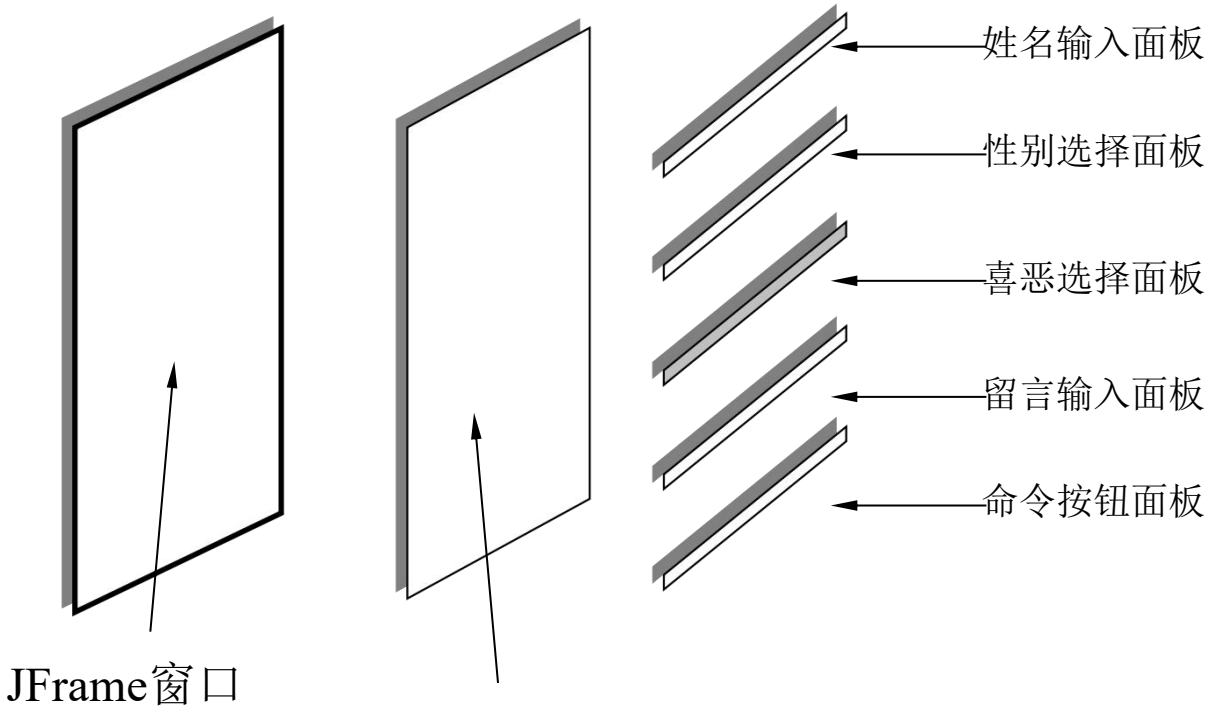
- GUI标准组件
- ：
  - ✓

GUI标准组件

- 简单构件：人机交互的基本工具(控制)
  - Button,Checkbox, Label等
  - TextComponent(TextArea,TextField)
  - Canvas
- 复杂构件:
  - Container(安放排列其他构件的容器)
  - Panel
  - Window

容器组件

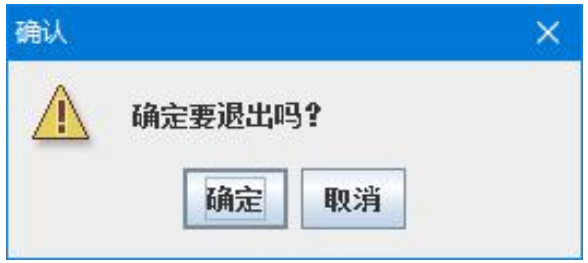
常用的GUI容器组件有面板(Jpanel)、JOptionPane、带滚动条(JscroolPanel)的面板等



How to Make Dialogs：JOptionPane

java.awt.Component
-java.awt.Container
-javax.swing.JComponent
-javax.swing.**JOptionPane**

question
information
warning
error




```
JOptionPane.showConfirmDialog(this,"确定要退出吗？ ","确认",
JOptionPane.OK_CANCEL_OPTION,JOptionPane.WARNING_MESSAGE);
```

Method Name	Description
showConfirmDialogs	Asks a confirming question, like yes/no/cancel.
showInputDialog	Prompt for some input.
showMessageDialog	Tell the user about something that has happened.
showOptionDialog	The Grand Unification of the above three.

https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html

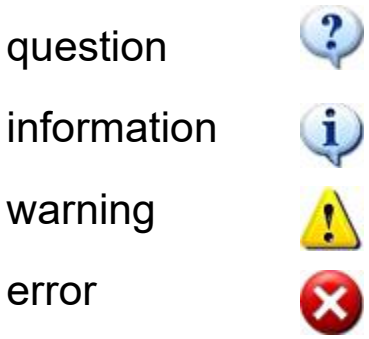
https://docs.oracle.com/javase/8/docs/api/javax/swing/JOptionPane.html

A Sample GUI Program

```
import javax.swing.*;
public class ASampleGUIProgram  {
   public static void main(String[] args)   {
      String appleString,orangeString;
      int appleCount,orangeCount,totalFruitCount;

      appleString = JOptionPane.showInputDialog("Enter number of apples:");
      appleCount = Integer.parseInt(appleString);
      orangeString = JOptionPane.showInputDialog("Enter number of oranges:");
      orangeCount = Integer.parseInt(orangeString);
      totalFruitCount = appleCount + orangeCount;

      JOptionPane.showMessageDialog(null, "The total number of fruits = " +
     totalFruitCount);

      System.exit(0);
} }
```

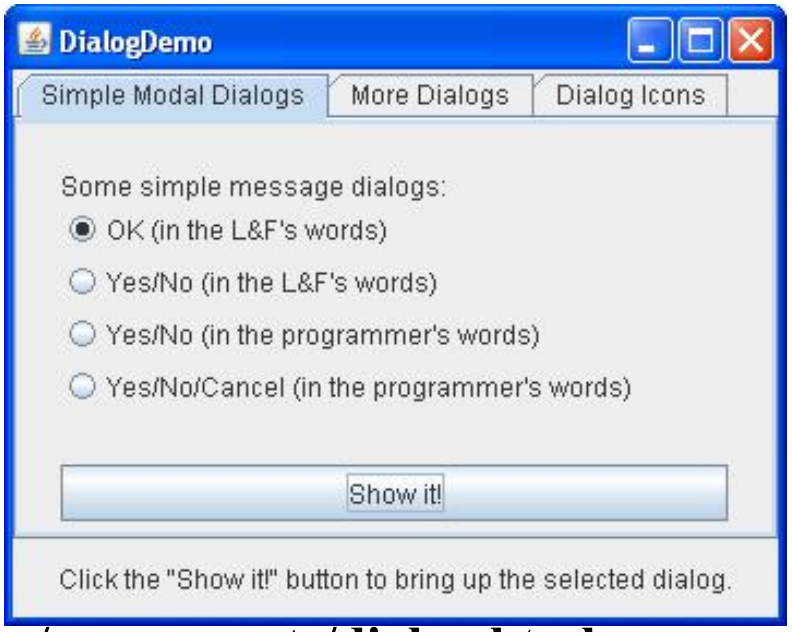
Using Swing Components: Examples

https://docs.oracle.com/javase/tutorial/uiswing/examples/components/index.html#DialogDemo



JOptionPane.showMessageDialog(frame, "Eggs are not supposed to be green.");

The DialogDemo Example



https://docs.oracle.com/javase/tutorial/uiswing/components/dialog.html

Container（容器）

- 排列其中包容的构件
  - 定位布局策略 set/getLayout()
- 包容其他基本构件
  - 增加构件: add( )
  - 删除构件: remove( )
- 控制是否显示容器及其中组件
  - setVisible(true/false)

Container容器组件

- Window —无边框、菜单的空白窗口。不需要其他组件支撑，独立显示。
  - Frame: 用于Application，含边框、标题和菜单的独立窗口
  - Dialog: 依赖于 Frame的非独立窗口。没有菜单条，不能改变大小。
- Panel (面板)—属于无边框容器。必须放在Window组件中(或Web浏览器窗口)才能显示。它为一矩形区域，在其中可摆放其他组件，可以有自己的布局管理器。
  - 无边框容器包括Panel和Applet。其中Applet为Panel的子类
- 基本方法
  - add(Component comp) 将指定组件放到容器中
  - getComponent(int index) 获取指定序号的组件
  - remove(Component comp) 删除指定组件
  - setLayout(LayoutManager mgr) 设置容器布局

6.3 JAVA的布局

- 布局管理器（layout manager)
  - 用于控制组件在容器中的布局
  - 负责确定组件在容器中的位置和大小。
  - 调用容器的setLayout(布局管理器对象) 方法，为容器指定某种布局。
  - 当容器需要定位组件和确定组件大小时，就会给布局管理器对象发消息，让它完成该项工作。
- 直接管理组件
  - 调用容器的 setLayout(null) 方法，关闭布局管理器。
  - 调用每一个组件的setLocation()方法决定组件位置。
  - 调用每一个组件的setSize()方法决定其大小。

布局管理器种类

在Java的GUI界面设计中，布局控制是通过为容器设置布局编辑器来实现的。Java.awt包中共定义了五种布局编辑类，每一个布局编辑类对应一种布局策略：

- FlowLayout：组件在一行中按加入的先后顺序从左至右水平排列，排满后折行，每行中的组件都居中排列。
- BorderLayout：把容器空间划分为北、南、西、东、中五个区，每加入一个组件都应说明把这个组件加在那个区域中。
- CardLayout：每一个组件作为一个卡片，容器仅显示多个卡片中的某一个。
- GridLayout：以行和列的网格形式安排组件。
- GridBagLayout：更复杂、功能更强的网格布局。

JDK8.0 布局管理器种类

Several AWT and Swing classes provide layout managers for general use:

- BorderLayout
- BoxLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout
- GroupLayout
- SpringLayout

https://docs.oracle.com/javase/tutorial/uiswing/layout/index.html

file:///D:/JavaEx/tutorial/uiswing/layout/visual.html

FlowLayout布局管理器

- 构造方法：
  - ✓ FlowLayout() ;
  - ✓ FlowLayout(int align, int hgap, int vgap)；

  align ： 对齐方式：LEFT CENTER RIGHT

  hgap ： 组件水平间距 （象素）

  vgap ： 组件垂直间距
- 无参数的构造方法创建的FlowLayout对象，其对齐方式为CENTER居中方式，组件间的横纵间距都为5个像素。

FlowLayout布局管理器

```
public class FlowLayoutButton extends Frame{
   public static void main(String[] args)   {
      Frame f = new Frame("FlowLayout");
      Button b1,b2,b3;
    f.setSize(400,300);                  //不起作用
    f.setBackground(Color.gray);
    f.setLayout(new FlowLayout(FlowLayout.LEFT,10,20));
    b1 = new Button("button1");
    b2 = new Button("button2");
    b3 = new Button("button3");
    f.add(b1);
    f.add(b2);
    f.add(b3);
    f.addWindowListener(new WindowDestroyer()); //关闭窗口
    f.setVisible(true);
    f.pack();     //询问布局管理器窗体大小
    }  }
```

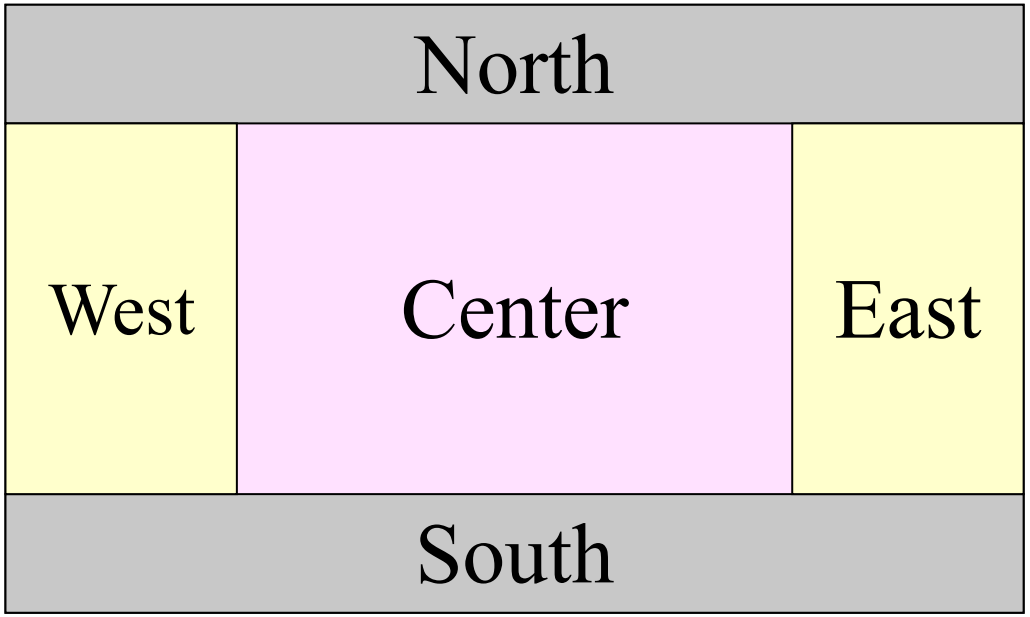
边界布局（BorderLayout）

BorderLayout是较常用的布局，也是Frame 默认的。

setLayout(new BorderLayout())

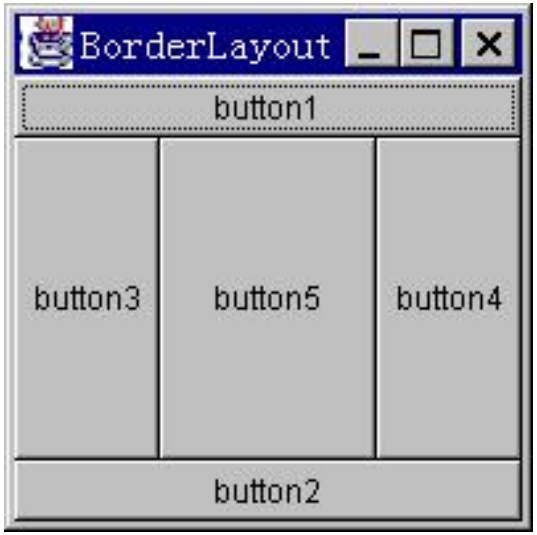
add(b1,BorderLayout.NORTH);

当容器大小改变，组件相对位置不会改变。

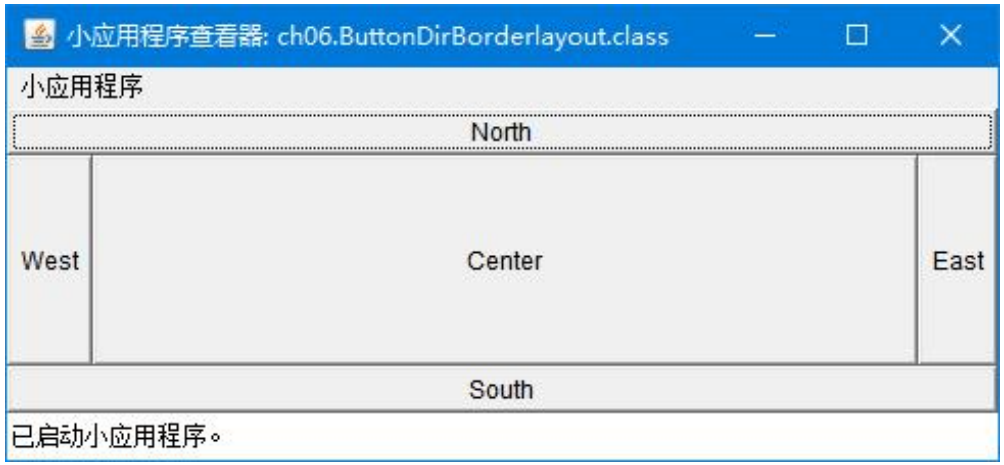


BorderLayout布局管理器

```
f = new Frame("BorderLayout");
f.setSize(200,200);
b1 = new Button("button1");
        …
b5 = new Button("button5");
f.add(b1,BorderLayout.NORTH);
f.add(b2,BorderLayout.SOUTH);
f.add(b3,BorderLayout.WEST);
f.add(b4,BorderLayout.EAST);
f.add(b5,BorderLayout.CENTER);
f.setVisible(true);
```

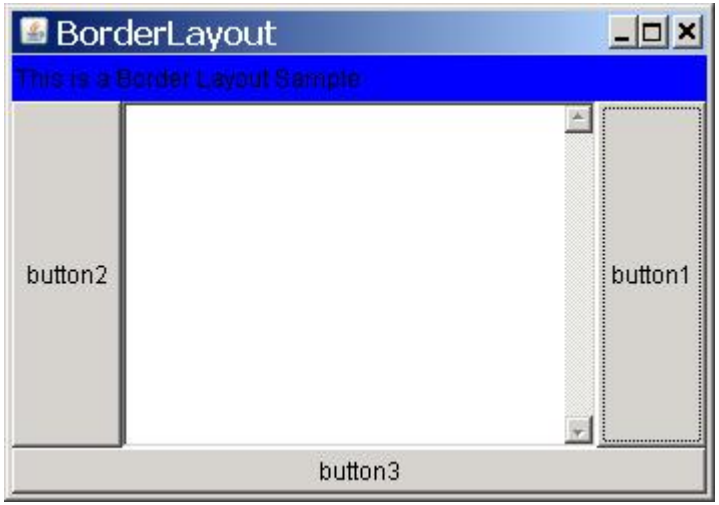


```
package ch06;
import java.awt.*;
import java.applet.Applet;
public class ButtonDirBorderlayout extends Applet {
        public void init() {
                setLayout(new BorderLayout());
                add(new Button("North"), BorderLayout.NORTH);
                add(new Button("South"), BorderLayout.SOUTH);
                add(new Button("East"), BorderLayout.EAST);
                add(new Button("West"), BorderLayout.WEST);
                add(new Button("Center"), BorderLayout.CENTER);
        }
}
```



BorderLayout2.java

```
public class BorderLayout2 extends Frame {
    public static void main(String[] args)    {
     Frame f = new Frame("BorderLayout");
     Label l1,l2;              Button b1,b2,b3;

   f.setBackground(Color.gray);
   // f.setLayout(new BorderLayout()); //默认，可省略
   l1 = new Label("This is a Border Layout Sample");
   l1.setBackground(Color.blue);
   TextArea t1 = new TextArea();
    b1 = new Button("button1");
   b2 = new Button("button2");
   b3 = new Button("button3");
   f.add(l1,BorderLayout.NORTH);
   f.add(b1,BorderLayout.EAST);
   f.add(b2,BorderLayout.WEST);
   f.add(b3,BorderLayout.SOUTH);

   f.add(t1,BorderLayout.CENTER);

   f.addWindowListener(new WindowDestroyer()); //关闭窗口
   f.setVisible(true);
   f.pack();
   } }
```



网格布局（GridLayout）

- 创建GridLayout对象作为布局编辑器，指定划分网格的行数和列数(网格大小一样）。

  setLayout(new GridLayout(行数, 列数));

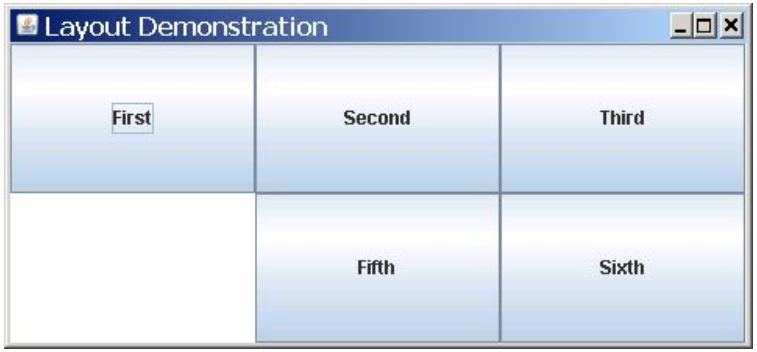
  setLayout(new GridLayout(行数, 列数,行间隔,列间隔));

- 调用容器的方法add()将组件加入容器，组件填入容器的顺序将按照第一行第一个、第一行第二个、……
- 每个网格中都必须填入组件，如果希望某个网格为空白，可以为它加一个空的标签：

  add (new Label());

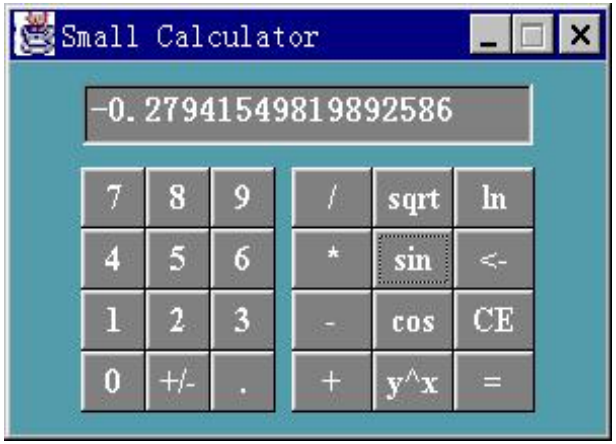
GridLayout 网格布局

```
public void go()   {
   Frame f = new Frame("GridLayout");
        f.setTitle("Layout Demonstration");
        f.setLayout(new GridLayout(2, 3));
       JButton bt1 = new JButton("First");
       f.add(bt1);
       JButton bt2 = new JButton("Second");
       f.add(bt2);
       JButton bt3 = new JButton("Third");
       f.add(bt3);
       JLabel label4 = new JLabel("");//Empty string label
       f.add(label4);
       JButton bt5 = new JButton("Fifth");
       f.add(bt5);
        JButton bt6 = new JButton("Sixth");
       f.add(bt6);
```



布局例

```
Frame fm = new Frame()
fm.setLayout(new FlowLayout());
  tf = new TextField(22);
fm.add(tf);
  Panel p1 = new Panel();
    p1.setLayout(new GridLayout(4,3));
    p1.add(…);
  Panel p2 = new Panel();
    p2.setLayout(new GridLayout(4,3));
    p2.add(…);
fm. add(p1);  fm. add(p2);
```

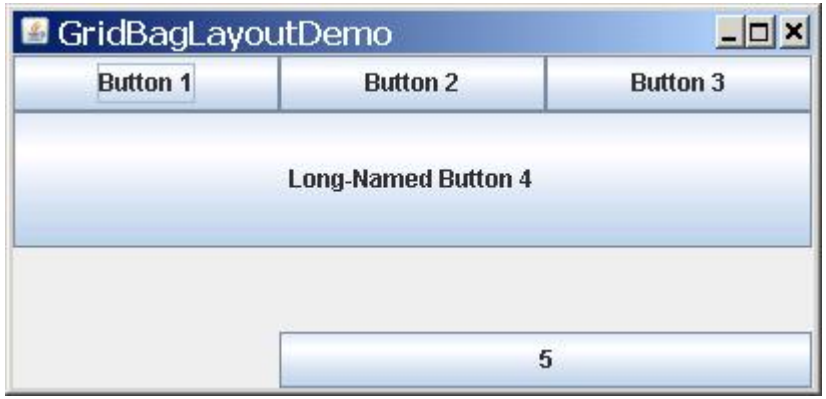


复习EX1:
GridArr.java

网格包布局（GridBagLayout）

- 每个GridBagLayout对象维护一个动态矩形网格，
- 每个组件可占据一个或多个单元作为它的显示区域。
- 每一个组件都与一个GridBagConstraints类的实例相连，以指定在显示区域中如何摆放,并可确定大小。
- 步骤：
  - setLayout(new GridBagLayout());
  - GridBagConstraints gbc = new GridBagConstraints();
  - gbc.gridx = 0; …(属性赋值)
  - add(button1 , gbc)

```
pane.setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();
```



D:\JavaEx\Swing\GridBagLayoutDemo.java

GridBagLayout布局管理器

设置GridBagConstraints类实例属性值：

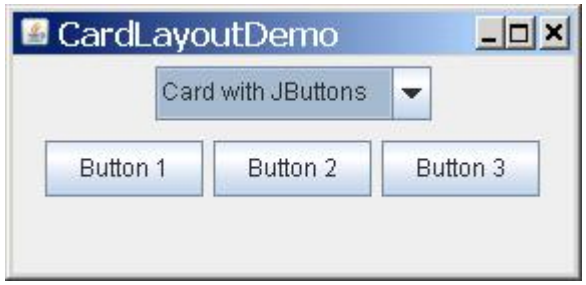
- gridx　gridy　组件显示区域的左上角单元格坐标
- gridwidth gridheight　指定显示区域行、列单元数
- RELATIVE　紧挨着前一个组件摆放
- fill　当显示区域大于组件时如何扩充组件：
  HORIZONTAL 水平充满　VERTICAL　垂直充满
  BOTH 全部填充　NONE　不调整（隐含）
- ipadx ipady　指定组件间最小间隔
- anchor 当显示区域大于组件时如何摆放组件(对齐方式)
  CENTER（隐含）, NORTH, EAST, SOUTH, WEST,
  SOUTHEAST, NORTHEAST,
  SOUTHWEST, NORTHWEST.

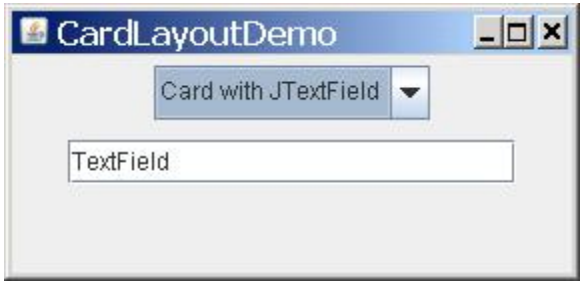
卡片布局(CardLayout)

- 使用CardLayout的容器表面可以容纳多个组件，将每一个组件视为一张卡片，同一时刻只能显示一个组件。
- setLayout(new CardLayout())
- CardLayout方法（按序指定组件）
  - first(Container parent) 显示第一个版面的内容
  - last(Container parent) 显示最后一个版面的内容
  - previous(Container parent) 显示前一个版面的内容
  - next(Container parent) 显示下一个版面的内容
- CardLayout方法(按名显示组件）
  - addLayoutComponent(String name, Component c)
  - show(Container parent, String name)

D:\JavaEx\Swing\CardLayoutDemo.java

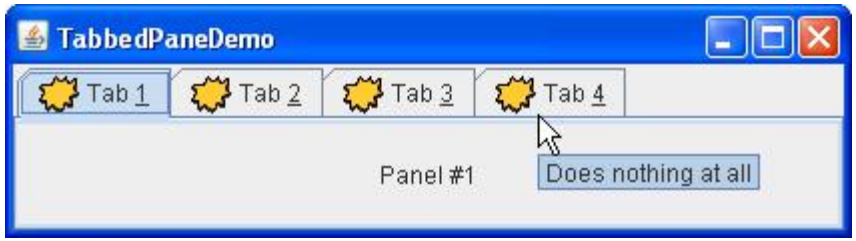
```
JPanel cards; //a panel that uses CardLayout
    public void addComponentToPane(Container pane) {
    JPanel comboBoxPane = new JPanel(); //use FlowLayout
    String comboBoxItems[] = { BUTTONPANEL, TEXTPANEL };
    JComboBox cb = new JComboBox(comboBoxItems);
    cb.setEditable(false);
    cb.addItemListener(this);
    comboBoxPane.add(cb);

    //Create the "cards".
    JPanel card1 = new JPanel();
    card1.add(new JButton("Button 1"));
    card1.add(new JButton("Button 2"));
    card1.add(new JButton("Button 3"));

    JPanel card2 = new JPanel();
    card2.add(new JTextField("TextField", 20));

    //Create the panel that contains the "cards".
    cards = new JPanel(new CardLayout());
    cards.add(card1, BUTTONPANEL);
    cards.add(card2, TEXTPANEL);

    pane.add(comboBoxPane, BorderLayout.PAGE_START);
    pane.add(cards, BorderLayout.CENTER);
}
```





JTabbedPane类

- With the JTabbedPane class, you can have several components, such as panels, share the same space. The user chooses which component to view by selecting the tab corresponding to the desired component. If you want similar functionality without the tab interface, you can use a card layout instead of a tabbed pane.



https://docs.oracle.com/javase/tutorial/uiswing/components/tabbedpane.html

确定容器布局

- 缺省的布局管理器
  - Window、Frame、Dialog — **BorderLayout**: 缺省的窗口容器的管理器
  - Panel、Applet — **FlowLayout**: 缺省的Panel布局管理器
- 选择布局管理器的方法
  - 建立布局管理器类的对象
  - 利用容器的 setLayout 为容器指定布局(即指定一个布局管理器的对象)

确定容器布局

- 改变缺省布局管理器的方法；

例：BorderLayout B=new BorderLayout();

C1.setLayout(B);

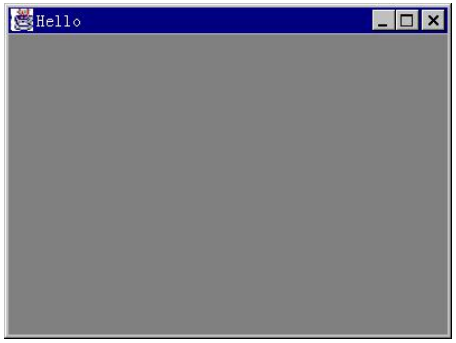
or

C1.setLayout(new BorderLayout());

例：将myFrame布局设定为FlowLayout类型

myFrame.setLayout(new FlowLayout());

Frame



Frame(String title)

构造一个新的不可见的frame

隐含的布局管理器是：BorderLayout

setLayout(…) 设置新的布局管理器。

add(Component comp)

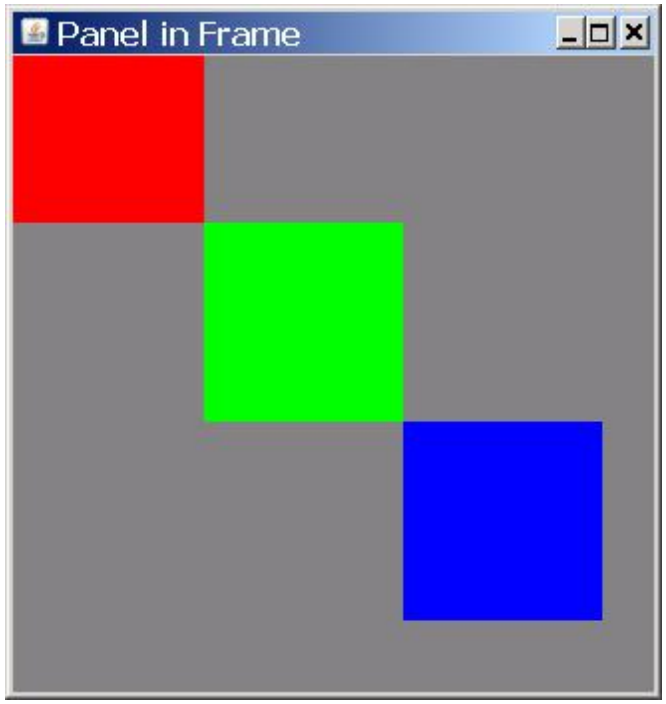
在容器上增加一个组件（在容器为不可见的状态时）。

void setSize(int width, int height) 设置组件大小

setVisible(true) 设置组件可见

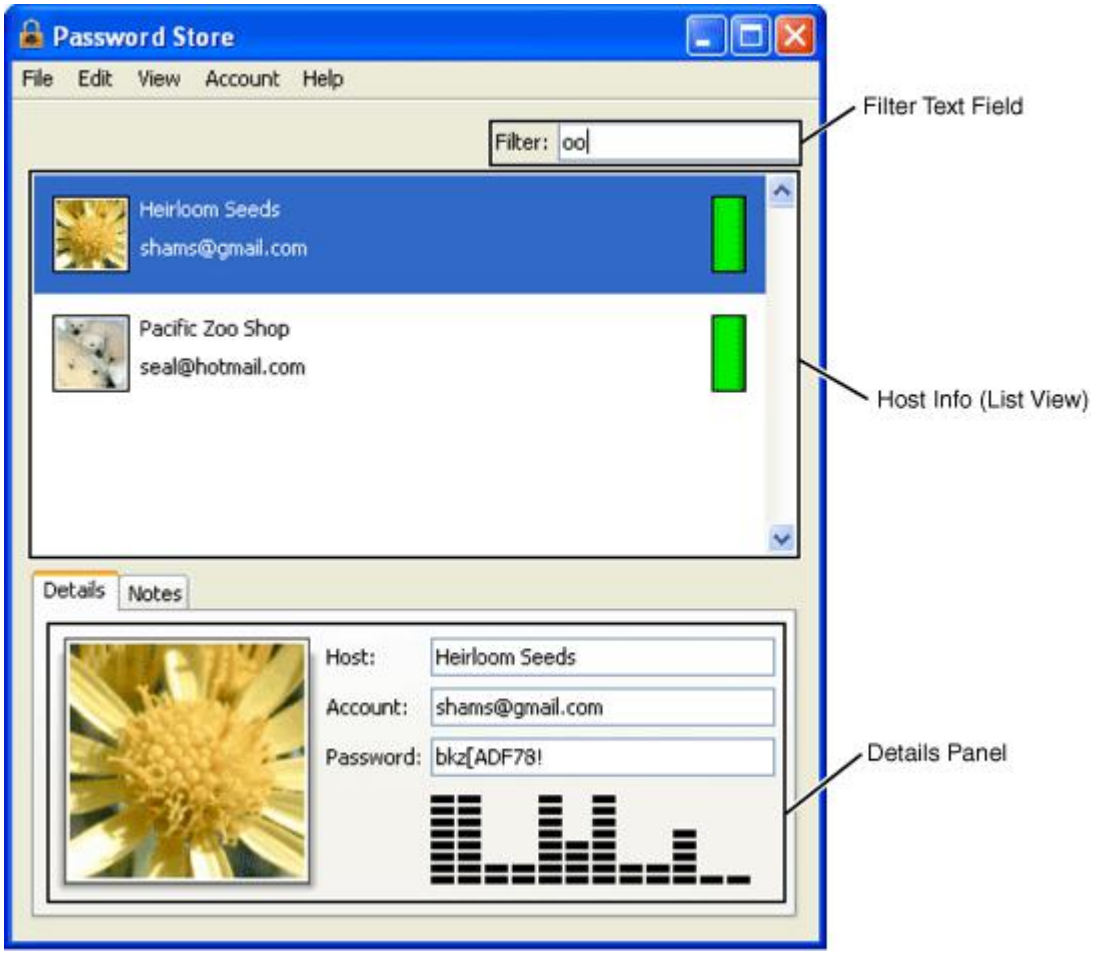
Frame — Panel（FramePanel.java）

```
Frame fm1 = new Frame("Panel in Frame");
fm1.setSize(400,350);
fm1.setBackground(Color.gray);
fm1.setLayout(null);
  Panel pn1 = new Panel();
  pn1.setSize(100,100);
  pn1.setBackground(Color.red);
  pn1.setLocation(0,50);
fm1.add(pn1);
     …
fm1.addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e) {
              System.exit(0);
          }
       });
fm1.setVisible(true);
```



AWT/Swing 例程

- PasswordStore
- demo



2017-10-26

6.4 Java的事件处理

- 什么是事件?
  - 用户用于交互而产生的键盘或鼠标动作。
  - 响应用户的动作称为处理事件。
  - 在类Event中，定义了所有的事件处理方法；
  - 小应用已经继承了这些方法。
- Java事件处理
  - 使用单一的java.awt.Event类来接受所有类型的事件；
  - 使用action(), handleEvent( )以及其他的一些事件处理方法（如鼠标事件、键盘事件、焦点事件）来进行相应事件的事件处理。
  - 缺点：造成长if分支语句组，不利于面向对象的编程。

Java1.1 对事件处理的改进

- 引入java.awt.event包
- 更加面向对象，易于理解
- 定义了事件的“发生者”和“监听者”对象
  按钮事件、文本事件、选择事件、调整事件、
  鼠标事件、键盘事件等
- 事件以类层次来表达，取代了1.0.2中的单一的
  类，并可以自定义事件类型

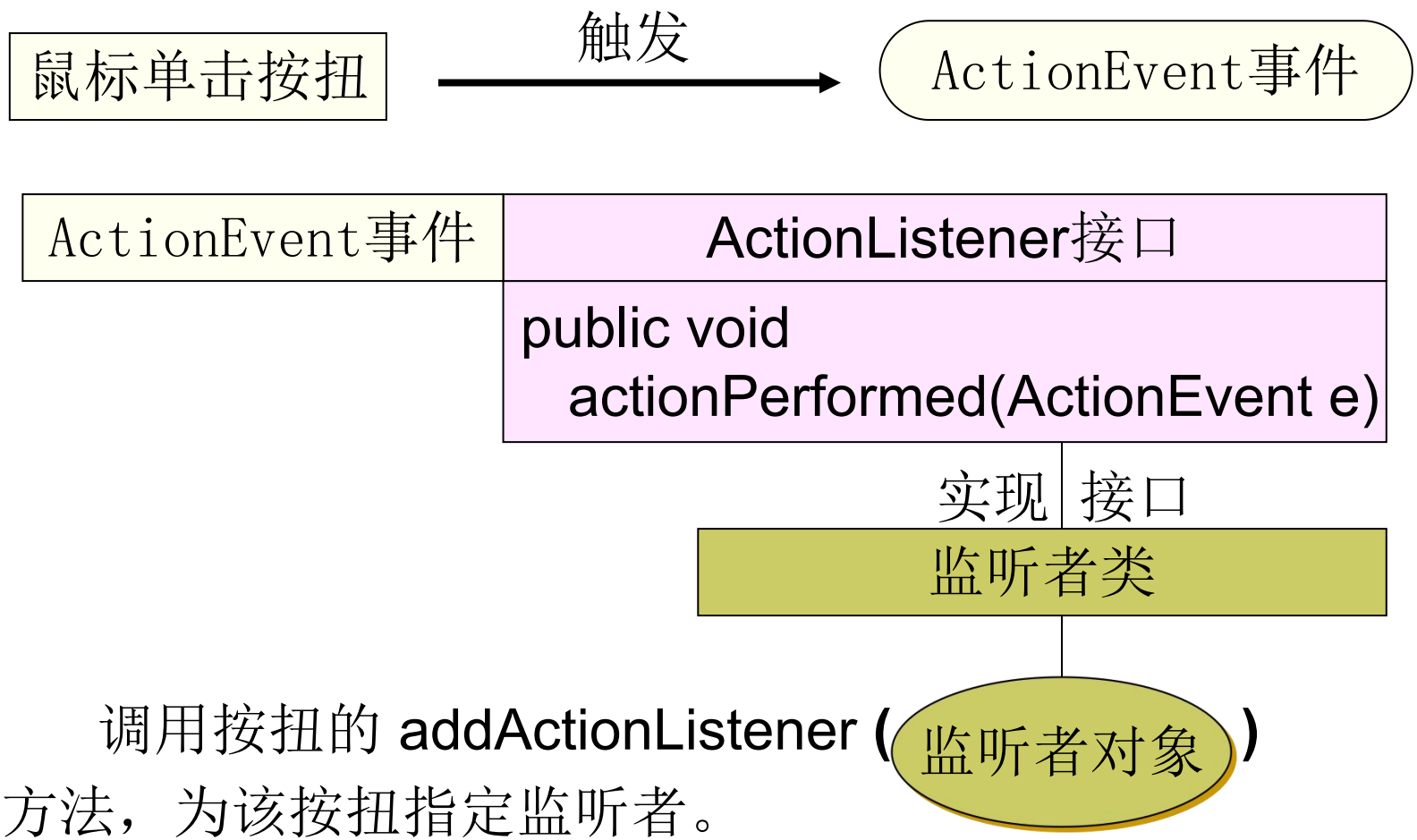
事件及处理机制

- 用户操作GUI组件时会引发各种事件。
- 事件：描述“发生了什么事情”的对象。
  系统根据用户的操作构造出相应事件类的对象。
- 事件源：事件的产生地。
- 事件处理程序：是一个方法，它接收一个事件对象、分析它、并完成对该事件的处理。
- 每个事件有一个相应的监听者接口，它规定了能够接收（并处理）该类事件的方法的规范。
- 监听者：实现了监听者接口的类，它包含有事件处理程序。
- 编程人员要为事件源指定监听者对象（即指定处理某种事件的事件处理程序）。

事件处理机制

- 组件（事件发生者）触发一个相应类型的事件
- 此事件由相应类型的Listener（事件监听者）接收并处理

事件及处理机制



调用按扭的 addActionListener ( ) 方法，为该按扭指定监听者。

实现步骤

- 根据需要定义相应类型的监听者类，在类的定义中完成事件的处理（建议使用inner class定义）；
- 创建事件监听者对象；
- 为将会触发事件的组件C注册相应的事件监听者对象（使用C的addXXXListener()方法）。
- 例：ButtonAct.java

```
public class ButtonAct extends Frame {
   … … …
   public ButtonAct() {
         … … … //为b1注册事件监听者B1
         b1.addActionListener(new B1());
         add(b1);
         … … …
   }
//利用inner class结构定义监听者类
   class B1 implements ActionListener {
//利用actionPerformed方法进行事件处理
         public void actionPerformed(ActionEvent e){
                  who.setText("Button 1");
         }
   }
   … … …
} ///:~
```

事件与监听者类型

- XXXListener接口与XXXEvent
  - ✓ XXX为特定类型
  - ✓ 常用类型：

    Action, Focus, Key, Window,

    MouseEvent vs.

    Mouse/MouseMotionListener
- 注册与取消：组件包含的新方法

  addXXXListener()与removeXXXListener()
- 参考：eventtables.html

监听者接口中有什么？

- 每种Listener接口均定义了一套abstract方法，编程者必须在监听者类中实现这些方法来做事件处理；
- 例：

ActionListener: actionPerformed()

WindowListener:

windowOpened/Closing/Closed/Activated/Deactivated/Iconified/Deiconified()

使用Listener Adapter

- 由于接口中的方法为abstract方法，所以在监听者类中要实现所有的方法，较为烦琐。
- 为了简化编程，引入了Adapter。具有两个以上方法的监听者接口均对应一个XXXAdapter类，提供了接口中每个方法的缺省实现。

使用Listener Adapter（续）

- 例：

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

- 参考：MouseEventApplet_1.html
- 例：MouseEventApplet_1.java
- 基本训练
  J:\JavaEx\tutorial\uiswing\index.html

事件类

- ActionEvent类：属于动作事件，包括：点击按钮；双击一个列表中的选项;选择菜单项；在文本框中按回车。最典型的例子是：鼠标点击按钮所产生的事件，经ActionListener接口触发一连串动作，再由actionPerformed()方法完成这些动作。
  - e.getSource()　返回事件发生的对象(名)
  - e.getActionCommand()　返回按扭名(动作事件的命令字符串)。
- ActionListener 接口
  - public void actionPerformed(ActionEvent e)

图形用户界面例

```
import java.awt.*;
import java.awt.event.*;
public class a  {
  public static void main(String args[]){
    MyFrame form1 = new MyFrame();
  }
}
class MyFrame extends Frame  {
  MyFrame()    {
    super("窗口标题");
    setSize(200,100);
    setFont(new Font("TimesRoman",Font.BOLD,20));
    setLayout(new FlowLayout());
    Button  b = new Button("close");
    add(b);
    b.addActionListener(new Bprocess());
    setVisible(true);
   }
}
 class Bprocess  implements  ActionListener {
      public void actionPerformed(ActionEvent e) {
          System.exit(0);
      }
 }
```

监听者类

- 某个类可以兼职也可以专职做这件事（实现监听者接口）。
- 一个类若实现一个接口，必须实现接口的全部方法。

  例如：ActionListener接口只有一个方法：

  void actionPerformed(ActionEvent e)
- 实现接口的类可以是：
  - ✓ 一般类
  - ✓ 内部类
  - ✓ 匿名类

事件处理有关问题

- 有哪些事件类 (java.awt.event) ？
- 每个事件类的监听者接口？
- 每个监听者接口规定的方法？
- 多方法的监听者接口的适配器（实现接口的抽象类）是什么？
- 组件的事件委托（注册）方法？

Listeners that All Swing Components Support

- component listener Listens for changes in the component's size, position, or visibility.
- focus listener Listens for whether the component gained or lost the keyboard focus.
- key listener Listens for key presses; key events are fired only by the component that has the current keyboard focus.
- mouse listener Listens for mouse clicks, mouse presses, mouse releases and mouse movement into or out of the component's drawing area.
- mouse-motion listener Listens for changes in the mouse cursor's position over the component.
- mouse-wheel listener Listens for mouse wheel movement over the component.
- Hierarchy Listener Listens for changes to a component's containment hierarchy of changed events.
- Hierarchy Bounds Listener Listens for changes to a component's containment hierarchy of moved and resized events.

Listener实现

https://docs.oracle.com/javase/tutorial/uiswing/events/caretlistener.html

- How to Write an Action Listener
- How to Write a Caret Listener
- How to Write a Change Listener
- How to Write a Component Listener
- How to Write a Container Listener
- How to Write a Document Listener
- How to Write a Focus Listener
- How to Write an Internal Frame Listener
- How to Write an Item Listener
- How to Write a Key Listener
- How to Write a List Data Listener
- How to Write a List Selection Listener
- How to Write a Mouse Listener
- How to Write a Mouse-Motion Listener
- How to Write a Mouse-Wheel Listener
- How to Write a Property Change Listener
- How to Write a Table Model Listener
- How to Write a Tree Expansion Listener
- How to Write a Tree Model Listener
- How to Write a Tree Selection Listener
- How to Write a Tree-Will-Expand Listener
- How to Write an Undoable Edit Listener
- How to Write Window Listeners

窗口事件

- 关闭窗口框时引发 Window**Event**事件
- 委托 **add**WindowListener(new Wclose());
- 定义监听者类

```
class Wclose  implements  Window Listener {
 public void windowClosing(WindowEvent e) {
        System.exit(0);   }
```

- WindowListener类有7 个方法，必须都实现
- Window**Adapter** 接收窗口事件的抽象适配器类，用空内容实现了WindowListener接口的所有方法
- class Wclose extends WindowAdapter {…}

ActionEvent动作事件

- 引发原因：
  - 单击按扭，双击列表框中选项，选择菜单项，文本框中的回车
- 事件监听接口：ActionListener
- 接口方法：
  - actionPerformed(ActionEvent e)
- 组件注册该事件方法：
  - addActionListener(监听者）

TextEvent文本事件

- 引发原因：
  - 文本框或文本区域内容改变
- 事件监听接口：TextListener
- 接口方法：
  - textValueChanged(TextEvent e)
- 组件注册该事件方法：
  - addTextListener(监听者）

ItemEvent选择事件

- 引发原因：
  - 改变列表框中的选中项
  - 改变复选框选中状态
  - 改变下拉选单的选中项
- 事件监听接口：ItemListener
- 接口方法：
  - itemStateChanged(ItemEvent e)
- 组件注册该事件方法：
  - addItemListener(监听者）

AdjustmentEvent调整事件

- 引发原因：
  - 操作滚动条改变滑块位置
- 事件监听接口：AdjustmentListener
- 接口方法：
  - adjustmentValueChanged(AdjustmentEvent e)
- 组件注册该事件方法：
  - addAdjustmentListener(监听者）

KeyEvent事件

- 引发原因：
  - 敲完键（KEY-TYPED)
  - 按下键 (KEY-PRESSED)
  - 释放键 (KEY-RELEASE)
- 事件监听接口：KeyListener
- 接口方法：
  - keyPressed(KeyEvent e)　　键已被按下时调用
  - keyReleased(KeyEvent e)　　键已被释放时调用
  - keyTyped(KeyEvent e)　　键已被敲完时调用
- KeyEvent方法：char ch = e.getKeyChar();
- 事件监听适配器（抽象类）KeyAdapter
- 组件注册该事件方法：
  - addKeyListener(监听者）

MouseEvent事件

- 引发原因: (鼠标作用在一个组件上)
  - 鼠标事件: 鼠标键按下，鼠标键抬起，单击鼠标，鼠标光标进入一个组件，鼠标光标离开一个组件。
  - 鼠标移动事件: 鼠标移动，鼠标拖动
- 鼠标事件监听接口1:
  - MouseListener　　接受鼠标事件
- 该接口方法:
  - mouseClicked(MouseEvent e)
  - mouseEntered(MouseEvent e)　鼠标光标进入一个组件
  - mouseExited(MouseEvent e)　鼠标光标离开一个组件

MouseEvent事件（续）

- 鼠标事件监听适配器（抽象类）MouseAdapter
- 鼠标事件监听接口2:
  - MouseMotionListener　　接受鼠标移动事件
- 该接口方法：
  - mouseMoved(MouseEvent e)　鼠标光标在组件上移动
  - mouseDragged(MouseEvent e)　用鼠标拖动一个组件
- 鼠标移动事件监听适配器 MouseMotionAdapter
- 组件注册鼠标事件方法:
  - addMouseListener(监听者）
- 组件注册鼠标移动事件方法:
  - addMouseMotionListener(监听者）

MouseEvent事件（续）

- MouseEvent方法
  - e.getClickCount()　　=1 单击　=2 双击
  - Point e.getPoint()　　取鼠标光标位置
  - int e.getX()　int e.getY()　　取鼠标光标位置
  - e.getModifiers() = e.BUTTON1_MASK　鼠标左键
    = e.BUTTON3_MASK　鼠标右键

WindowEvent事件

- 引发原因：
  - 有关窗口操作引发的事件
- 事件监听接口WindowListener
- 接口方法
  - windowActivated(WindowEvent e)
    激活窗口
  - windowClosed(WindowEvent e)
    调用dispose方法关闭窗口后
  - windowClosing(WindowEvent e)
    窗口正在被关闭(试图利用窗口关闭按钮关闭窗口)
  - windowDeactivated(WindowEvent e)
    本窗口成为非活动窗口

WindowEvent事件(续）

- windowDeiconified(WindowEvent e)
  窗口从最小化恢复为普通窗口
- windowIconified(WindowEvent e)
  窗口变为最小化图标
- windowOpened(WindowEvent e)
  窗口被打开成为可见时

- 接口适配器 **WindowAdapter**
- 注册事件方法
  - addWindowListener

FocusEvent事件

- 引发原因：
  - ✓ 组件获得焦点
  - ✓ 组件失去焦点
- 事件监听接口 FocusListener
- 接口方法：
  - ✓ focusGained(FocusEvent e)　组件获得焦点时调用
  - ✓ focusLost(FocusEvent e)　组件失去焦点时调用
- 接口适配器：FocusAdapter
- 组件注册该事件方法：addFocusListener

TextEvent事件

- 引发原因：
  - 当组件（如文本框）文本改变时引发
- 事件监听接口：TextListener
- 接口方法：
  - textValueChanged(TextEvent e)
- 组件注册该事件方法：
  - addTextListener

ComponentEvent 事件

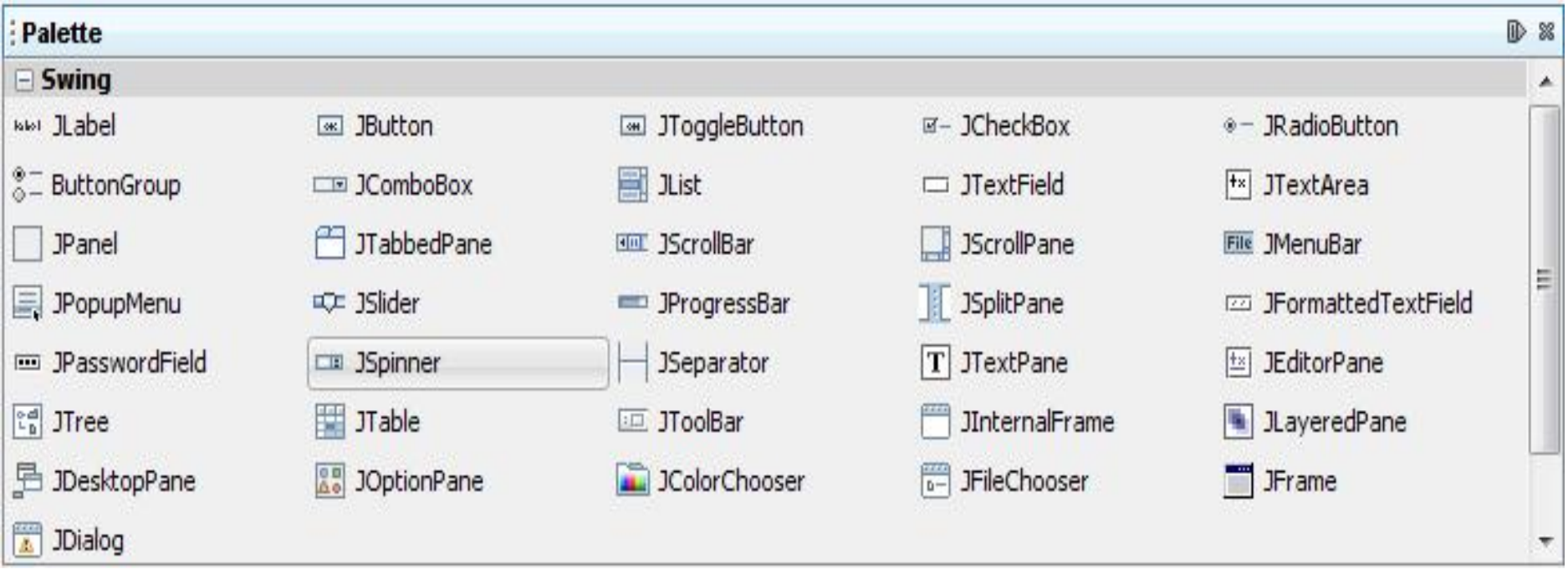
- 引发原因：
  - 当组件移动、改变大小、改变可见性时引发
- 事件监听接口：ComponentListener
- 接口方法：
  - componentHidden(ComponentEvent e) 组件隐藏
  - componentMoved(ComponentEvent e) 组件移动
  - componentResized(ComponentEvent e) 组件改变大小
  - componentShown(ComponentEvent e) 组件变为可见
- 接口适配器 ComponentAdapter
- 组件注册该事件方法：
  - addComponentListener

ContainerEvent 事件

- 引发原因：
  - 当容器内增加或移走组件时引发
- 事件监听接口：ContainerListener
- 接口方法
  - componentAdded(ContainerEvent e) 容器内加入组件
  - componentRemoved(ContainerEvent e) 从容器中移走组件
- 接口适配器 ContainerAdapter
- 容器注册该事件方法：
  - addContainerListener

2018-11-12

6.5 GUI标准组件



file:///G:/tutorial/uiswing/index.html
file:///G:/tutorial/uiswing/components/componentlist.html
file:///G:/tutorial/uiswing/examples/components/index.html#table

http://www.gwtproject.org/doc/latest/tutorial/gettingstarted.html

JAVA tutorial: http://pan.baidu.com/share/link?shareid=2779168916&uk=1444198655

tutorial/uiswing/components/componentlist.html

- How to Use Buttons, Check Boxes, and Radio Buttons
- How to Use Color Choosers
- How to Use Combo Boxes
- How to Make Dialogs
- How to Use Editor Panes and Text Panes
- How to Use File Choosers
- How to Use Formatted Text Fields
- How to Make Frames (Main Windows)
- How to Use Internal Frames
- How to Use Labels
- How to Use Layered Panes
- How to Use Lists
- How to Use Menus
- How to Use Panels
- How to Use Password Fields
- How to Use Progress Bars
- How to Use Root Panes
- How to Use Scroll Panes
- How to Use Separators
- How to Use Sliders
- How to Use Spinners
- How to Use Split Panes
- How to Use Tabbed Panes
- How to Use Tables
- How to Use Text Areas
- How to Use Text Fields
- How to Use Tool Bars
- How to Use Tool Tips
- How to Use Trees

6.5 GUI标准组件

- Label(标签)
  - 构造方法

    Label myl = new Label(“标签内容”);
    Label a = new Label(“information”);
    Label a = new Label(“information”, Label.CENTER);
  - 方法

    a.setText(“新内容”);　　设置标签内容
    String a.getText()　　读取标签内容
- Button(按钮)
  - Button myB1 = new Button();
  - Button myB2 = new Button(“Cancel”);

http://www.gwtproject.org/doc/latest/tutorial/gettingstarted.html

GUI标准组件

- Checkbox(检测盒)
  - ✓ 创建时指定检测盒的标签

```
Checkbox cb1 = new Checkbox ();
Checkbox cb2 = new Checkbox (“registered”);
Checkbox cb3 = new Checkbox (“registered”, true);
Checkbox cb4 = new Checkbox (“registered”, true, group);
```

  - ✓ 获取和设置检测盒的状态

```
cb.setState(true);
boolean b=cb.getState();
```

Checkbox组件

- 构造方法
  - Checkbox(String label)
  - Checkbox(String label, boolean state)
  - Checkbox(String label, boolean state, CheckboxGroup group)
- 方法：
  - boolean getState()
  - setState(boolean state)

GUI标准组件

- CheckboxGroup(单选按钮组)
  - 单选按钮组是一组Checkbox的集合，是将一组Checkbox 按扭组成单选按扭组件
  - 首先创建CheckboxGroup,再加入单个按钮
- 方法：
  - Checkbox getSelectedCheckbox()
    取得目前选取的复选框对象
  - setSelectedCheckbox(Checkbox box)
    设定目前选取的复选框对象

CheckboxGroup组件

- 例: setLayout(new GridLayout(3, 1));

```
CheckboxGroup cbg = new CheckboxGroup();
Checkbox  ck1 = new Checkbox(“one”, true, cbg );
Checkbox  ck2 = new Checkbox(“two”, false, cbg );
Checkbox  ck3 = new Checkbox(“three”, false, cbg );
add(ck1);
add(ck2);
add(ck3);
```

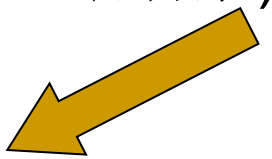
把CheckboxGroup加入容器时需要把其中的每个复选按钮逐个加入到容器中，而不能使用CheckboxGroup对象一次性地加入。

TextField行编辑框组件

- 构造方法(String, int)
  - TextField f1 = new TextField(30);
  - TextField f1 = new TextField(“abc”,30);
  - 建30个字符宽的行编辑框
- 方法：
  - String getText()　　读文本框内容
  - setText(“abc”)　　设置编辑框内容
  - setEchoChar(char c)　设置回显字符
- TextComponent 方法：
  - setEditable(false);
  - selectAll()

TextArea组件

- 构造方法(String, int , int)
  - TextArea t1 = new TestArea(10,45);
    建10行、45列的文本区域
  - TextArea t1 = new TestArea(“abc”,10,45);
  - TextArea t1 = new TestArea(“abc”,10,45，滚动条);
- 滚动条指定
  - SCROLLBARS_BOTH
  - SCROLLBARS_HORIZONTAL_ONLY
  - SCROLLBARS_VERTICAL_ONLY
  - SCROLLBARS_NONE
- 方法
  - append(String str)　　追加文本



List 列表框

- 首先创建List对象，再调用add ()方法加入List列表的各选项。列表可以实现多选多
- 构造方法
  - List(int rows, boolean multipleMode)

  rows 显示行数； multipleMode 是否允许多选
- 方法：
  - list1.add (“class A”);
  - list1.add (“class B”);
  - list1.add (“class C” , 0);　　指定加入选项位置

List 列表框

- 方法
  - int getSelectedIndex()　　取被选项索引
  - int[] getSelectedIndexes()　　取被选项索引（多选）
  - String getSelectedItem()　　取被选项
  - String[] getSelectedItems()　　取被选项（多选）
  - void select(int index)　　选中指定的项
  - String getItem(int index)　　按索引号取出该项
  - int getItemCount()　　取出项数
  - String [ ] getItems( )　　取出所有的项

Choice下拉选单

- 下拉选单提供了多选一机制，创建下拉选单包括创建下拉选单对象和添加选项两个步骤

例：

```
Choice ColorChooser = new Choice();
ColorChooser.add("Green");
ColorChooser.add("Red");
ColorChooser.add("Blue");
String getSelectedItem()
```

Scrollbar 滚动条

- 构造方法

```
mySlider = new Scrollbar(Scrollbar.HORIZONTAL ,
                0 , 1 , 0 , Integer.MAX_VALUE);
```

  - ✓ 滚动条方向
  - ✓ 滑块初始位置
  - ✓ 滑块尺寸
  - ✓ 滚动槽最小值
  - ✓ 滚动槽最大值
- 方法
  - ✓ int getValue()　返回滑块当前位置
  - ✓ setUnitIncrement(1); 设置单位增量（点按两端箭头）
  - ✓ setBlockIncrement(50); 设置块增量（点按滚动槽）

菜单

- 菜单须依附于一个实现了MenuContainer接口的对象：Frame

  构建菜单结构：创建菜单的步骤
  - ✓ 创建菜单条（MenuBar）
  - ✓ 创建菜单（Menu），加入相应菜单条
  - ✓ 创建菜单项（MenuItem），加入相应菜单
  - ✓ 使菜单条依附于拥有它的对象:setMenuBar()
- 编写响应菜单操作的代码(ActionEvent)

菜单

- 创建菜单条（MenuBar)

  MenuBar m_MenuBar = new MenuBar();
- 创建菜单（Menu），加入菜单条

  Menu menuFile = new Menu(“File”);　　//创建菜单

  m_MenuBar.add(menuFile);　　//将菜单加入菜单条
- 创建菜单项（MenuItem)，并加入相应菜单

  MenuItem　f1= new MenuItem(“Open”), //创建各菜单项

  MenuItem　f2 = new MenuItem("Close"),

  menuFile.add(f1);　　//加入菜单

  menuFile.add(f2);
- 将菜单条放入frame:

  myFrame.setMenuBar(m_MenuBar);
- 编写响应菜单操作的代码(ActionEvent)

例： MenuFrame.java

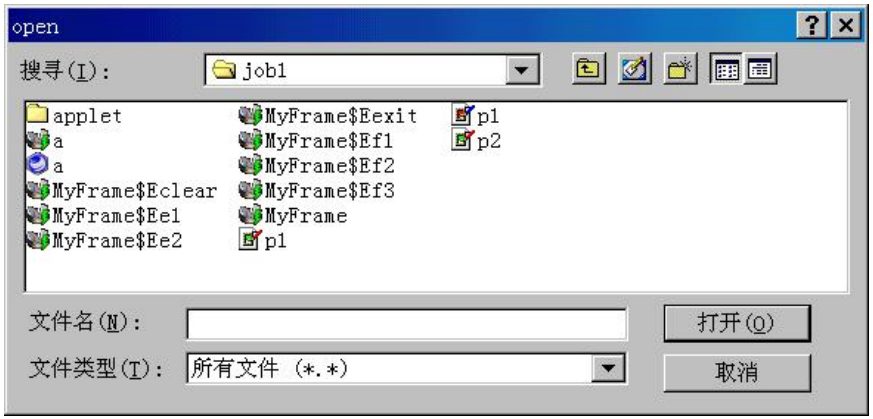
```
class MenuFrame extends Frame {
    Label lb=new Label(“here comes command”);//模拟菜单命令

    MenuBar m_MenuBar=new MenuBar();  //创建菜单条
    Menu menuFile = new Menu(“File”);  //创建菜单
    MenuItem[] file={new MenuItem(“Open”),  //创建各菜单项
                     new MenuItem("Close"),
                     new MenuItem("Exit") };

    MenuFrame() {
        m_MenuBar.add(menuFile); //把菜单加入菜单条
        menuFile.add(file[0]);  //把各菜单项加入菜单
        menuFile.add(file[1]);
        menuFile.add(file[2]);

        setMenuBar(m_MenuBar);  //把菜单条加入Frame
```

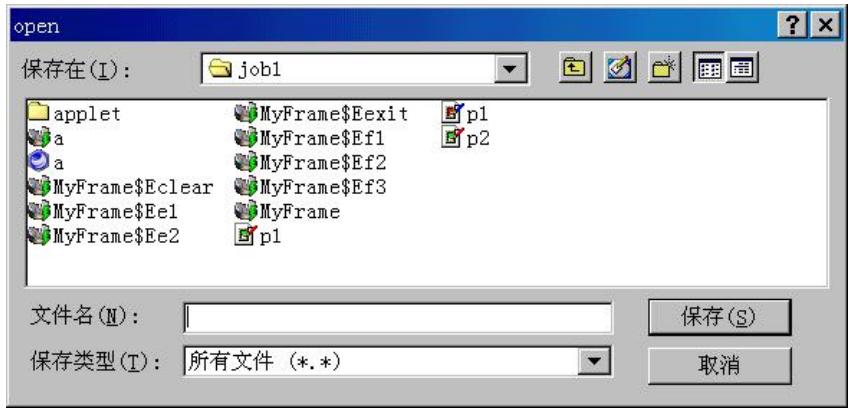
```
        setLayout(new FlowLayout());
        add(lb);  //加入Label, 用于输出菜单项命令
        ActionListener ml=new ML();
        for(int i=0;i<3;i++){  //为每个菜单项注册监听者
            file[i].addActionListener(ml); //此例多个菜单项
        }   //对应一个监听者, 但实际编程时建议使用一对一结构
        addWindowListener(new WL());
    }

    class ML implements ActionListener { //Action事件监听者类
        public void actionPerformed(ActionEvent e) {
            lb.setText(e.getActionCommand());  //输出菜单命令
        }
    }
    … … … …
}
```

文件对话框

FileDialog(Frame parent, String title, int mode)

- parent　对话框所属窗体
- title　对话框标题
- mode 对话框模式
  - ✓ FileDialog.LOAD　打开文件
  - ✓ FileDialog.SAVE　保存文件





文件对话框

```
class MyFrame extends Frame{
 …
MyFrame getMyFrameIns(){   return this;  }
```

子类方法代码片段：

```
FileDialog f = new FileDialog(
      getMyFrameIns(),"open",FileDialog.SAVE);
f.setVisible(true);
String fname = f.getDirectory() + f.getFile() ;
ig = getToolkit().getImage(fname);
```

参考资料

- https://developers.google.com/web-toolkit/tools/download-gwtdesigner?hl=zh-CN

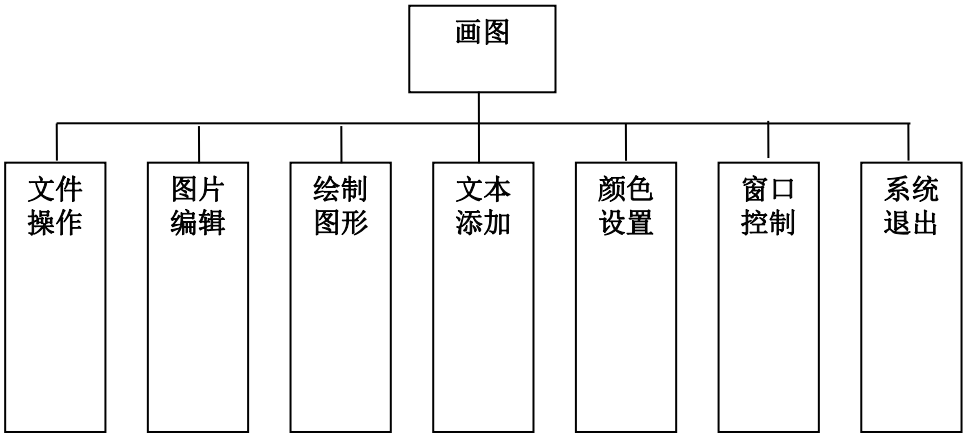
(原instantiations.com (或swt-designer.com）已被收购)

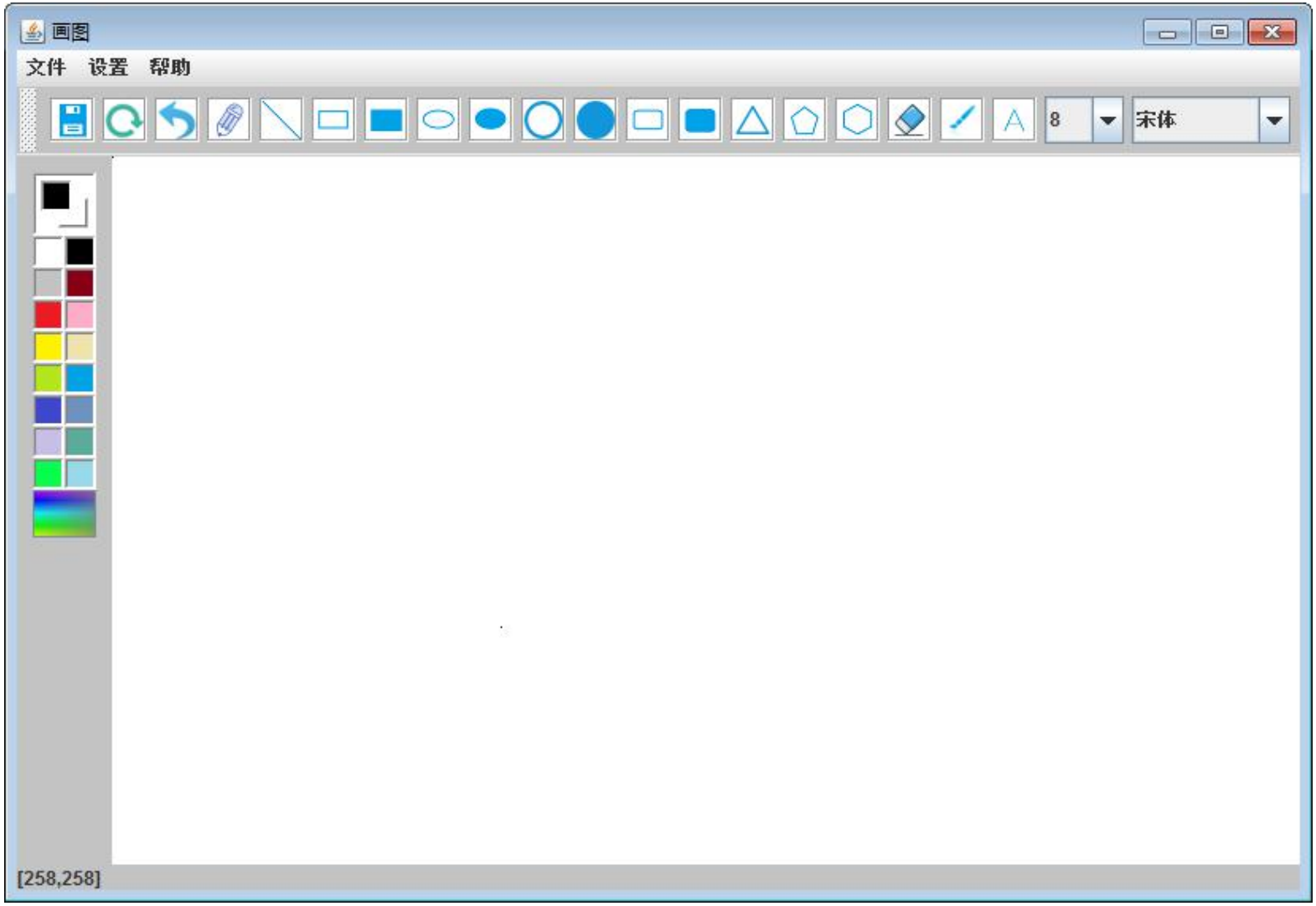
http://code.google.com

- Java tutorial

6.6 案例：画图软件

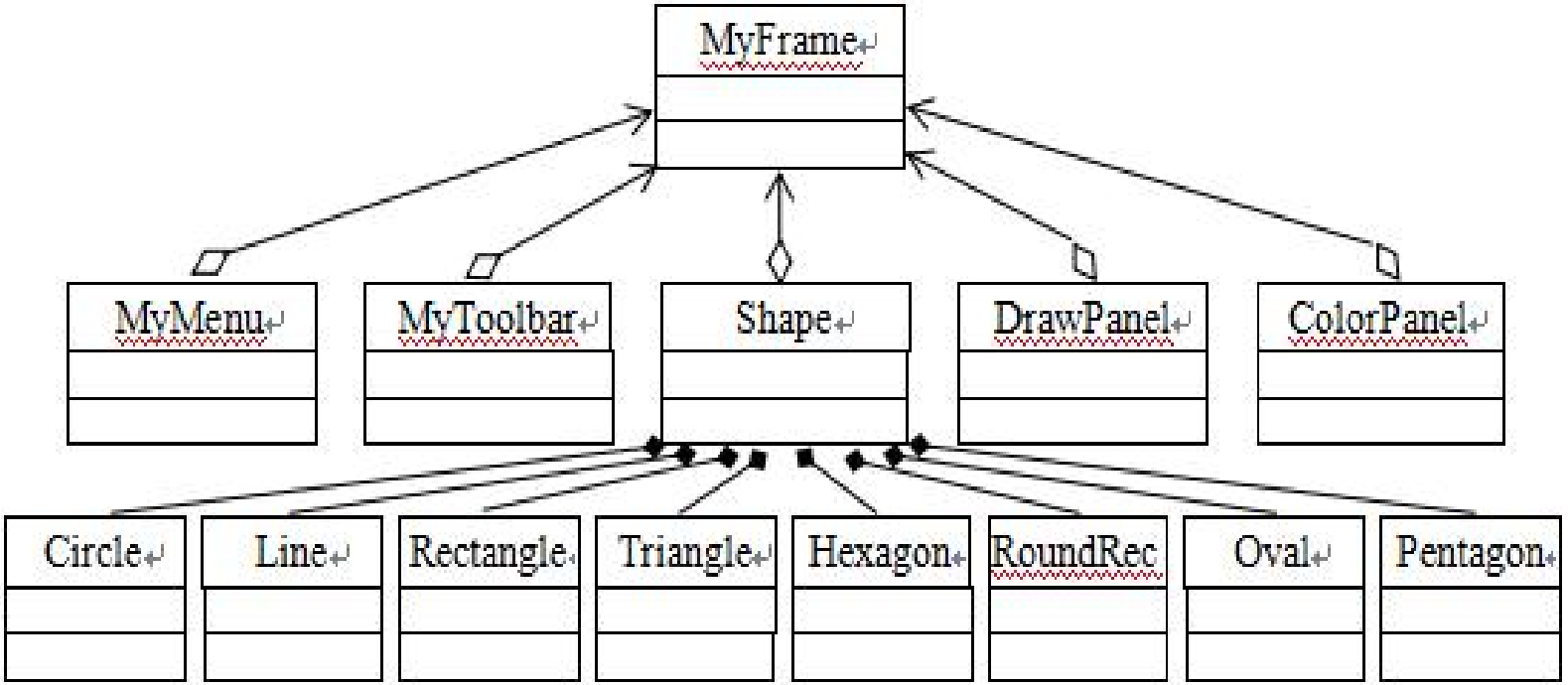
- 利用Java程序设计并实现一个画图软件，画图软件具有美观的用户界面。能够利用实现的画图软件实现直线、曲线、圆、椭圆、三角形、矩形、五边形和六边形等基本图形的绘制功能，以及绘制填充图形，添加文本功能，实现橡皮檫功能，对绘制图进行修改、撤销等功能，设置画笔的粗细和颜色，以及图片jpg、bmp、gif和png格式的存储和读取功能。
- 通过分析，利用Java图形用户界面设计实现画图软件，主要包括：常用图形绘制、添加文字、颜色设置、编辑和修改功能，以及画图软件界面控制功能。系统功能结构如图6-20所示




画图
文件
设置
帮助
8
宋体
[258,258]

系统类之间的层次关系

系统设计一个抽象图形绘制Shape类，包含绘制图形的基本属性和抽象方法draw(Graphics2D g)；绘制图形类继承Shape类并重载draw()方法。系统类之间的层次关系如图6-22所示。



课后作业2

1. tutorial/uiswing/examples/components/index.html#table 表中选择2个组件例子，进行汉化。
2. 计算器设计
3. 记事本设计
4. 停车场计费程序

上机实习

- 通知：
  - 作业提交到 yexjclass@126.com 专用邮箱。