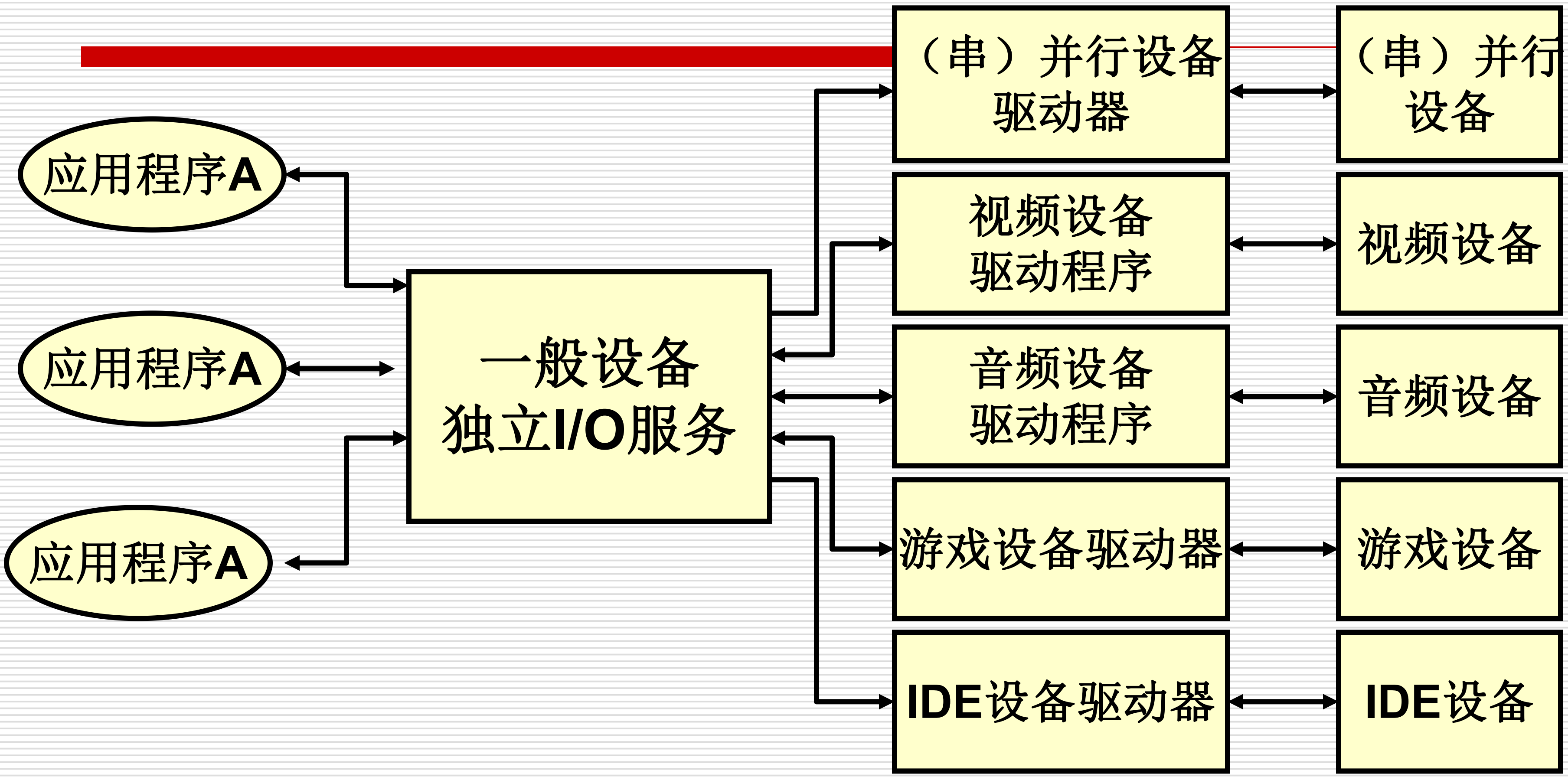
Unit 4. Advanced Class Implementation

4.1 节问题簿

- Java IO的原理：介质流与过滤流之间的关系是否理解？
- 面向字节流的基本数据类型数据的读写是否掌握？
- 面向字符流的文件和控制台的读写是否掌握？

- 程序员通常需要操作I/O设备，提供有用的输入/输出操作。
- 不同的I/O设备有不同的特性，例如：键盘、鼠标、显示器、打印机、扫描仪、摄像头，以及硬盘等各种辅存设备。

应用程序 操作系统 设备驱动程序 物理设备



- C++和Java设计者对输入/输出提出了一种方案，即输入/输出操作是一种基于字节流的操作, 输入输出包括以下3个方面的内容：
  - （1）对系统指定的标准设备的输入和输出。即从键盘输入数据，输出到显示器屏幕。这种输入输出称为标准的输入输出，简称标准I/O。
  - （2）以外存磁盘文件为对象进行输入和输出，即从磁盘文件输入数据，数据输出到磁盘文件。以外存文件为对象的输入输出称为文件的输入输出，简称文件I/O。
  - （3）对内存中指定的空间进行输入和输出。通常指定一个字符数组作为存储空间(实际上可以利用该空间存储任何信息)。这种输入和输出称为字符串输入输出，简称串I/O。

4.1.1 Java IO system(调用不同的类实现不同的功能)

- Java I/O libraries use the abstraction of a ***stream***, which represents any **data source** or **sink** as an object capable of producing or receiving pieces of data.
- The stream hides the details of what happens to the data inside the actual I/O device.

1. Java IO package (java.io.*)

- 8-bit byte stream
  - InputStream and OutputStream
  - 这类流以字节为处理单位。
- 16-bit Unicode stream
  - Reader and Writer
  - 这类流以16位的Unicode编码表示的字符为处理单位。

2. 抽象类 InputStream

- 从输入流(数据源)中读取数据，常用方法：
  - public abstract int read() throws IOException
  - public int read(byte[ ] b) throws IOException
  - public int read(byte [ ] b, int off, int len) throws IOException
  - public int available() throws IOException
  - public long skip(long n) throws IOException
- 关闭输入流
  - public void close() throws IOException

3. 抽象类 OutputStream

- 输出数据到相应的输出流(输出的目标)
  - public void write(int b) throws IOException
  - public void write(byte[ ] b) throws IOException
  - public void write(byte [ ] b, int off, int len) throws IOException
- 刷空所有输出流，并输出所有被缓存的字节到相应的输出流。
  - public void flush() throws IOException
- 关闭输出流
  - public void close() throws IOException

3. 抽象类 OutputStream(cont.)

- InputStream and OutputStream是抽象类。
  - By inheritance, Kind of Stream(Media Stream)： 通过继承上述抽象类，可以创建与存储介质直接连接的流。即表示流在哪一个介质上面发生。
- Kind of InputStream(继承InputStream)
  - FileInputStream
  - StringBufferInputStream
  - ByteArrayInputStream

$\left.\begin{array}{l}\text{FileInputStream}\\\text{StringBufferInputStream}\\\text{ByteArrayInputStream}\end{array}\right\}$ 继承InputStream

4. Kind of InputStream(cont.)

- StringBufferedInputStream
  - 将String作为数据来源；
- ByteArrayInputStream
  - 允许内存的一个缓冲区当作输入流使用；
- FileInputStream
  - public class FileInputStream extends InputStream
  - 例如构造函数：FileInputStream(String name)
  - 读取文件中的信息，以文件作为数据来源；
- For Example FileInputStreamDemo.java

文件名

4. Kind of InputStream(cont.)

- System.in
  - is the "standard" input stream.
  - Typically, this stream corresponds to keyboard input.
  - 将键盘输入的数据作为输入流使用。
  - in是类 System的静态变量,
    - 即 public static final InputStream in
- SystemIn.java

5. Kind of OutputStream

- **Kind of OutputStream**
  - **ByteArrayOutputStream**
  - **FileOutputStream**

$\left.\begin{array}{l}\text{ByteArrayOutputStream}\\ \text{FileOutputStream}\end{array}\right\}$ 继承OutputStream

- ByteArrayOutputStream
  - 建立一块位于内存中的缓冲区。所有输出的数据都被置于该缓冲区中。

5. Kind of OutputStream(cont.)

- FileOutputStream

文件名

  - **public class FileOutputStream extends OutputStream**
  - 例如构造函数：**FileOutputStream(String name)**
    - 将输出数据写至文件；
- For example
  - **FileOutputStreamDemo.java**

6. Filtered Stream

- 类FilterInputStream和FilterOutputStream加工inputStream和OutputStream以提供更多的功能(操作)。
  - java-io.png
- 类FilterInputStream 和 inputStream的关系：
  - 继承关系： FilterInputStream继承inputStream（过滤输入流具备输入流的功能）
  - 关联关系： FilterInputStream维护一个私有的关联属性：-inputStream（过滤输入流加工的对象），通　　　　过Filtered Stream构造函数的参数初始化该私有属性。

6. Filtered InputStream

- 继承FilterInputStream的类
  - ✓ BufferedInputStream
  - DataInputStream

6. Filtered InputStream(cont.)

- BufferedInputStream
  - BufferedInputStream(InputStream in)
    - 为读取的数据申请一份可用的缓冲区，用于提高输入处理的效率。运用它便可避免“每次想要取得数据时都得进行实际读取动作”。它所代表的意义是“使用缓冲区”。…
- DataInputStream
  - DataInputStream(InputStream in)
  - 从流中读取基本类型(int、char、long等)的数据.
    - 含有一份完整接口(public方法)，让你得以读取各种基本类型的数据（查看帮助文档）。

```
DataInputStream in =
    new DataInputStream(
        new BufferedInputStream(
            new FileInputStream("Data.txt") ) ) ;
```

- For example IOByteStream.java

7. Filtered outputStream

- 继承FilterOutputStream的类
  - ✓ BufferedOutputStream
  - DataOutputStream
  - PrintStream
- 类FilterOutputStream 和 OutputStream的关系：
  - 继承关系： FilterOutputStream继承outputStream
  - 关联关系： FilterOutputStream维护一个私有的关联属性：-outputStream；

7. Filtered outputStream(cont.)

- BufferedOutputStream
  - BufferedOutputStream(OutputStream out)
    - 避免每次想要写入数据时都得执行实际的写入动作。它所代表的意义正是“使用缓冲区”。可以调用flush()来清出缓冲区内容。
- DataOutputStream
  - DataOutputStream(OutputStream out)
  - 可以将各种基本类型的数据写至流。含有一份完整接口(public 方法)，让你得以写各种基本类型的数据(查看帮助文档)。

4.1.1 Java IO system

```
DataOutputStream out =
    new DataOutputStream(
       new BufferedOutputStream(
         new FileOutputStream("Data.txt") ) );
```

- For example IOByteStream.java

7. Filtered outputStream(cont.)

- PrintStream extends FilterOutputStream
  - PrintStream(OutputStream out)
  - 产生格式化的输出结果，处理的是数据的显示。
    - 可以打印各种基本类型的数据和字符串类型的数据等
      - 查看帮助文档
- out是类 System的PrintStream类型的静态变量, 即

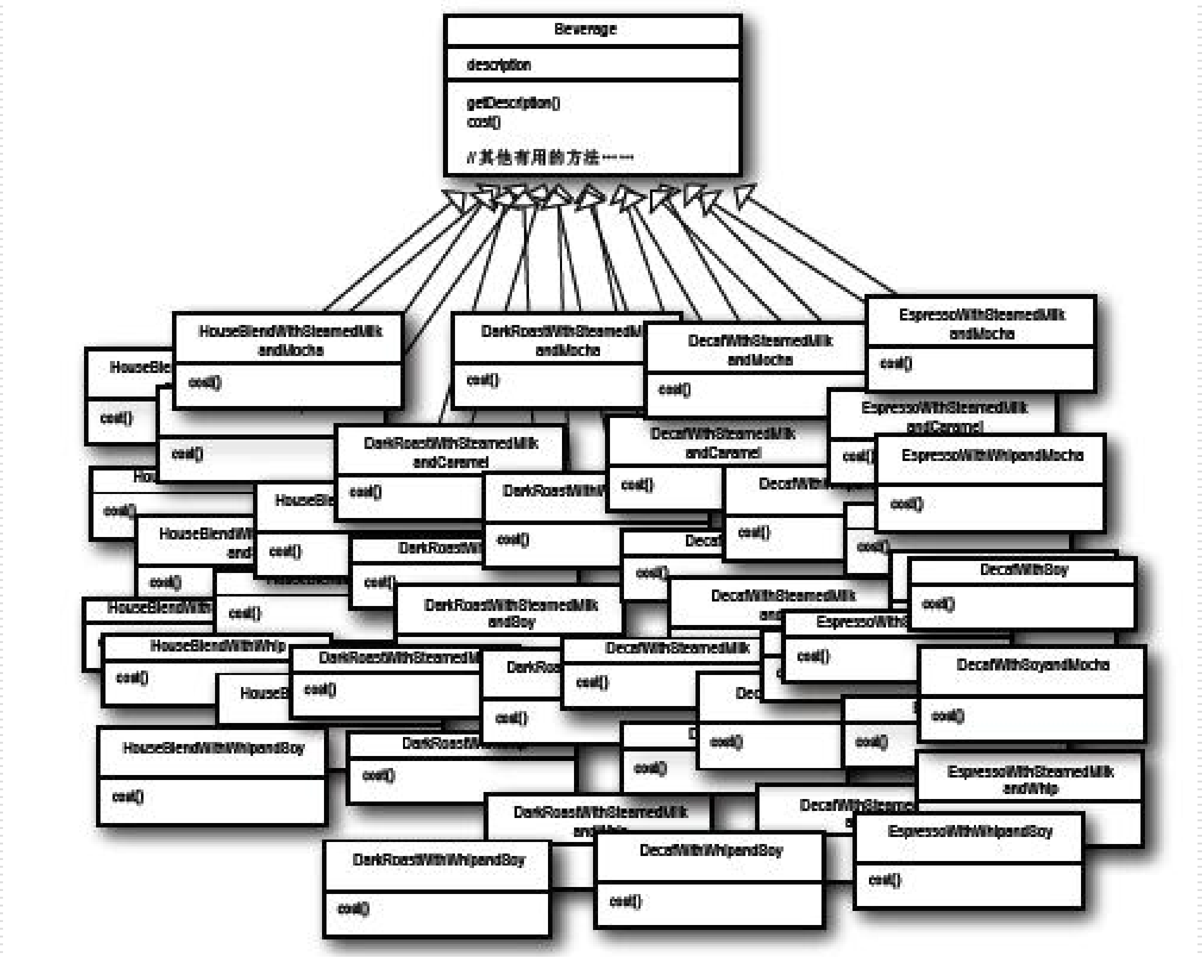
  public static final PrintStream out

  - System.out.println(…);
  - System.out.print(…);
  - System.out 将输出的数据打印到显示屏上。

Decorator设计模式

- Java I/O 程序库(java.io.*)的设计使用了Decorator设计模式。
  - 当为满足各种必要的功能组合，造成单纯的继承动作产生大量subclass时，往往使用Decorator设计模式。
  - Java I/O 程序库往往需要不同功能的组合，这也正是为什么要使用Decorator设计模式的原因。

Decorator设计模式(续)


Beverage
description
getDescription()
cost()
// 其他有用的方法……
HouseBlendWithSteamedMilk andMocha
DarkRoastWithSteamedMilk andMocha
DecafWithSteamedMilk andMocha
EspressoWithSteamedMilk andMocha
EspressoWithSteamedMilk andCaramel
DarkRoastWithSteamedMilk andCaramel
DecafWithSteamedMilk andCaramel
EspressoWithWhipandMocha
DecafWithSoy
DarkRoastWithSteamedMilk andSoy
DecafWithSteamedMilk
HouseBlendWithWhip
DecafWithSoyandMocha
HouseBlendWithWhipandSoy
EspressoWithSteamedMilk andWhip
EspressoWithWhipandSoy
DarkRoastWithSteamedMilk andWhip
DarkRoastWithWhipandSoy
DecafWithWhipandSoy
cost()

Decorator设计模式(续)

- **Decorator设计模式：**
  - 在原有的基础上，每添加一个装饰，就可以增加一种功能。
  - 动态地给一个对象添加一些额外的职责(在原来对象的基础上提供更多的功能)。就扩展功能而言， 它比生成子类方式更为灵活。
- Decorator.png

4.1.1 Java IO system

- Java中的IO允许程序员无限扩展IO的功能，实现想要的功能，需要4个步骤：
  - 1. 创建两个分别继承了FilterInputStream和FilterOutputStream的子类
  - 2. 重写read()和write()方法来实现自己想要的功能。
  - 3. 可以定义或者重写其它方法来提供附加功能。
  - 4. 这两个类由于在功能上是对称的，它们要一起被使用。
- 通过上述步骤可以无限扩展IO的功能。

8. 抽象类Readers & Writers

- Java 1.1 对其核心的I/O stream程序库进行了十分重大的改变；加入Reader和Writer提供对16-bit Unicode字符的支持。新程序库的设计与旧程序库相比能够提供更快的速度。

抽象类 Reader

- 从输入流(数据源)中读取数据
  - public int read() throws IOException
  - public int read(char[ ] cbuf) throws IOException
  - public int read(char[ ] cbuf, int off, int len) throws IOException
  - public long skip(long n) throws IOException
- 关闭输入流
  - public abstract void close() throws IOException

抽象类 Reader及其子类(cont.)

- FileReader (对比：FileInputStream)
  - public class FileReader extends InputStreamReader
  - FileReader(String fileName)
- StringReader(对比：StringBufferInputStream)
  - public class StringReader extends Reader
  - StringReader(String s)
- CharArrayReader(对比：ByteArrayInputStream)
  - public class CharArrayReader extends Reader
  - CharArrayReader(char[] buf)
- For example
  - 比较：FileInputStreamDemo.java

抽象类 Writer(cont.)

- 输出数据到相应的输出流(输出的目标)
  - public void write(int c) throws IOException
  - public void write(char[ ] cbuf) throws IOException
  - public void write(char[ ] cbuf, int off, int len) throws IOException
  - public void write(String str) throws IOException
  - public void write(String str,int off, int len) throws IOException
- 刷空所有输出流，并输出所有被缓存的字节到相应的输出流。
  - public abstract void flush() throws IOException
- 关闭输出流
  - public abstract void close() throws IOException

抽象类 Writer及其子类

- FileWriter(对比：FileOutputStream)
  - public class FileWriter extends OutputStreamWriter
  - FileWriter (String fileName)
- StringWriter (对比： no corresponding class)
  - public class StringWriter extends Writer
  - StringWriter()
- CharArrayWriter(对比： ByteArrayOutputStream)
  - public class CharArrayWriter extends Writer
  - CharArrayWriter (char[] buf)
- 比较：FileOutputStreamDemo.java

Filter class

BufferedInputStream	✓ BufferedReader
DataInputStream	Use DataInputStream (Except when you need to use readLine( ), you should use a BufferedReader)
BufferedOutputStream	BufferdWriter
DataOutputStream	无对应类
PrintStream	PrintWriter

BufferedReader

- BufferedReader
  - public BufferedReader(Reader in)
  - public String readLine() throws IOException
    - Returns: A String containing the contents of the line, not including any line-termination characters, or null if the end of the stream has been reached.
  - when you need to use readLine( ), you should use a BufferedReader

BufferedReader(cont.)

- System.in

✓
```
BufferedReader  stdIn =
        new BufferedReader(new
                InputStreamReader(System.in))
```

✓
```
String  input = stdIn.readLine();
```

“桥接(bridge)”类

- Java提供的“桥接(bridge)”类允许将“byte”继承体系和“character”继承体系中的classes搭配运用。
  - 桥接类“InputStreamReader”将InputStream转换为Reader;（如何转？？）
    - public class InputStreamReader extends Reader
    - InputStreamReader(InputStream in)，通过构造函数将InputStream转换为Reader
  - 桥接类“OutputStreamWriter”将OutputStream转换为Writer; （？？）

BufferedReader(cont.)

- 从文件中读取一行字符串：
- BufferedReader fileIn =

```
BufferedReader fileIn =
            new BufferedReader(
                   new FileReader( “filename” ));
```

  - the FileReader constructor not only creates an object, it opens the specified file for reading.
  - The FileReader constructor that takes a filename as argument can throw a FileNotFoundException.
  - 文件所在目录：e://文件1/文件夹2/filename

BufferedReader(cont.)

```
String  line =   fileIn.readLine();
    while (line != null)  {
   // process line
    line = fileIn.readLine();
}
```

- Whenever you want to use readLine( ), you should use a BufferedReader. Other than this, DataInputStream is still a “preferred” member of the I/O library.

Filter class

◆ BufferedInputStream	✓ BufferedReader
◆ DataInputStream	◆ Use DataInputStream (Except when you need to use readLine( ), you should use a BufferedReader)
◆ BufferedOutputStream	✓ BufferdWriter
◆ DataOutputStream	◆ 无对应类
◆ PrintStream	✓ PrintWriter

PrintWriter

- PrintWriter
  - public class PrintWriter extends Writer
  - 构造函数:
    - ✓ PrintWriter(Writer out)
    - ✓ PrintWriter(Writer out, boolean autoFlush)
    - ✓ PrintWriter(OutputStream out)
    - ✓ PrintWriter(OutputStream out, boolean autoFlush)

PrintWriter(cont.)

- System.out is a PrintStream
  - PrintWriter has a constructor that takes an OutputStream as an argument.
    - PrintWriter stdOut =
      new PrintWriter(System.out, true);
    - stdOut.println("A line of output.");

PrintWriter(cont.)

- 写数据到指定的文件：
- PrintWriter fileOut =

  new PrintWriter(new FileWriter(“filename”));
  - fileOut.println("A line of output.");
    - When a file is opened for writing and the specified file does not exist, a new file, initially empty, is created by the host system.
    - When the one-argument FileWriter constructor opens a file, it erases the contents of the file.
  - 例如：CopyFile.java

PrintWriter(cont.)

- When data needs to be *appended* to a file, the two-argument FileWriter constructor should be used:
  - If the second argument passed to this constructor is true, the contents of the file are preserved when the file is opened. Otherwise the contents are erased.
- PrintWriter fileAppend =

  new PrintWriter(new FileWriter(“filename” , true));

总结

- 使用java.io中的类的注意事项：
  - 所有的IO操作都有可能抛出异常
- 在应用中，尽可能先尝试使用Reader和Writer，一旦无法成功编译程序，就会发觉自己非得使用byte-oriented程序库不可。
  - Java.util.zip程序库就是byte-oriented而非char-oriented。
- BufferedReader
- PrintWriter

3.1 Input and Output Programming

- **3.1.1 Java IO system**
- **3.1.2 Using File I/O in the Library System**
- **Exam info.**

3.1.2 Using File I/O in the Library System

1. Reading Data from a File

- The library system contains two kinds of catalog items: books and recordings.
  - Every line in catalog.dat stores exactly one catalog item.

3.1.2 Using File I/O in the Library System

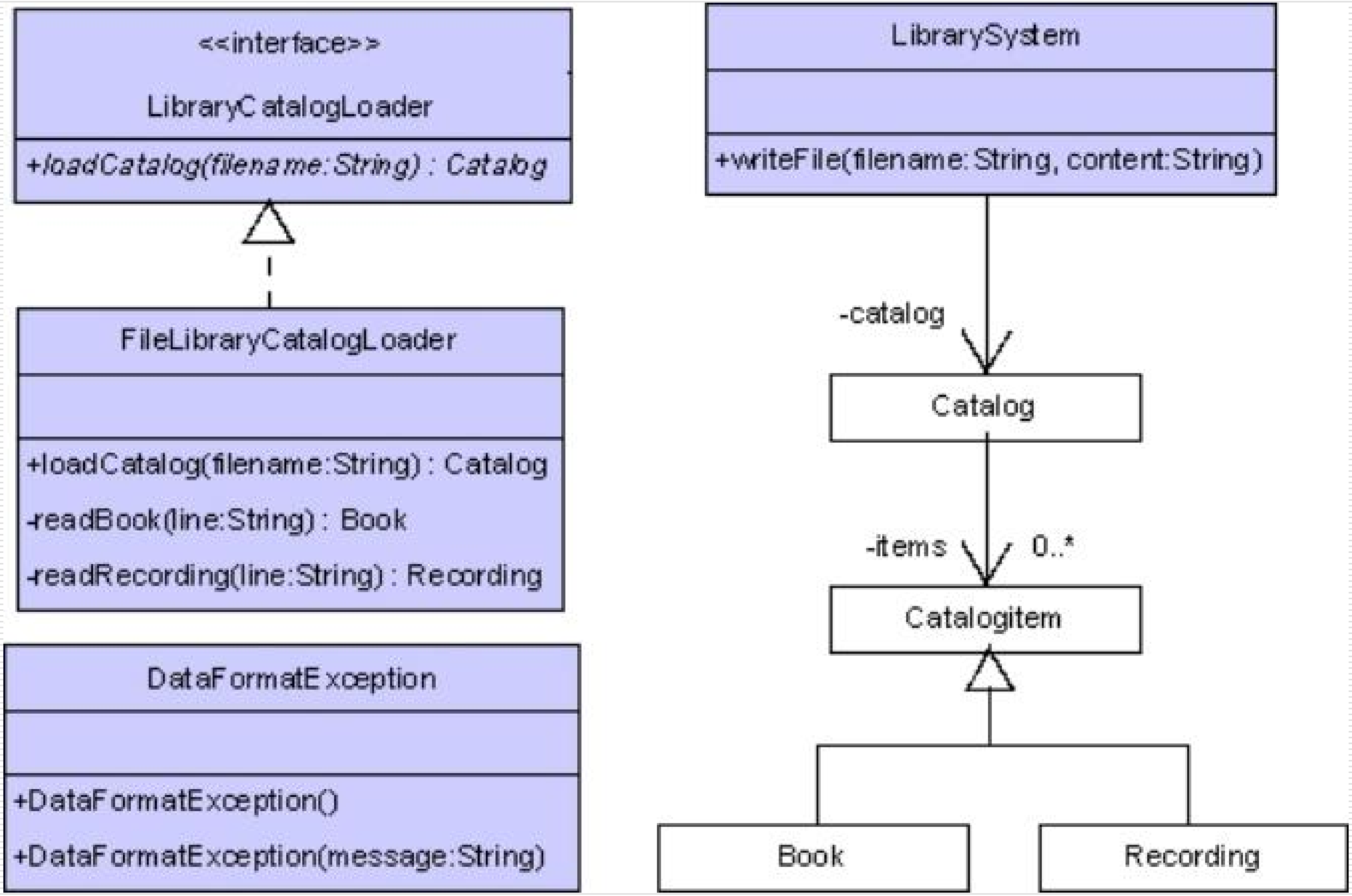
- The lines with book data have the following format:

  Book_*code_title_year_author_numberOfPages*

- The lines with recording data have the following format:

  Recording_*code_title_year_performer_format*

- LibraryCatalogLoader.java
- FileLibraryCatalogLoader.java
- DataFormatException.java



<<interface>>
LibraryCatalogLoader
+loadCatalog(filename:String) : Catalog
FileLibraryCatalogLoader
+loadCatalog(filename:String) : Catalog
-readBook(line:String) : Book
-readRecording(line:String) : Recording
DataFormatException
+DataFormatException()
+DataFormatException(message:String)
LibrarySystem
+writeFile(filename:String, content:String)
-catalog
Catalog
-items
0..*
Catalogitem
Book
Recording

3.1.2 Using File I/O in the Library System

2. Writing Data to a File

- LibrarySystem.java

Unit 4. Advanced Class Implementation

- 4.2 Graphical User Interface
  - 4.2.1 Swing Components and Containers
  - 4.2.2 Swing Event Handling
  - 4.2.3 Class JFileChooser

AWT vs. Swing Components vs.SWT

- Java应用程序界面设计主要涉及AWT(Abstract Window Toolkit)，Swing两大类库，Swing是AWT的扩展和功能的加强。
  - java.awt ---AWT components
    - 跨平台的GUI框架，使用本地窗口组件，调用操作系统的内部API，支持各个操作系统平台所提供的窗口组件的集合；
  - javax.swing-----Swing components
    - 创建了一个新的框架使用模拟窗口组件来代替本地窗口组件，丰富了窗口组件，在不同的平台上表现一致，有能力提供本地窗口系统不支持的其它特性。消耗内存，不适用PDA和移动电话等小型设备；

4.2 Graphical User Interface

- IBM创建了新的GUI库SWT，使用本地窗口组件，如果一个窗口组件在主机平台上不能获得，SWT会模拟这个窗口组件，运行速度快。
- SWT Designer是一种基于SWT技术的可视化界面设计工具。
- 对于Java界面程序开发，手工写界面代码的能力是非常重要的。
- 本节讲解Java界面程序开发(Swing)必备的核心原理。

4.2.1 Swing Components and Containers

1. Swing Component and Container
2. Swing widgets 分类及JFrame
3. Adding Components to JFrame and JPanel
4. Layout Fundamentals
5. 图形用户界面程序搭建框架
6. atomic widget
7. Composite and Component-Container
8. Intermediate containers

1 Swing Components and Container

- Component类是所有界面组件类的祖先，用来表示一个图形组件。
- A container is a special type of component that is used to organize,manage,and present other components. But a component is not necessarily a container.
  - container.png
  - Since a container can contain components,and a container is a component,then a container may contain other containers.

Common Component Properties and Behaviors

- Because all of the AWT and Swing components are descended from a common ancestor class, java.awt.Component, they all share a number of attributes and methods inherited from Component:

Common Component Properties and Behaviors

- Component:
  - public void setBackground(Color c)
  - public void setForeground(Color c)
  - public void setFont(Font font)
  - public void setEnabled(boolean b)
  - public void setVisible(boolean b)
- Jcomponent:
  - public void setBorder(Border border)
    - JComponent 拥有一个setBorder( ),允许将各种有趣的边框放置在任何看得到的组件上。

4.2.1 Swing Components and Containers

1. **Swing Component and Container**
2. **Swing widgets 分类及JFrame**
3. **Adding Components to JFrame and JPanel**
4. **Layout Fundamentals**
5. 图形用户界面程序搭建框架
6. **atomic widget**
7. **Composite and Component-Container**
8. **Intermediate containers**

2. Swing widgets 分类及JFrame

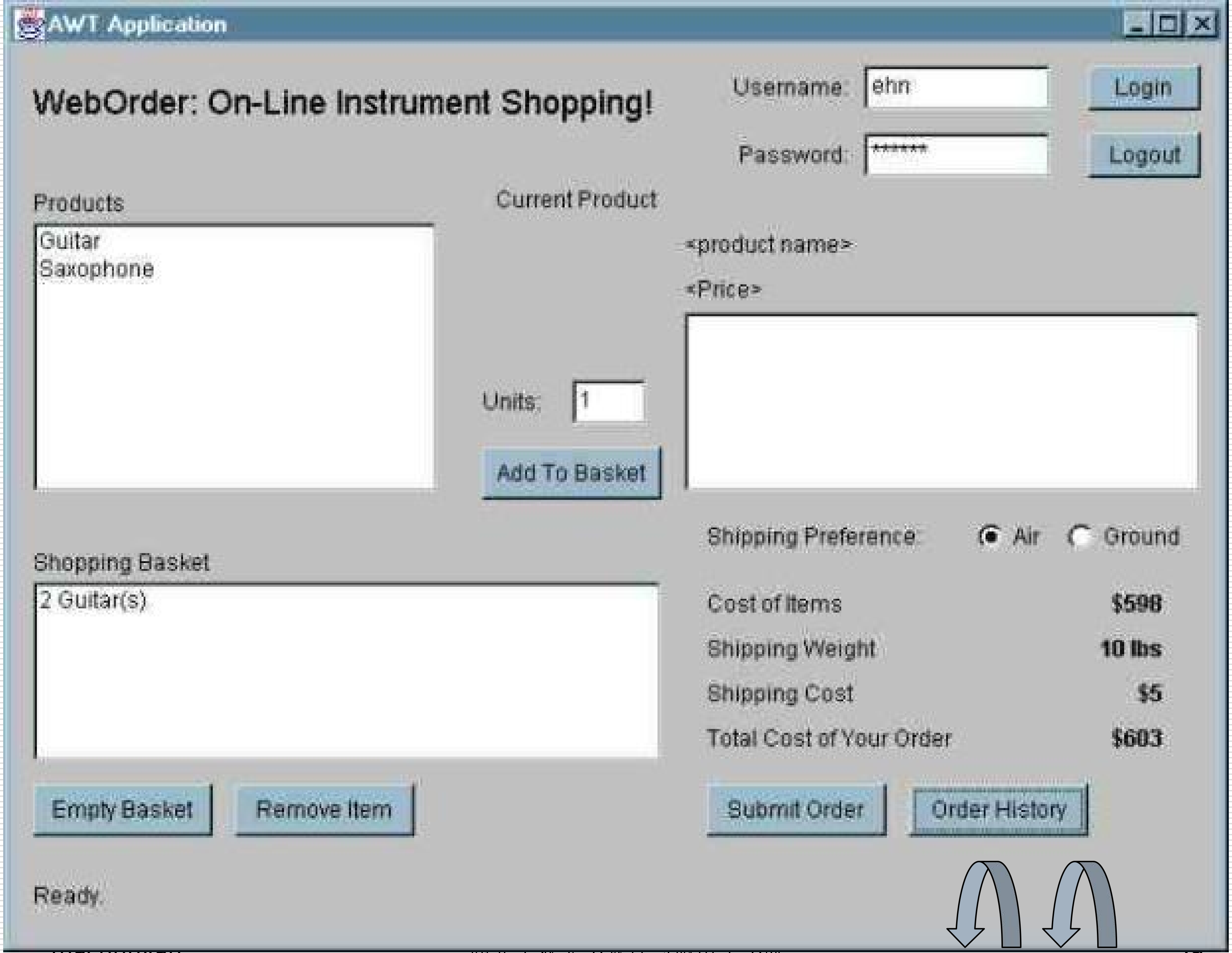
- These are three broad types of Swing widgets(javax.swing.*):
  - atomic widgets(or components)
  - intermediate containers
  - top-level containers

Atomic widgets(or components):

- An atomic widget is one that corresponds to a basic GUI feature such as a button or label.
  - These atomic widgets self-sufficient and cannot contain other widgets.
- A set of frequently used atomic widgets include :

1) Atomic widgets for user input with the mouse:

✓ JButton;JCheckBox;JComboBox; JRadioButton;JList;JSlider;JTree


AWT Application

WebOrder: On-Line Instrument Shopping!

Username: ehn

Password: *******

Login

Logout

Products

Guitar
Saxophone

Current Product

<product name>

<Price>

Units: 1

Add To Basket

Shipping Preference: [✓ Air] Ground

Shopping Basket

2 Guitar(s)

Cost of Items	$598
Shipping Weight	10 lbs
Shipping Cost	$5
Total Cost of Your Order	$603

Empty Basket

Remove Item

Submit Order

Order History

Ready.

Atomic widgets(or components):

2) 不可编辑信息的显示：向用户显示不可编辑信息的组件:

- ✓ JLable;JProgressBar;JProgressMonitor;
- ✓ 为组件设置对用户有帮助的提示信息：

  Tool tips---the JComponent method setToolTipText("String") sets up a small text label that appears briefly,to inform the user, when the mouse cursor lingers over a widget's window.

Atomic widgets(or components):

3）可编辑信息的显示：向用户显示能被编辑的格式化信息的组件：

如JColorChooser, JFileChoose, JFileChooser, JTable, JTextArea，JTextField。

Intermediate containers

- **An intermediate container can contain and manage other widgets(atomic widgets or intermediate container)to form a composite widget.**
- 用于容纳界面元素，以便在布局管理器的设置下可容纳更多的组件，实现容器的嵌套。
  - **JPanel;JScrollPane; JSplitPane; JTabbedPane; JToolBar; JLayeredPane; JDesktopPane; JInternalFrame; JRootPane**

Top-level containers

- A top-level container holds intermediate containers and atomic widgets.
  - 顶级容器不能被其它的容器包含。
- A GUI application usually has a top-level widget that contains and manages other widgets in the program.

Top-level containers include:

- Swing has four root-window containers:
  - JFrame for general use
    - 在构建桌面Java应用程序时，通常创建一个JFrame来容纳所需的组件，以提供期望的GUI外观和功能。
  - JDialog for creating pop-up dialog boxes
  - JApplet for writing Swing based applets.
  - JWindow

- In a typical GUI application, the main method performs these steps:
  1. Create a top-level container, usually a JFrame.
  2. Compose the GUI by adding widgets(intermediate containers and atomic widgets) to containers in a desired layout.
  3. Set up and register event handlers(listeners) to respond to user interaction.
  4. Display the GUI and return.

JFrame

- **JFrame是java的主框架，几乎所有的Java应用程序界面都是在主框架之中设计的。**
  - **We’ll create a program/class called FrameTest,and within the main( ) method of that class we’ll perform the bare minimum steps necessary to create and display a JFrame:**
- **FrameTest.java**

3.2.1 Swing Components and Containers

1. Swing Component and Container
2. Swing widgets 分类及JFrame
3. Adding Components to JFrame and JPanel
4. Layout Fundamentals
5. 图形用户界面程序搭建框架
6. atomic widget
7. Composite and Component-Container
8. Intermediate containers

3. Adding Components to JFrame and JPanel

- 使用窗体(JFrame)的内容嵌版提供的 add( )方法将原子组件逐一添加到窗体上进行显示。

```
JFrame frame = new JFrame("Frame Name");
Container contentPane = frame.getContentPane();
contentPane.add(new JLabel("Label Name"));
```

- 使用窗体提供的 add( )方法将原子组件逐一添加到窗体上进行显示。

```
JFrame frame = new JFrame("Frame Name");
frame.add(new JLabel("Label Name"));
```

  - FrameTest1.java

添加组件到面板(JPanel)

- JPanel是一个没有明显边界的中级容器，它不能象JFrame那样单独使用(显示)，它必须放置在另一个容器中，被用作将多个其它图形化Component方便地组织在一起的手段；
- 通过JPanel提供的add()方法放置各种原子组件或中级容器。
  - JLabel stuff = new JLabel("I am a label");
  - JPanel aPanel = new JPanel( );
  - aPanel.add(stuff);
- 一个JPanel也可以放置/布局其它的JPanel;

添加组件到面板(JPanel)

- 使用窗体(JFrame)的内容嵌版或窗体提供的add( )方法将面板(JPanel)添加到窗体上(JFrame)显示面板上摆放的原子组件。
- FrameTest2.java
- An import aspect of GUI programming is achieving a desired appearance for the user interface.
  - Layout： the positioning and sizing of child windows inside a containing window.

3.2.1 Swing Components and Containers

1. **Swing Component and Container**
2. **Swing widgets 分类及JFrame**
3. **Adding Components to JFrame and JPanel**
4. **Layout Fundamentals**
5. **图形用户界面程序搭建框架**
6. **atomic widget**
7. **Composite and Component-Container**
8. **Intermediate containers**

4. Layout Fundamentals

- Layout management is the process of determining the layout. Java allows you to take either of these two approaches:
  ① Automatic layout----Use a layout manager to determine the layout automatically.
  ② Manual layout----Use no layout manager, specify the position and size of each widget window explicitly.

①Automatic layout

- 为了使生成的图形用户界面具有良好的平台无关性，Java语言提供了布局管理器——LayoutManager 来管理组件在容器中的布局。
  - 每个容器都可以指定一个布局管理器，当容器需要对某个组件进行定位或判断其大小尺寸时，就会调用其对应的布局管理器。
- 布局管理器LayoutManager是一个接口。
  - 具体布局组件的算法分别封装在一个实现布局管理器接口的类中（见下页），容器可以在运行时刻使用不同的布局管理器。

① Automatic layout(cont.)

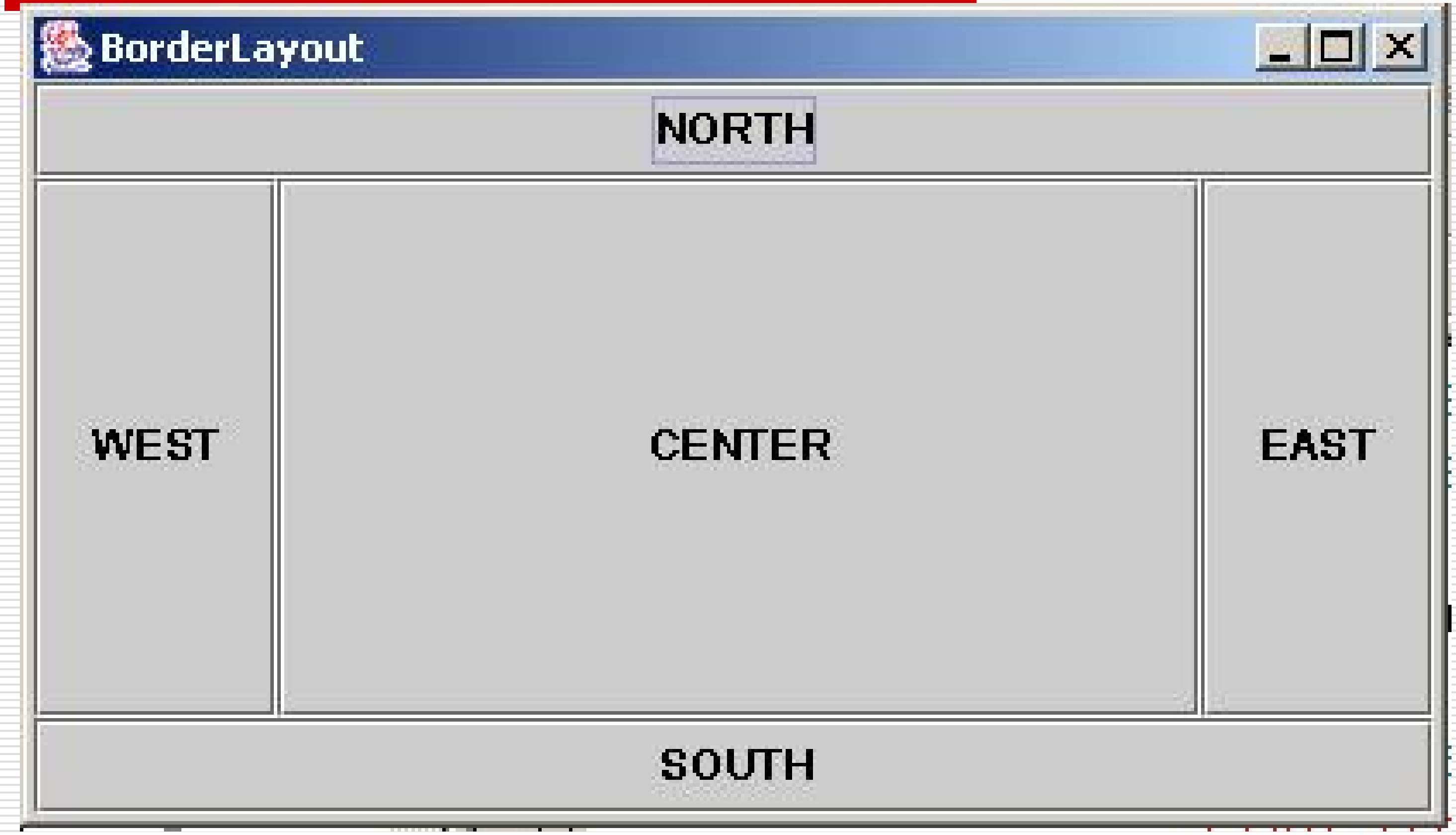
- There are five common **Layout Managers** defined in the core Java language.
  - BorderLayout
  - FlowLayout
  - GridLayout

①Automatic layout(cont.)

- 如何给一个容器指定一个具体的布局管理器?
  - We either use the default layout manager that is associated with a container such as:
    - JFrame 缺省布局管理器: BorderLayout
    - JPanel 缺省布局管理器: FlowLayout
  - Or explicitly assign a specific LayoutManager to a container using the contain’s setLayout( ) method.
    - void setLayout(LayoutManager mgr)

1) BorderLayout

- **BorderLayout is the default layout manager for JFrame.**
- **The BorderLayout subdivides a Container into five regions.**
  - 每个区域只能放置一个组件。
  - **as shown below for a JFrame:**


BorderLayout
NORTH
WEST
CENTER
EAST
SOUTH

1) BorderLayout

- 如果不指定对象的摆放区域，缺省为CENTER
- 可以通过容器添加组件时的add方法的第二个参数指出组件摆放区域，add方法的第二个参数的取值：
  - BorderLayout.NORTH :置于顶端
  - BorderLayout.SOUTH :置于底部
  - BorderLayout.WEST :置于右侧
  - BorderLayout.EAST:置于左侧
  - BorderLayout.CENTER:填满中央区域，与四周组件相接，或延展至边缘。
- FrameTest2.java
- BorderLayoutDemo.java

1) BorderLayout

- **5个区域中的任何一个都可以为空，其它非空区域可以延伸到空区域留出的空间。**
- **BorderLayout控制的Container只能包含5个Component?**
  - 可以利用**JPanel** 使得**BorderLayout** 中某个区域显示多个组件，达到设计复杂用户界面的目的 。

2) FlowLayout

- **FlowLayout 是JPanel，JApplet的缺省布局管理器。**
  - 其组件的放置规律是从上到下、按照对齐方式从左到右进行放置，如果当前行已放置不下该组件，则放置到下一行。
- 可以利用JPanel 使得FlowLayout中某个区域显示多个组件，达到设计复杂用户界面的目的 。
- FlowLayoutDemo.java

2) FlowLayout(cont.)

- 构造方法主要下面几种：
  - **FlowLayout();**
    - 缺省的对齐方式居中对齐，横向间隔和纵向间隔都是缺省值5个象素。
  - **FlowLayout(FlowLayout.LEFT);**
    - 居左对齐，横向间隔和纵向间隔都是缺省值5个象素。
  - **FlowLayout(FlowLayout.RIGHT,20,40);**
    - 第一个参数表示组件的对齐方式，指组件在这一行中的位置是居中对齐、居右对齐还是居左对齐，第二个参数是组件之间的横向间隔，第三个参数是组件之间的纵向间隔，单位是象素。

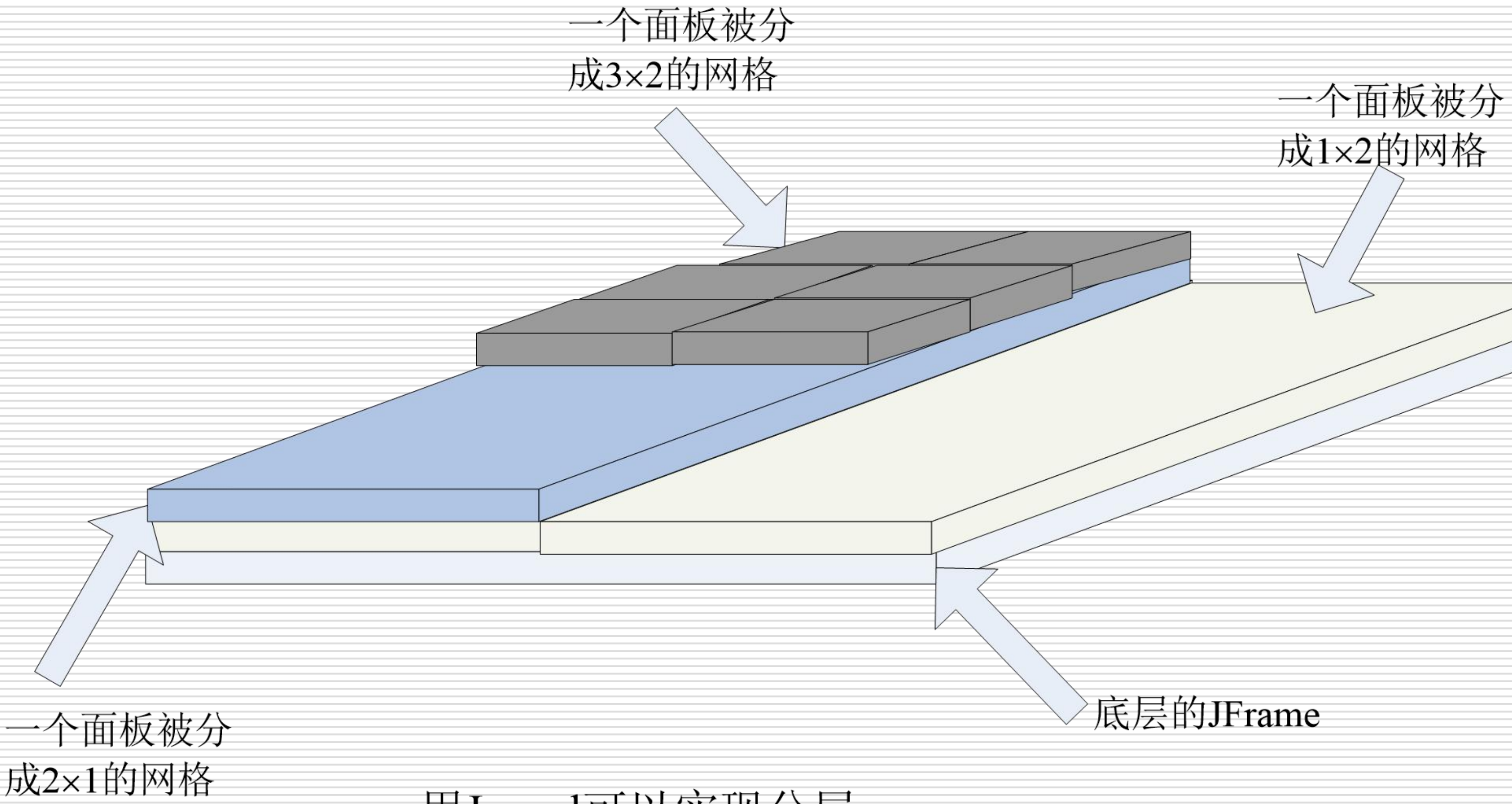
3) GridLayout

- GridLayout可以建立一个组件表格，而且当组件被放置到容器中时，会依序由左至右、由上至下摆放在每个格子(grid)里头。
- 在构造函数中指定要求的列数和行数，各行各列所分配的空间的大小是相同的。
  - **GridLayout grid=new GridLayout(7,3);**
- 可以利用**JPanel** 使得**GridLayout**中某个区域显示多个组件，达到设计复杂用户界面的目的。
- GridLayoutDemo.java

②Manual layout

- 如果采用无布局管理器 setLayout(null)，则必须使用setLocation(),setSize(),setBounds()等方法手工设置组件的大小和位置，此方法会导致平台相关，不鼓励使用

- 在复杂的图形用户界面设计中，为了使布局更加易于管理，具有简洁的整体风格，一个包含了多个组件的容器本身也可以作为一个组件加到另一个容器中去，容器中再添加容器，这样就形成了容器的嵌套。



用Jpanel可以实现分层

3.2.1 Swing Components and Containers

1. Swing Component and Container
2. Swing widgets 分类及JFrame
3. Adding Components to JFrame and JPanel
4. Layout Fundamentals
5. 图形用户界面程序搭建框架
6. atomic widget
7. Composite and Component-Container
8. Intermediate containers

5. 图形用户界面程序搭建框架：

- **继承JFrame创建用户界面程序**
  - **public class JFrameName extends JFrame**
  - 示例：**JFrame.java**
- **继承JPanel创建用户界面程序**
  - **public class JPanelName extends Jpanel**
    - 建立一个JPanel中级容器，把要显示的界面元素添加到该中级容器中，然后把该中级容器放置在JFrame内；
  - 示例：**JPane.java**

4.2.1 Swing Components and Containers

1. Swing Component and Container
2. Swing widgets 分类及JFrame
3. Adding Components to JFrame and JPanel
4. Layout Fundamentals
5. 图形用户界面程序搭建框架 ★★
6. 原子组件(atomic widget)举例
7. Intermediate containers 举例

6. 原子组件举例(atomic widgets)：

① Component JLabel

② Component JButton

③ Component JRadioButton

④ Component JTextField

⑤ Component JTextArea

⑥ Component JList

①Component JLabel

- Components of class JLabel can display text, an image, or both.
- JLabel 的构造函数
  - JLabel()
  - JLabel(Icon image)
  - JLabel(Icon image,int horizontalAlignment)
  - JLabel(String text)
  - JLabel(String text,Icon icon,int horizontalAlignment)
  - JLabel(String text, int horizontalAlignment)

①Component JLabel(cont.)

- 参数 horizontalAlignment的取值：
  - JLabel. LEFT
  - JLabel. CENTER
  - JLabel. RIGHT
  - JLabel. LEADING
  - JLabel. TRAILING
- public class ImageIcon implements Icon …
  - ImageIcon icon = new ImageIcon("bananas.jpg", "an image with bananas");
- Class JLabelDemo creates a window with three JLabel components:JLabelDemo.java

② Component JButton

- Components of class JButton can display text, an image, or both.
- Class JButtonDemo creates a window with two JButton components:
  - JButtonDemo.java

③ Component JRadioButton

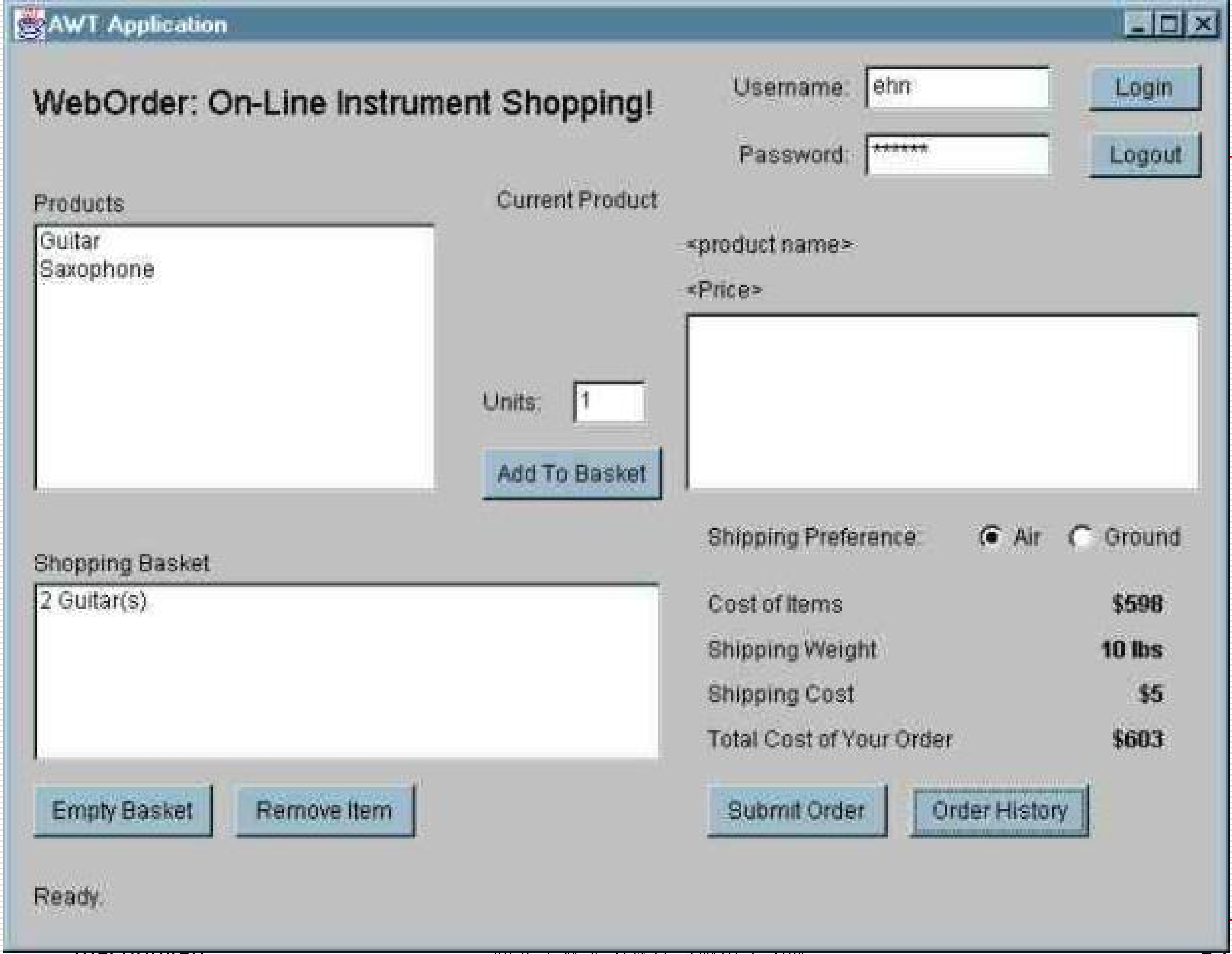
- Components of class JRadioButton can be selected or deselected by the user.
- JRadioButtonDemo.java
- If you want radio buttons to behave in an “exclusive or” fashion, you must add them to a “button group.”

③ Component JRadioButton (cont.)

- public class ButtonGroup
  - Creating a set of buttons with the same ButtonGroup object means that turning "on" one of those buttons turns off all other buttons in the group.
  - public void add(AbstractButton b)
    - any AbstractButton can be added to a ButtonGroup.
- ButtonGroups.java

④ Component JTextField

- Components of class JTextField let the user enter (or edit) a single line of text.
  - public JTextField(String text, int columns)
  - public void setHorizontalAlignment(int alignment)
  - public void setEditable(boolean b)
  - public void setText(String t)
- JTextFieldDemo.java


AWT Application

WebOrder: On-Line Instrument Shopping!

Username: ehn

Password: *******

Login

Logout

Products

Guitar
Saxophone

Current Product

<product name>

<Price>

Units: 1

Add To Basket

Shipping Preference: [● Air] Ground

Shopping Basket

2 Guitar(s)

Cost of Items	$598
Shipping Weight	10 lbs
Shipping Cost	$5
Total Cost of Your Order	$603

Empty Basket

Remove Item

Submit Order

Order History

Ready.

⑤Component JTextArea

- Components of class JTextArea let the user enter (or edit) multiple lines of text and also used to display blocks of text.
- Class JTextArea
  - public JTextArea(String text, int rows, int columns)
  - public void setEditable(boolean b)
  - public void setText(String t)
  - public void append(String str)

⑤Component JTextArea(cont.)

- A JTextArea component does not have scroll bars. If scroll bars are needed, the JTextArea is wrapped in a JScrollPane, which provides the scroll bars.
  - 通过JScrollPane，可以控制滚动条的使用与否——允许垂直滚动条、水平滚动条、两者兼得、或两者都不许

⑤Component JTextArea(cont.)

- public JScrollPane(Component view, int vsbPolicy, int hsbPolicy)
  - vsbPolicy的取值：
    - JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED
    - JScrollPane.VERTICAL_SCROLLBAR_ALWAYS
    - JScrollPane.VERTICAL_SCROLLBAR_NEVER
  - hsbPolicy的取值：
    - JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED
    - JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS
    - JScrollPane.HORIZONTAL_SCROLLBAR_NEVER

⑤Component JTextArea (cont.)

- 如何为JTextArea添加滚动条？
- t = new JTextArea(“t1”, 1, 20);

```
JScrollPane sp = new JScrollPane(t,
    JScrollPane.VERTICAL_SCROLLBAR_NEVER,
    JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
```

- JTextAreaDemo.java

⑥ Component JList

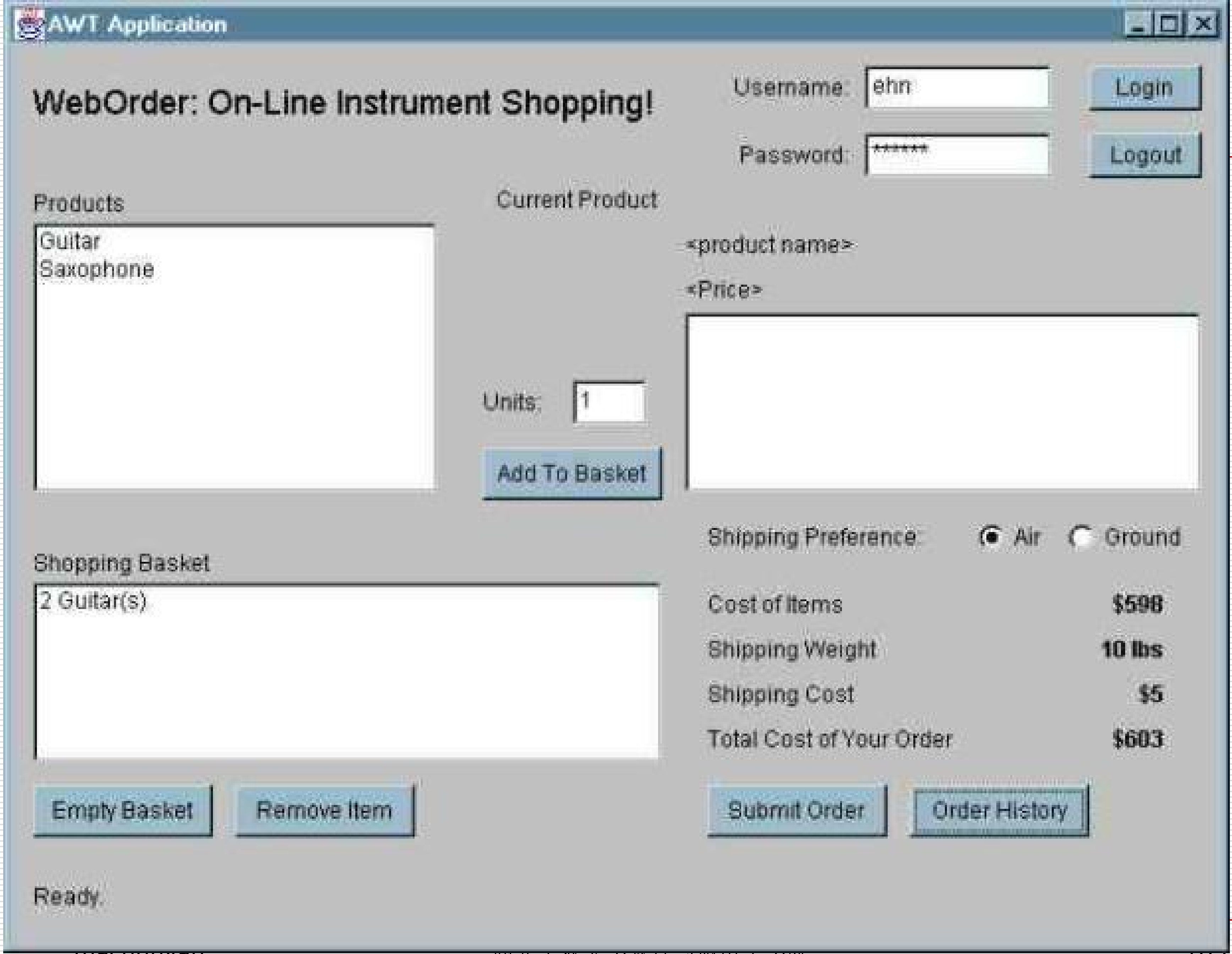
- **JList显示一个列表框，它会在屏幕上持续占用固定行数的空间显示列表内的项目。**
- **Components of class JList let the user select one or more elements from a list.**
- 对比：组合框JComboBox
  - 与一组单选按钮的功能类似，组合框(下拉列表)强制用户从一组可能的元素中只选择一个。

⑥ Component JList (cont.)

- JList构造函数：
  - JList()
  - JList(ListModel dataModel)
  - JList(Vector<?> listData)
  - JList(Object[ ] listData)
    - 如果将某个String数组的内容加入JList，将该String数组传入JList的构造函数，便会自动建构出整份列表。

⑥ Component JList (cont.)

- A JList component does not have scroll bars.
  - If scroll bars are needed, the JList is wrapped in a JScrollPane, which provides the scroll bars.
- JListDemo.java



WebOrder: On-Line Instrument Shopping!

Username: ehn

Password: *******

Products

Guitar
Saxophone

Current Product

<product name>

<Price>

Units: 1

Shipping Preference: [✓ Air] Ground

Shopping Basket

2 Guitar(s)

Cost of Items	$598
Shipping Weight	10 lbs
Shipping Cost	$5
Total Cost of Your Order	$603

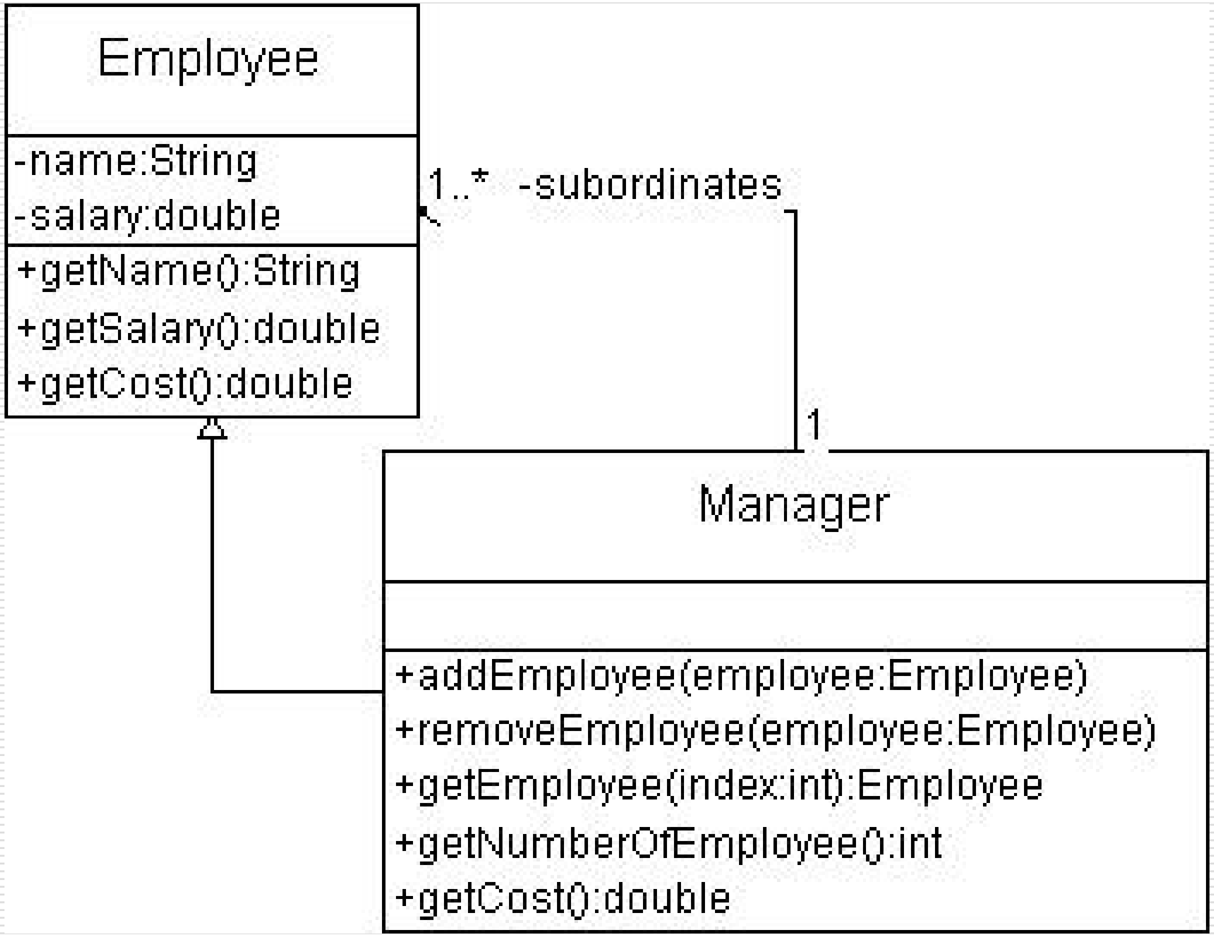
Ready.

3.2.1 Swing Components and Containers

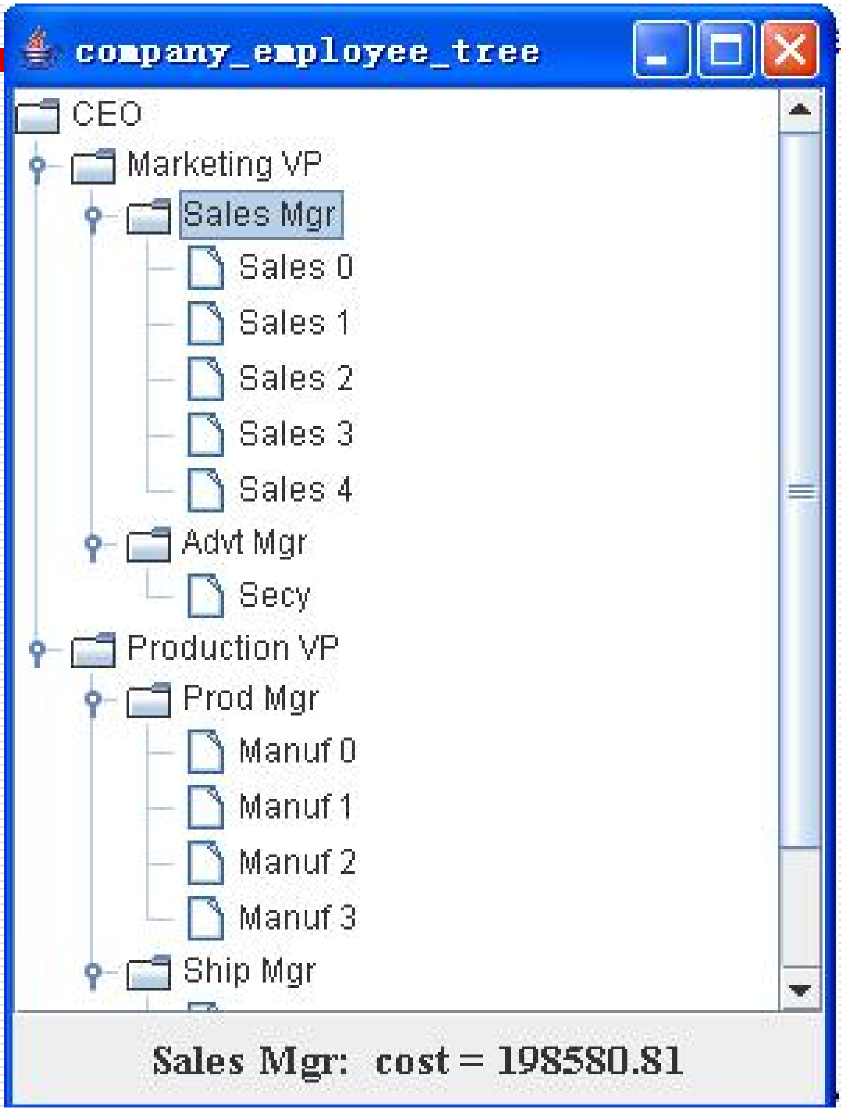
1. Swing Component and Container
2. Swing widgets 分类及JFrame
3. Adding Components to JFrame and JPanel
4. Layout Fundamentals
5. 图形用户界面程序搭建框架
6. atomic widget
7. Composite and Component-Container
8. Intermediate containers

7. Composite设计模式和Component-Container

- 在Java/AWT中Component-Container是最为明显的一个Composite设计模式的应用。
- Composite设计模式：将对象组合成树形结构以表示“部分–整体”的层次结构。
- Container.png
- 可以很容易的对界面中的所有元素进行组织。
  - 上面类图的对象图就是一个树。这样组织的结构使得对于组件的处理变得方便。通过层层的递归，可以实现足够复杂的体系。
  - Container-tree.png
- employee-salary.doc


Employee
-name:String
-salary:double
+getName():String
+getSalary():double
+getCost():double
1..*
-subordinates
1
Manager
+addEmployee(employee:Employee)
+removeEmployee(employee:Employee)
+getEmployee(index:int):Employee
+getNumberOfEmployee():int
+getCost():double

- **JTree:树状组件**
- 例如:
  - **employee.java**
  - **CompanyGUI.java**



Sales Mgr: cost = 198580.81

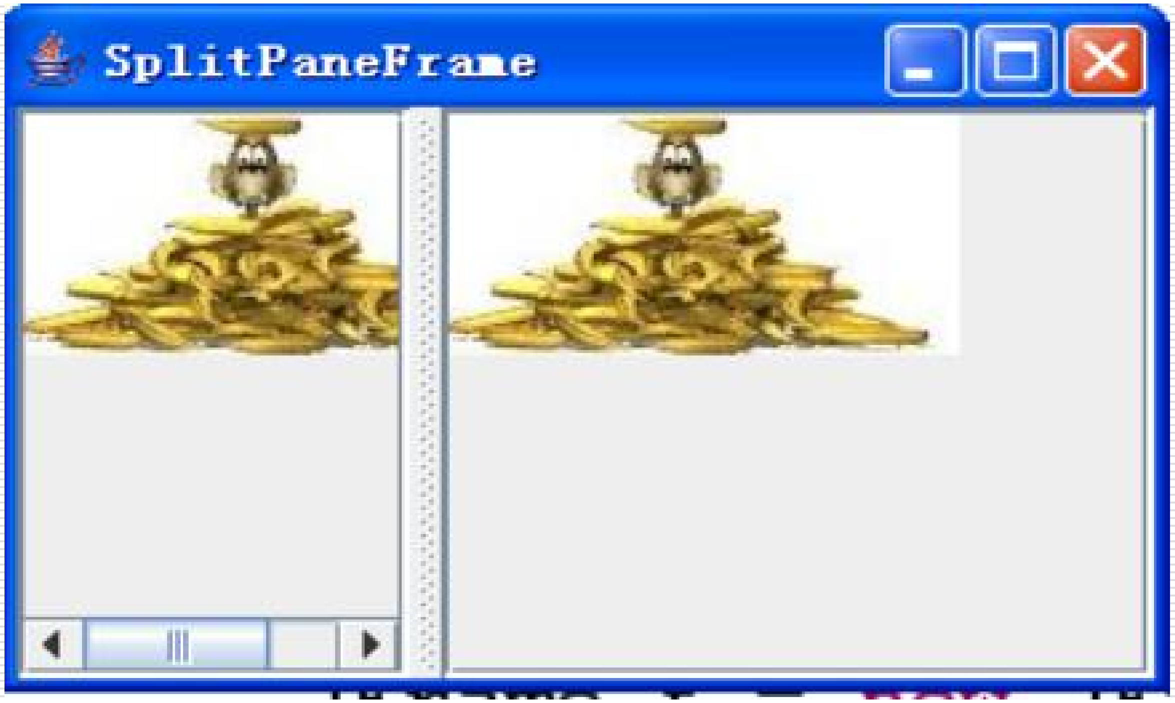
3.2.1 Swing Components and Containers

1. Swing Component and Container
2. Swing widgets 分类及JFrame
3. Adding Components to JFrame and JPanel
4. Layout Fundamentals
5. 图形用户界面程序搭建框架
6. atomic widget
7. Composite and Component-Container
8. Intermediate containers

8. Intermediate containers 举例

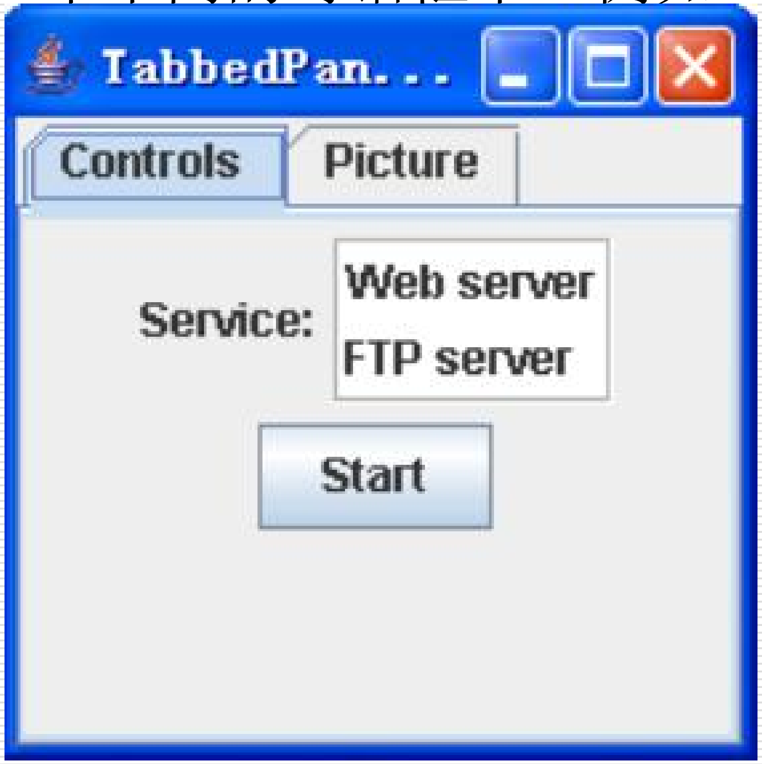
① JSplitPane提供可拆分窗口，支持水平拆分和垂直拆分并带有滑动条。

✓ 例如：SplitPaneFrame.java



8. Intermediate containers 举例(续)

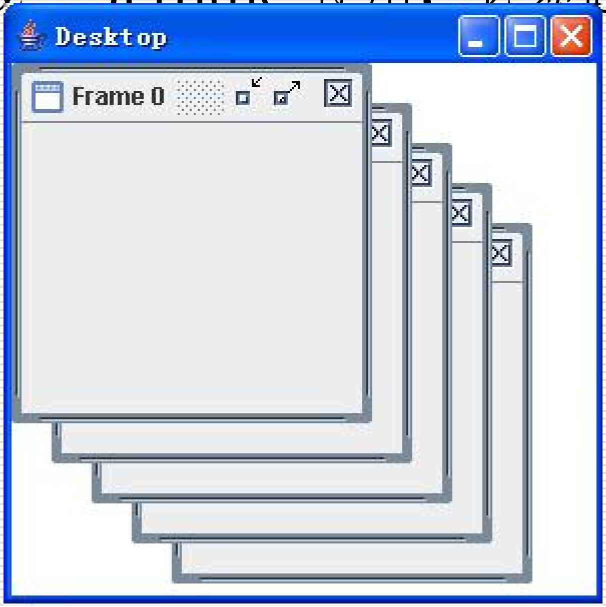
② JTabbedPane创建“页签式的对话框”，这种对话框中沿着窗体的一边有类似文件夹的页签，当在页签上点击时，就会进入另一个不同的对话框中。例如：TabbedPaneFrame.java



8. Intermediate containers 举例(续)

③ JDesktopPane 和JInternalFrame

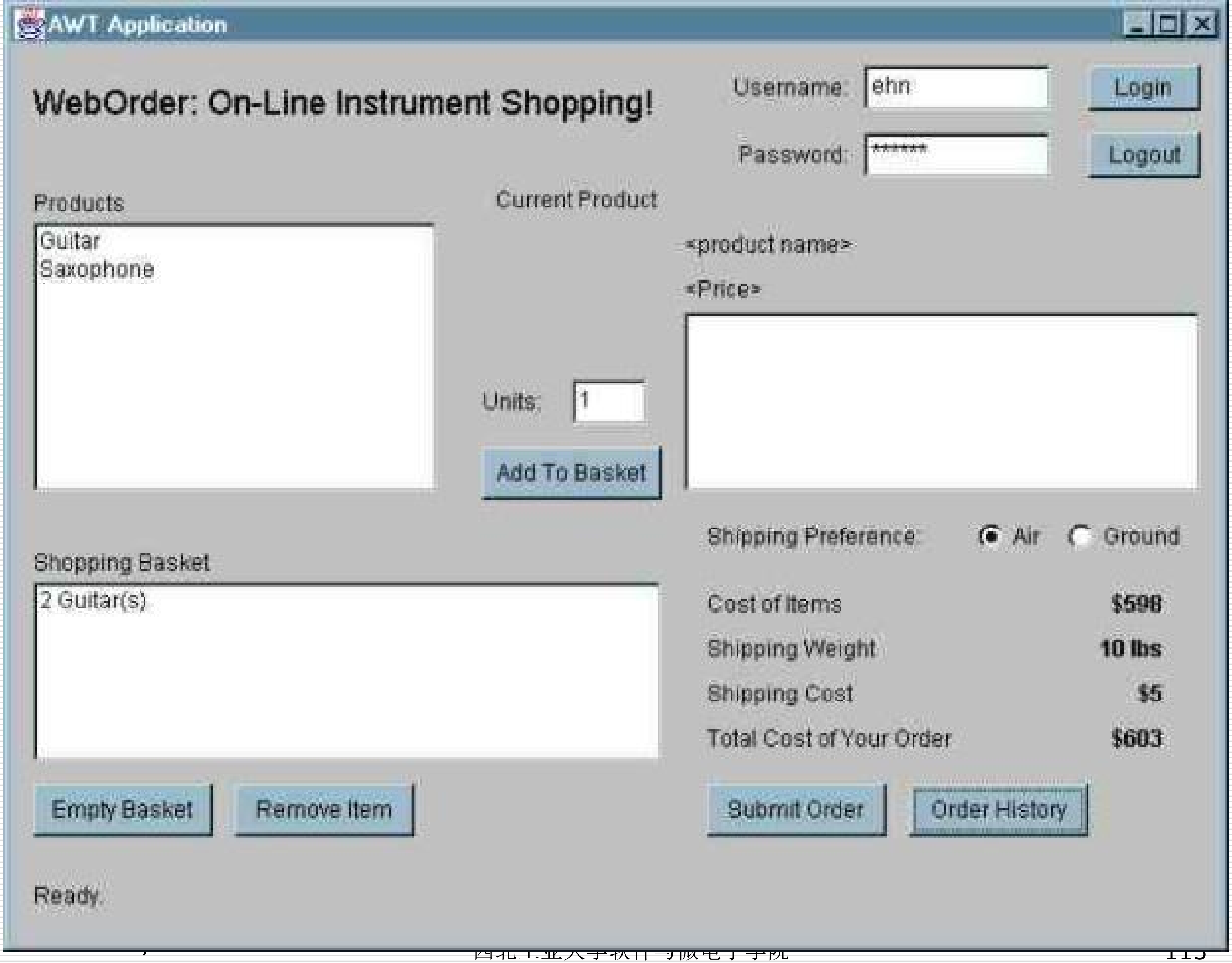
- **JDesktopPane:**一个容器，必须置于一JFrame内，用于容纳子窗口JInternalFrame。
- **JInternalFrame:**可当成一 JFrame 使用，必须通过add方法添加到 JDesktopPane内， e就如同一个窗口在另一个窗口内部
- 例如：**Desktop.java**



4.2.2 Swing Event Handling

1. 今日问题簿:
   - 事件处理机制的原理是否掌握?

- **We have gotten a sense of how to build a GUI----that is a 'view'(how a GUI looks).**
- **We will learn how to recognize events(how a GUI behaves)---i.e. users' interaction with the GUI/view------and how to control how the application responds to them.**


AWT Application

WebOrder: On-Line Instrument Shopping!

Username: ehn

Password: *******

Login

Logout

Products

Guitar
Saxophone

Current Product

<product name>

<Price>

Units: 1

Add To Basket

Shipping Preference: [✓ Air] Ground

Shopping Basket

2 Guitar(s)

Cost of Items	$598
Shipping Weight	10 lbs
Shipping Cost	$5
Total Cost of Your Order	$603

Empty Basket

Remove Item

Submit Order

Order History

Ready.

1. Events—Basic Concepts

- GUI events are generated when the user interacts with a component on the GUI.
  - For example,when a JButton is clicked, ActionEvent(用户对界面的操作在java语言上的描述)，the type of event that is generated.

1. Events—Basic Concepts (续)

- Each type of event is represented by a distinct class.
  - As with virtually everything else in Java,events are objects!
- java.awt.event---classes related to AWT event handling
- javax.swing.event----classes related to Swing event handling.

1. Events—Basic Concepts (续)

- 事件列表：Event&Listener.xls
- When we create a GUI component, it automatically has the ability to generate events whenever a user interacts with it---we need do nothing to get this phenomenon to occur.

2. Event handling

- In order to handle events,we need to do two things when Programming a GUI:

  ① 创建Listener(监听器)类

  - 对于某种类型的事件XXXEvent, 要想接收并处理这类事件, 必须定义实现监听器接口XXXListener监听器类;

  ② 调用组件的addXXXListener(…)方法注册Listener对象，就可以监听该组件触发的XXXEvent。

2. Event handling

① 创建Listener类

- You must create a listener class that must implement the appropriate interface.
  - For example we may create an object of a class that implements interface ActionListener, a type of listener that is capable of listening to ActionEvents.
  - ButtonEventsDemo.java
- Event&Listener.xls

2. Event handling

② 调用组件的addXXXListener()方法注册Listener对象。

- We must register the Listener object with the *specific* Component object(s) that we want the listener to listen to.
  - This registration is performed by calling an addXXXListener( ) method in the event-firing component,
    - in which “XXX” represents the type of event listened for.

2. Event handling

- **So as a programmer, all you do is create a listener object and register it with the component that's firing the event.**
- **例如：ButtonEventsDemo1.java**
- 内部类（inner class）是被定义于另一个类中的类，使用内部类的主要原因是由于：
  - 一个内部类的对象可访问外部类的成员方法和变量，包括私有的成员。
  - 实现事件监听器时，采用内部类、匿名类编程非常容易实现其功能。
  - 内部类往往应用在AWT的事件处理机制中。
  - 详细语法参阅《java编程思想》第八章

ActionEvent and ActionListener

- **Components:** JRadioButton, JButton,JTextField,…
- **Type of Events :** ActionEvent
- **Type of Listener: interface** ActionListener
- **Defined in Package:** Java.awt.event

3. Basic Event Types

```
public interface  ActionListener {
    public void actionPerformed(ActionEvent e) ;
}
```

- 所有的行为类事件监听器都要实现ActionListener接口，在ActionListener中只定义了一个actionPerformed()方法，对于每个注册了这个监听器的组件，当其中发生了行为类事件（比如按钮按下）时，就会调用actionPerformed()方法来响应该事件。
- For example:
  - ButtonEventsDemo.java

WindowEvent and WindowListener

- **Components: JFrame, JDialog**
- **Type of Events : WindowEvent**
- **Type of Listener: interface WindowListener**
- **Defined in Package: Java.awt.event**
- **For example: Calculator.java**

3. Basic Event Types

```
public interface  WindowListener {
  public void windowOpened(WindowEvent e)
  public void windowClosing(WindowEvent e)
  public void windowClosed(WindowEvent e)
  public void windowActivated(WindowEvent e)
  public void windowDeactivated(WindowEvent e)
  public void windowIconified(WindowEvent e)
  public void windowDeiconified(WindowEvent e)
}
```

3. Basic Event Types

- 大部分是通过继承事件监听器接口来处理事件的。但是继承Java接口我们必须实现出接口中的所有方法。
  - 有些接口包含了大量的函数(如WindowListener)，如果要一个个实现是件很麻烦的事，Java中定义了相应接口的Adapter适配器类来解决这种情况。
- 适配器类已经实现了接口的所有方法，而我们只要继承适配器就可在代码内做我们想做的事情。即只要覆写我们想要实现的方法。
  - 例如： WindowListener接口的适配器类为WindowAdapter 例如：
    - abastract class WindowAdapter

3. Basic Event Types

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

- For example: Calculator.java

3. Basic Event Types

- java.awt.event包中定义的事件适配器类包括以下几个：
  - ComponentAdapter(组件适配器)
  - ContainerAdapter(容器适配器)
  - FocusAdapter(焦点适配器)
  - KeyAdapter(键盘适配器)
  - MouseAdapter(鼠标适配器)
  - MouseMotionAdapter(鼠标运动适配器)
  - WindowAdapter(窗口适配器)

3. Basic Event Types
ListSelectionEvent and ListSelectionListener

- **Components:** **JList**
- **Type of Events :** **ListSelectionEvent**
- **Type of Listener:interface** **ListSelectionListener**
- **Defined in Package:** **Javax.swing.event**

  - public interface **ListSelectionListener {**

```
    public void valueChanged(ListSelectionEvent e) ;
}
```

3. Basic Event Types

- public class JList extends JComponent
- public void setSelectionMode(int selectionMode)

  selectionMode的取值：
  - ListSelectionModel.SINGLE_SELECTION
  - ListSelectionModel.SINGLE_INTERVAL_SELECTION
  - ListSelectionModel.MULTIPLE_INTERVAL_SELECTION
- public void setListData(Object[] listData)
  - 设置列表项
- example：FruitListDemo.java

3. Basic Event Types

- 如果你想选取列表内的项目，只要调用getSelectedValues (多选)就可以得到一个String数组，其中的值便是被选取的项目。或者调用getSelectedValue(单选)得到第一个所选择的选项。
  - public Object[ ] getSelectedValues()
  - public Object getSelectedValue()
- **public boolean getValueIsAdjusting()**

  ----it returns true when the mouse is pressed or dragged,

  false when the mouse is released.

3.2.3 Class JFileChooser

1. Dialog Boxes

- 对话框是一种会从另外一个窗口中弹出的窗口。其目的在于处理某个特别的问题，但不希望这些细节搞乱原本的窗口。对话框被大量运用于窗口程序设计环境中。
- 如果想生成对话框，必须继承JDialog，它是另一种类型的Window，就像Jframe， JDialog也有布局管理器（其缺省为BorderLayout)。
- *Modal* dialog box：
  - When an application opens a modal dialog box, the rest of the application stops responding to the user so the user is forced to respond to the dialog box.

- 产生JDialog之后,
  - 必须调用其setVisible(true)函数加以显示，启动对话框。
  - 当对话框被关闭时，需调用dispose()，释放该对话框动用的资源。
- 例如：DialogsDemo.java

Common Dialog

- Javax.swing.JOptionPane provide these useful static methods to create custom-designed dialogs:
  - showConfirmDialog(…)
  - showMessageDialog(…)
  - showOptionDialog(…)
  - showInputDialog(…)
- 例如： DialogsDemo.java

2. File dialogs

- 几乎所有的图形化操作系统都支持文件打开和存储的对话框操作，所以Swing API 提供了java class JFileChooser(位于javax.swing包)将此封装起来，提供更简便的使用。
- The dialog box created by JFileChooser is *modal*

- **public class JFileChooser**
  - **public void setFileSelectionMode(int mode)**
  - **public int showOpenDialog(Component parent)**
  - **public int showSaveDialog(Component parent)**
  - **public File getSelectedFile()**
  - **public File getCurrentDirectory()**

- 调用showOpenDialog()产生 “open file”对话框；调用showSaveDialog()产生 “save file”对话框；这两个函数会持续到对话框被使用者关闭后才返回。
  - **the open dialog box allows the user to locate a file while the save dialog box lets the user specify the directory where the file will be saved and the name of the saved file.**

- 用户选择结果的查询：
  - 通过方法**showOpenDialog**和**showSaveDialog**的返回值(下面三种返回值)确定用户是否取消对话框的操作。
    - **JFileChooser.CANCEL_OPTION**
    - **JFileChooser.APPROVE_OPTION**
    - **JFileCHooser.ERROR_OPTION**
      - **if an error occurs or the dialog is dismissed**
  - 可以通过方法**getSelectedFile()**和**getCurrentDirectory()**查询用户选择的结果，如果它们传回**null**，就表示用户取消了对话框。

- JFileChooser fileChooser = new JFileChooser();
- 下例试用了两种类型的JFileChooser对话框，
  - 其中一个用来打开文件，另一个用来存储，所有重要动作都发生在两个按钮被按下时触发的动作监听器(ActionListener)中：
  - TextEditor.java