软件体系结构

Zhenyan Ji

Beijing Jiaotong University

结构型模式—适配器模式

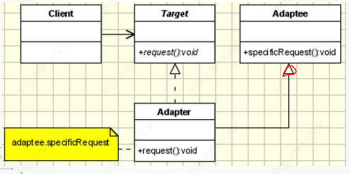
适配器模式

- 适配器模式：使一个类和接口不匹配的其他类进行交互
- 意图
  - 将一个类的接口转换为client类期望的另一个接口
  - 适配器让类可以协同工作，否则会因为接口不兼容无法进行。

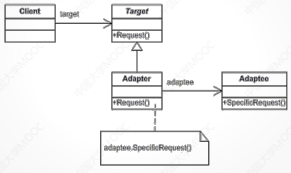
适配器模式

- 两种适配器：
  - 通过继承实现：从不兼容的类派生出新类并且添加我们需要的方法使得派生类满足预期接口。类适配器
  - 通过对象组合实现：新类中包含原始类并且在新类里创建方法以实现调用的转换。对象适配器

类适配器-基于 (多个的) 继承



对象适配器-基于组合

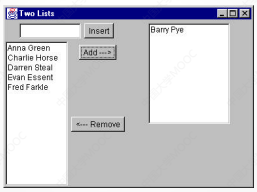


适配器模式

参与者

- Target : 定义client使用的特定于某个域的接口。
- Adapter: 将Adaptee接口适配到Target接口。
- Adaptee: 定义需被适配的已有接口
- Client： 与遵循Target接口的对象协作。

举例



使用JFC JList 类

JFC JList 类和AWT List明显不同

编写一个适配器使Jlist类看起来像List类似乎是一个明智的解决方案。

awt List class	JFC JList class
add(String);	---
remove(String)	---
String[] getSelectedItems()	Object[] getSelectedValues()

举例：

创建一个模拟List类的类。 首先，将所需的方法定义为一个接口。

```
public interface awtList {
        public void add(String s);
        public void remove(String s);
        public String[] getSelectedItems()
};
```

对象适配器

```
public class JawtList extends JScrollPane
        implements awtList {
        private JList listWindow;
        private JListData listContents;
        public JawtList(int rows) {
            listContents = new JListData();
            listWindow = new JList(listContents);
            getViewport().add(listWindow);      }
```

对象适配器

```
public void add(String s) {
                listContents.addElement(s);          }
        public void remove(String s) {
                listContents.removeElement(s);       }
        public String[] getSelectedItems() {
                Object[] obj =
            listWindow.getSelectedValues();
              String[] s = new String[obj.length];
              or (int i =0; i<obj.length; i++)
              s[i] = obj[i].toString();
              return s;   }
}
```

类适配器

```
public class JclassAwtList extends JList
    implements awtList {
    private JListData listContents;
    //-----------------------------------------
    public JclassAwtList(int rows) {
            listContents = new JListData();
            setModel(listContents);
            setPrototypeCellValue("Abcdefg
     Hijkmnop");
     }
```

Java中的适配器

在Java中有很多内置的适配器

- WindowAdapter, ComponentAdapter, ContainerAdapter, FocusAdapter, KeyAdapter, MouseAdapter, and MouseMotionAdapter.

Java中的适配器

```
public class mainFrame extends Frame
    implements WindowListener {
    public void mainFrame() {
      addWindowListener(this);
      //frame listens for window events
    }
    public void windowClosing(WindowEvent
    wEvt) {
      System.exit(0);
      //exit on System exit box clicked
    }
```

Java中的适配器

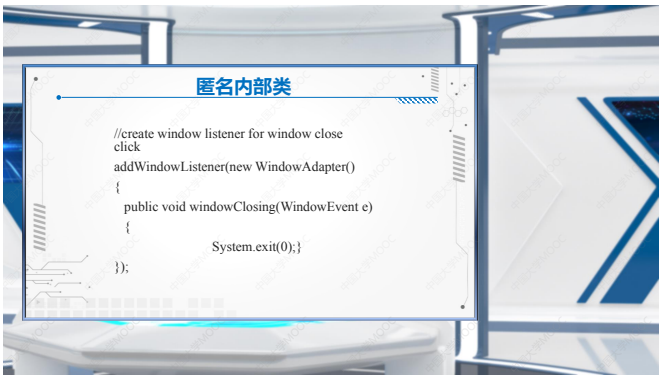
```
public void windowClosed(WindowEvent wEvt){}
public void windowOpened(WindowEvent wEvt){}
public void windowIconified(WindowEvent wEvt){}
public void windowDeiconified(WindowEvent
    wEvt){}
public void windowActivated(WindowEvent wEvt){}
public void windowDeactivated(WindowEvent
    wEvt){}
}
```

Java中的适配器

```
public class Closer extends Frame {
    public Closer() {
            WindAp windap = new WindAp();
            addWindowListener(windap);
            setSize(new Dimension(100,100));
            setVisible(true);    }
     static public void main(String argv[]) {
            new Closer();                }
}
```

Java中的适配器

```
//make an extended window adapter which
//closes the frame when the closing event is
  received
class WindAp extends WindowAdapter {
  public void windowClosing(WindowEvent e)
  {
                System.exit(0);              }
}
```


匿名内部类
//create window listener for window close
click
addWindowListener(new WindowAdapter()
{
public void windowClosing(WindowEvent e)
{
System.exit(0);}
});