软件体系结构

Zhenyan Ji

Beijing Jiaotong University

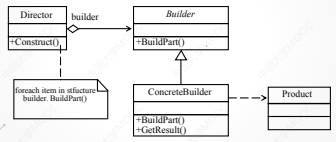
创建型模式:建造者模式

建造者模式

定义:

- 将复杂对象的构造与表示分开，以便相同的构建过程可以创建不同的表示。

建造者模式



建造者模式

参与者

Builder

- 定义抽象接口创建Product对象的部分

建造者模式

ConcreteBuilder

- 通过实现Builder接口来构建和组装产品的部分
- 定义并跟踪其创建的复杂对象表示
- 提供用于获取产品对象的接口

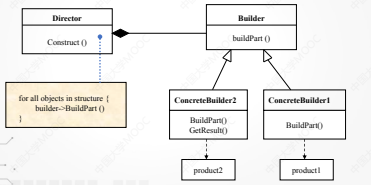
建造者模式

- Director
- 使用Builder接口构建对象

建造者模式

- Product
  - 代表正被创建的复杂对象。ConcreteBuilder构建了产品的内部表示并定义了产品的组装过程。

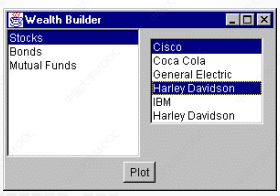
建造者模式结构



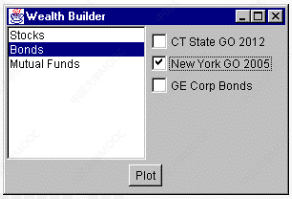
建造者模式——合作

- 客户程序创建Director对象并使用所需的Builder对象对其进行配置。
- 当产品的一部分需要被创建的时候，Director通知Builder。
- Builder处理来自主管的请求将产品part添加到产品中。
- 客户程序从Builder获取产品

例子



例子



例子

```
abstract class multiChoice {
  //This is the abstract base class that the
  //listbox and checkbox choice
  //panels are derived from
  Vector choices; //array of labels
  //-----------------------------------------
  public multiChoice(Vector choiceList) {
        choices = choiceList; //save list
  }
```

例子

```
  //to be implemented in derived classes
    abstract public Panel getUI();
  //return a Panel of components
    abstract public String[] getSelected();
  //get list of items
    abstract public void clearAll();
  //clear selections
  }
```

例子：建造者

```
class listboxChoice extends multiChoice
or
class checkBoxChoice extends multiChoice
```

例子：主管

create a simple Factory class that decides which of these two classes to return:

```
class choiceFactory {
  multiChoice ui;
  public multiChoice getChoiceUI(Vector choices){
        if(choices.size() <=3)
              ui = new checkBoxChoice(choices);
        else
              ui = new listboxChoice(choices);
        return ui;
  } }
```

建造者模式的特点

特点:

- 创建者可以改变其构建的产品的内部表示，并隐藏产品对象的组装细节。
- 每个特定的创建者独立于其他创建者和程序的其他部分。这提高了程序的模块性和可扩展性。
- 每个创建者根据数据逐步构建最终的产品对象。

建造者模式vs. 抽象工厂模式

建造者模式vs. 抽象工厂模式

- 相同点:两种模式都返回由多个方法和对象组成的类。
- 不同点:抽象工厂返回一系列相关的类；建造者根据呈现给其的数据构建一个复杂对象。