学号 E21514033 专业 软件工程 姓名 何铭春

实验日期 **4.26**  教师签字 成绩

实验报告

【实验名称】 生成器模式与原型模式编程实现

【实验目的】

1、理解并掌握生成器模式的结构和实现。

2、理解并掌握原型模式的结构和实现。

【实验原理】

1. 建造者模式（Builder Pattern）使用多个简单的对象一步一步构建成一个复杂的对象。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。一个 Builder 类会一步一步构造最终的对象。该 Builder 类是独立于其他对象的。
2. 原型模式（Prototype Pattern）是用于创建重复的对象，同时又能保证性能。这种类型的设计模式属于创建型模式，它提供了一种创建对象的最佳方式。这种模式是实现了一个原型接口，该接口用于创建当前对象的克隆。当直接创建对象的代价比较大时，则采用这种模式。

【实验内容】

抽象建造者：

abstract class Builder {

String CPU;

String RAM;

String monitor;

String keyBoard;

String mouse;

MotherBoard MB;

abstract void buildCPU();

abstract void buildRAM();

abstract void getMB();

abstract void buildMonitor();

abstract void buildKeyboard();

abstract void buildMouse();

abstract IBM getIBM();

}

具体建造者：

import java.util.\*;

public class ConcreteBuilder extends Builder {

public void buildCPU(){

Scanner in = new Scanner(System.in);

System.out.println("请输入选配的CPU型号：");

CPU = in.nextLine();

}

public void buildRAM(){

同上，输入并赋值

}

public void getMB(){

MB = new MotherBoard(CPU,RAM);

}

public void buildMonitor(){

同上，输入并赋值

}

public void buildKeyboard(){

同上，输入并赋值

}

public void buildMouse(){

同上，输入并赋值

}

}

建造指挥者：

public class Director {

Builder BD;

public Director(){}

public Director(Builder aBD){

BD = aBD;

}

//组装函数

public IBM getIBM(){

BD.buildCPU();

BD.buildRAM();

BD.getMB();

BD.buildMonitor();

BD.buildKeyboard();

BD.buildMouse();

IBM temp = new IBM(BD.MB,BD.monitor,BD.keyBoard,BD.mouse);

return temp;

}

}

主板类(实现了自身克隆):

public class MotherBoard implements Cloneable{

private String CPU;

private String RAM;

public MotherBoard(){}

public MotherBoard(String aCPU,String aRAM){

自身赋值

}

public MotherBoard clone(){

System.out.println("主板克隆方法被调用。");

return new MotherBoard(CPU,RAM);

}

public void show(){

输出自身变量

}

}

IBM类：

public class IBM implements Cloneable{

MotherBoard MB;

String monitor;

String keyBoard;

String mouse;

public IBM(){}

public IBM(MotherBoard aMB,String aMonitor,String aKeyBoard,String aMouse){

自身赋值

}

public IBM clone(){

System.out.println("电脑克隆方法被调用。");

MotherBoard tempM = MB.clone();

return new IBM(tempM,monitor,keyBoard,mouse);

}

public void setMonitor(String newMonitor){

monitor = newMonitor;

}

public void display(){

MB.show();

输出自身变量

}

}

测试类：

public class Test {

public static void main(String[] args) {

Builder test = new ConcreteBuilder();

Director hmc = new Director(test);

IBM spring = hmc.getIBM();

System.out.println("");

spring.display();

System.out.println("");

System.out.println("以下测试原型方法。");

IBM spring2 = spring.clone();

spring2.setMonitor("hello");

System.out.println("");

System.out.println("以下查看旧的对象是否改变。");

spring.display();

}

}

实验截图：



【小结或讨论】

1. 使用建造者类将零件生产固定起来，最后通过指导者类来拼装。使得整个结构不易发生改变。
2. 原型模式在于每一个类中添加克隆方法，并且注意在关联的克隆。