学号 E21514033 专业 软件工程 姓名

实验日期 **5.17**  教师签字 成绩

实验报告

【实验名称】 享元模式实现

【实验目的】

1. 熟悉享元模式应用
2. 分清外部状态和内部状态

【实验原理】

1. 保存一个已经生成的对象集合
2. 当客户端请求已经存在的对象类型时返回已经保存的类型

【实验内容】

1. web类(网站的父类)

interface Web {

void show();

}

1. 信息发布类:

public class SendMessage implements Web {

public void show(){

System.out.println("信息发布");

}

}

1. 博客类和产品展示类与信息发布类相同，仅仅输出不同
2. Customer类，用于保存自身名字，这些外部。

public class Customer implements Web {

Web aWeb;

String name;

public Customer(){}

//构造函数中传入需要的数据

public Customer(String name,Web aWeb){

this.name = name;

this.aWeb = aWeb;

}

public void show() {

System.out.print("客户"+name+":");

aWeb.show();

}

}

1. web工厂，用于管理web集合

import java.util.\*;

public class WebFactory {

public static final HashMap<String, Web> WebMap = new HashMap<>();

//获取web对象函数，利用哈希表保存对象

public static Web getWeb(String type) {

Web newWeb = (Web)WebMap.get(type);

//如果空对象，则新建并加入map

if(newWeb == null) {

if (type == "信息发布") {

newWeb = new SendMessage();

}

else if (type == "产品展示") {

newWeb = new Show();

}

else if (type == "博客留言") {

newWeb = new Blog();

}

else {

System.out.println("选择类型错误");

}

WebMap.put(type, newWeb);

}

return newWeb;

}

public static void showSize(){

System.out.println("当前保存的对象数:"+WebMap.size());

}

}

1. 测试函数

public class Test {

public static void main(String[] args) {

//初始化十个客户

Web a = new Customer("a",WebFactory.getWeb("信息发布"));

Web b = new Customer("b",WebFactory.getWeb("信息发布"));

Web c = new Customer("c",WebFactory.getWeb("信息发布"));

Web d = new Customer("d",WebFactory.getWeb("产品展示"));

Web e = new Customer("e",WebFactory.getWeb("产品展示"));

Web f = new Customer("f",WebFactory.getWeb("产品展示"));

Web g = new Customer("g",WebFactory.getWeb("博客留言"));

Web h = new Customer("h",WebFactory.getWeb("博客留言"));

Web i = new Customer("i",WebFactory.getWeb("博客留言"));

Web j = new Customer("j",WebFactory.getWeb("博客留言"));

//调用输出函数

a.show();

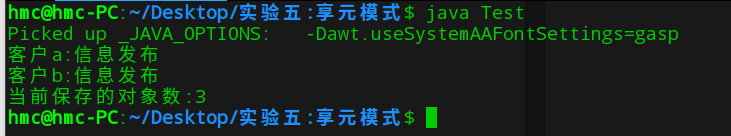
b.show();

WebFactory.showSize();

}

}

实验截图：



【小结或讨论】

1. 享元模式类似工厂模式+单件模式，使用工厂方法来控制对象的个数。
2. 对于外部状态，可以使用组合方法，在单独一个类中保存其独有的数据，和内部状态的一个对象。