

实验报告☑ 实践报告□

课程名称： 软件详细设计R

实验、实践名称： 模式综合实验

实验、实践地点： 行勉A105

专业班级： 软件2121班 学号： 2020005731

学生姓名： 张帅

指导教师： 曹若琛

2023年 4 月 5 日

目录

[一、实验目的和要求 2](#_Toc32701)

[二、实验平台 3](#_Toc26461)

[三、主要实验内容及结果 3](#_Toc11926)

[1.媒体播放器的实现 3](#_Toc25422)

[1.1 实验内容 3](#_Toc22508)

[1.2 代码实现 4](#_Toc27634)

[1.3 运行结果 9](#_Toc7083)

[2.会议管理系统的实现 9](#_Toc17616)

[2.1 实验内容 9](#_Toc28005)

[2.2 代码实现 10](#_Toc24777)

[2.3 运行结果 10](#_Toc29606)

[四、心得体会 10](#_Toc5796)

**一、实验目的和要求**

（1）实验目的

1．综合实例，熟练选用相应的多种设计模式。

2. 结合实例，使用 Java 实现多种设计模式的混用。

3. 通过实验，理解不同设计模式的使用动机，掌握不同设计模式的特点和运用场合，学习如何使用代码实现这些设计模式以及设计模式的混合使用。

（2）实验要求

1．独立完成实验。

2．书写实验报告书。

**二、实验平台**

使用JDK-11.0.1，IntelliJ IDEA 2020.3.1x64，笔记本电脑。

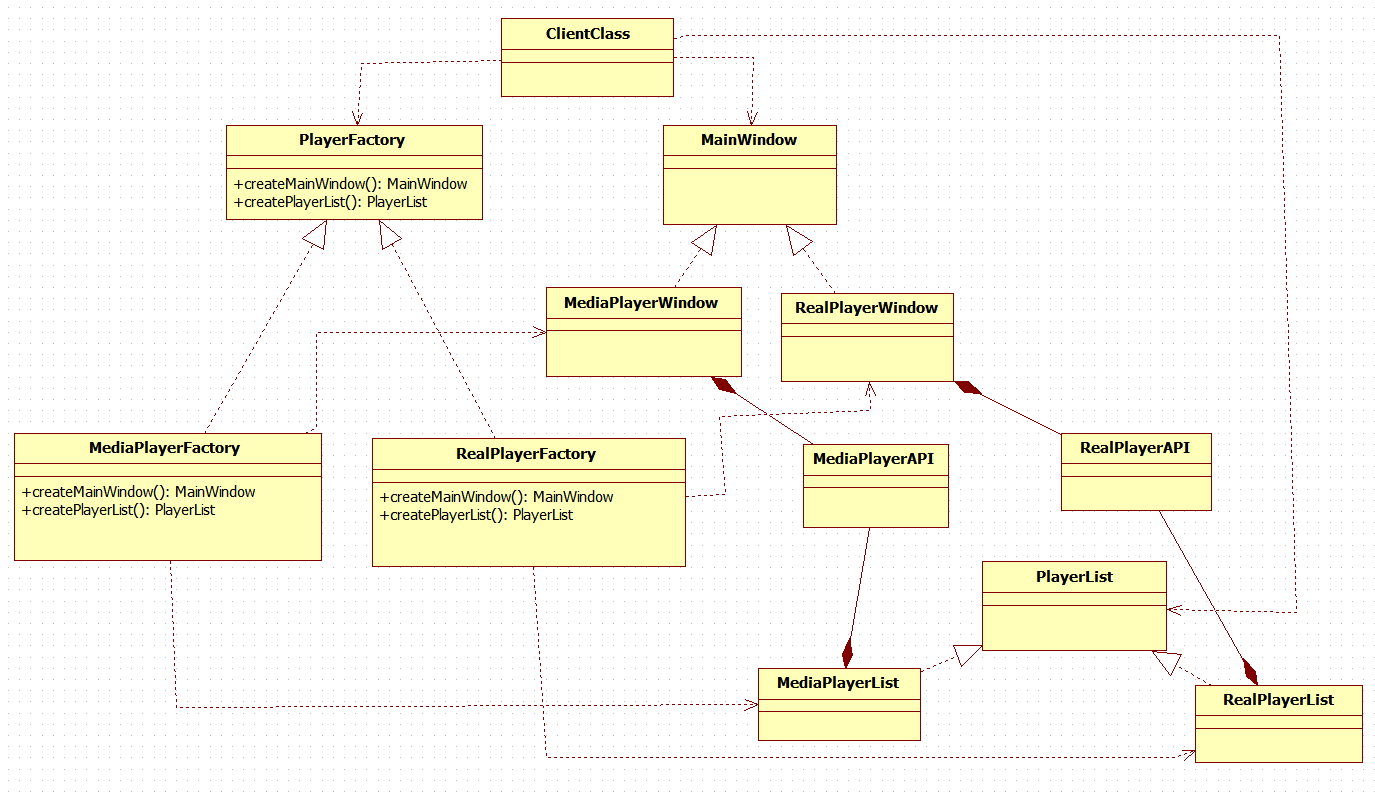
**三、主要实验内容及结果**

**1.媒体播放器的实现**

**1.1 实验内容**

（1）案例背景：

Windows Media Player 和 RealPlayer 是两种常用的媒体播放器，它们的 API 结构和调用方法存在差别，现在你的应用程序需要支持这两种播放器 API，并且将来可能还需要支持新的媒体播放器，根据给出的类图，回答该应用程序使用了哪些设计模式，并编程实现这个应用程序。



**1.2 代码实现**

**抽象工厂类 PlayerFactory：**

package edu.zhshio.one.support;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:23  
 \*/  
  
public interface PlayerFactory {  
  
 MainWindow createMainWindow();  
  
 PlayerList createPlayerList();  
  
}

**具体工厂类 MediaPlayerFactory：**

package edu.zhshio.one.playerfactory;  
  
import edu.zhshio.one.mainwindow.MediaPlayerWindow;  
import edu.zhshio.one.playerlist.MediaPlayerList;  
import edu.zhshio.one.support.MainWindow;  
import edu.zhshio.one.support.PlayerFactory;  
import edu.zhshio.one.support.PlayerList;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:35  
 \*/  
  
public class MediaPlayerFactory implements PlayerFactory {  
  
 @Override  
  
 public MainWindow createMainWindow(){  
  
 return new MediaPlayerWindow();  
  
 }  
  
 @Override  
  
 public PlayerList createPlayerList(){  
  
 return new MediaPlayerList();  
  
 }  
  
}

**具体工厂类 RealPlayerFactory：**

package edu.zhshio.one.playerfactory;  
  
import edu.zhshio.one.mainwindow.RealPlayerWindow;  
import edu.zhshio.one.playerlist.RealPlayerList;  
import edu.zhshio.one.support.MainWindow;  
import edu.zhshio.one.support.PlayerFactory;  
import edu.zhshio.one.support.PlayerList;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:30  
 \*/  
  
public class RealPlayerFactory implements PlayerFactory {  
  
 @Override  
  
 public MainWindow createMainWindow(){  
  
 return new RealPlayerWindow();  
  
 }  
  
 @Override  
  
 public PlayerList createPlayerList(){  
  
 return new RealPlayerList();  
  
 }  
  
}

**抽象窗口类 MainWindow：**

package edu.zhshio.one.support;  
  
public interface MainWindow {  
  
 public void window();  
  
}

**具体窗口类 MediaPlayerWindow：**

package edu.zhshio.one.mainwindow;  
  
import edu.zhshio.one.api.MediaPlayerAPI;  
import edu.zhshio.one.support.MainWindow;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:34  
 \*/  
  
public class MediaPlayerWindow implements MainWindow {  
  
 public MediaPlayerAPI mediaplayerAPI;  
  
 public MediaPlayerWindow() {  
  
 mediaplayerAPI=new MediaPlayerAPI();  
  
 }  
  
  
 @Override  
  
 public void window() {  
  
 System.out.println("MediaPlayerWindow");  
  
 }  
  
}

**具体窗口类 RealPlayerWindow：**

package edu.zhshio.one.mainwindow;  
  
import edu.zhshio.one.api.RealPlayerAPI;  
import edu.zhshio.one.support.MainWindow;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:34  
 \*/  
  
public class RealPlayerWindow implements MainWindow {  
  
 private RealPlayerAPI realplayerAPI=new RealPlayerAPI();  
  
  
 @Override  
  
 public void window() {  
  
 System.out.println("RealPlayerWindow");  
  
 }  
  
}

**抽象歌单类 PlayerList：**

package edu.zhshio.one.support;  
  
public interface PlayerList {  
  
 public void player();  
  
}

**具体歌单类 MediaPlayerList：**

package edu.zhshio.one.playerlist;  
  
import edu.zhshio.one.api.MediaPlayerAPI;  
import edu.zhshio.one.support.PlayerList;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:31  
 \*/  
  
public class MediaPlayerList implements PlayerList {  
  
 public MediaPlayerAPI mediaplayerAPI;  
  
 public MediaPlayerList() {  
  
 mediaplayerAPI=new MediaPlayerAPI();  
  
 }  
  
  
  
 @Override  
  
 public void player() {  
  
 System.out.println("MediaPlayerList");  
  
 }  
  
}

**具体歌单类 RealPlayerList：**

package edu.zhshio.one.playerlist;  
  
import edu.zhshio.one.api.RealPlayerAPI;  
import edu.zhshio.one.support.PlayerList;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:31  
 \*/  
  
public class RealPlayerList implements PlayerList {  
  
 private RealPlayerAPI realplayerAPI=new RealPlayerAPI();  
  
 public RealPlayerList(){  
  
 realplayerAPI = new RealPlayerAPI();  
  
 }  
  
 @Override  
  
 public void player() {  
  
 System.out.println("RealPlayerList");  
  
 }  
  
}

**API类 MediaPlayerAPI：**

package edu.zhshio.one.api;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:32  
 \*/  
  
public class MediaPlayerAPI {  
  
 public void window() {  
  
 System.out.println("MediaPlayer window");  
  
 }  
  
 public void player() {  
  
 System.out.println("MediaPlayer player");  
  
 }  
  
  
}

**API类 RealPlayerAPI：**

package edu.zhshio.one.api;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:33  
 \*/  
  
public class RealPlayerAPI {  
  
 public void window() {  
  
 System.out.println("RealPlayer window");  
  
 }  
  
 public void player() {  
  
 System.out.println("RealPlayer player");  
  
 }  
  
}

**辅助文件 XMLUtil：**

package edu.zhshio.one.util;/\*\*  
 \* @Auther: 张帅  
 \* @Date: 2024/3/7 - 03 - 07 - 11:15  
 \* @Description: edu.zhshio.builder.util  
 \* @version: 1.0  
 \*/  
  
import org.w3c.dom.Document;  
import org.w3c.dom.Node;  
import org.w3c.dom.NodeList;  
import org.xml.sax.SAXException;  
  
import javax.xml.parsers.DocumentBuilder;  
import javax.xml.parsers.DocumentBuilderFactory;  
import javax.xml.parsers.ParserConfigurationException;  
import java.io.File;  
import java.io.IOException;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/3/7 11:15  
 \*/  
  
public class XMLUtil {  
  
 public static Object getBean() {  
  
 DocumentBuilderFactory documentBuilderFactory = DocumentBuilderFactory.newInstance();  
 try {  
 DocumentBuilder documentBuilder = documentBuilderFactory.newDocumentBuilder();  
 Document document = documentBuilder.parse(new File("src/main/java/edu/zhshio/one/resource/beanconfig.xml"));  
  
 NodeList mealBuilderNames = document.getElementsByTagName("beanName");  
 Node mealBuilderName = mealBuilderNames.item(0).getFirstChild();  
 String mealBuilerNameVal = mealBuilderName.getNodeValue().trim();  
  
 Class<?> aClass = Class.forName(mealBuilerNameVal);  
 return aClass.newInstance();  
  
 } catch (ParserConfigurationException e) {  
 throw new RuntimeException(e);  
 } catch (IOException e) {  
 throw new RuntimeException(e);  
 } catch (SAXException e) {  
 throw new RuntimeException(e);  
 } catch (ClassNotFoundException e) {  
 throw new RuntimeException(e);  
 } catch (InstantiationException e) {  
 throw new RuntimeException(e);  
 } catch (IllegalAccessException e) {  
 throw new RuntimeException(e);  
 }  
 }  
}

**配置文件类 beanconfig.xml：**

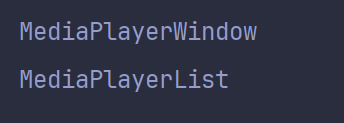
<?xml version="1.0"?>  
<config>  
 <beanName>edu.zhshio.one.playerfactory.MediaPlayerFactory</beanName>  
</config>

**用户测试类 ClientClass：**

package edu.zhshio.one;  
  
import edu.zhshio.one.support.MainWindow;  
import edu.zhshio.one.support.PlayerFactory;  
import edu.zhshio.one.support.PlayerList;  
import edu.zhshio.one.util.XMLUtil;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:37  
 \*/  
  
public class ClientClass {  
  
 public static void main(String[] args) {  
  
  
 PlayerFactory playerFactory= (PlayerFactory) XMLUtil.getBean();  
  
 MainWindow mainWindow = playerFactory.createMainWindow();  
  
 mainWindow.window();  
  
 PlayerList playerList = playerFactory.createPlayerList();  
  
 playerList.player();  
  
 }  
  
}

**1.3 运行结果**

该应用程序使用了工厂方法模式和适配器模式。



**2.会议管理系统的实现**

**2.1 实验内容**

（1）案例背景：

某会议管理系统的“会议通知发送”模块说明如下：

1. 行政管理人员可以给某个或者某些员工（Employee）发送会议通知，也可以给某个部门（Department）发送通知，如果给某个部门发送通知，将逐个给该部门每个员工发送会议通知。

（2）如果员工或者部门希望能够收到会议通知，必须先注册到一个会议列表（MeetingList）中，在发送通知时，系统将遍历会议列表，逐个将会议通知发送给注册用户（User）。

（2）实现步骤：

 根据题意，选用合适的两种设计模式。

 根据这两种设计模式，画出本题的类图。

 根据类图，实现“会议通知发送”模块的代码。**（附加题，会根据实现情况加平时成绩）**

（3）案例总结：

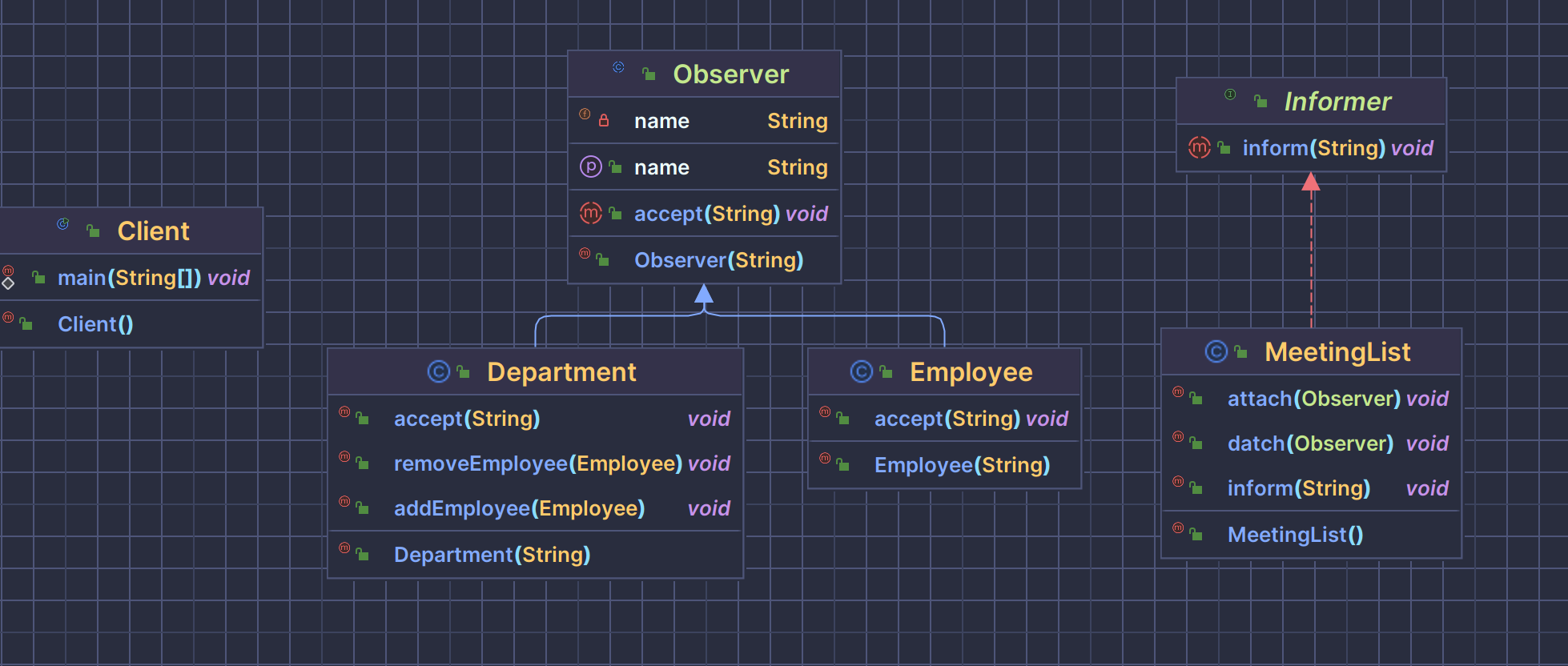
 设计模式的选用需要根据具体要求来进行。

 多种设计模式可以进行混合使用。

 混合使用时，需要根据选用的不同设计模式，对类图进行修改。

**2.2 代码实现**

绘制类图：



**抽象观察者 Observer：**

package edu.zhshio.two.support;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:49  
 \*/  
  
public abstract class Observer {  
  
  
 private String name;  
  
 public abstract void accept(String message);  
  
 public String getName() {  
 return this.name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Observer(String name) {  
 this.name = name;  
 }  
}

**抽象通知类 Inform:**

package edu.zhshio.two.support;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:56  
 \*/  
  
public interface Informer {  
  
 public void inform(String message);  
  
}

**具体观察者员工类 Employee：**

package edu.zhshio.two.observers;  
  
import edu.zhshio.two.support.Observer;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:57  
 \*/  
  
public class Employee extends Observer {  
  
 public Employee(String name) {  
 super(name);  
 }  
  
 @Override  
 public void accept(String message) {  
 System.out.println(getName() + " 接收到会议消息:" + message);  
 }  
}

**具体观察者类员工 Department：**

package edu.zhshio.two.observers;  
  
import edu.zhshio.two.support.Observer;  
  
import java.util.Map;  
import java.util.TreeMap;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午8:59  
 \*/  
  
public class Department extends Observer {  
  
  
 private Map<String, Observer> subFiles = new TreeMap<>();  
  
 @Override  
 public void accept(String message) {  
 String departmentName = getName();  
 System.out.println("---------------------" + departmentName + "---------------------");  
 subFiles.values()  
 .stream()  
 .forEach(  
 observer ->{  
 observer.accept(message);  
 }  
 );  
 }  
  
 public void addEmployee(Employee employee) {  
 subFiles.put(employee.getName(), employee);  
 }  
  
 public void removeEmployee(Employee employee) {  
 subFiles.remove(employee.getName());  
 }  
  
  
 public Department(String name) {  
 super(name);  
 }  
}

**具体消息通知类 MeetingList：**

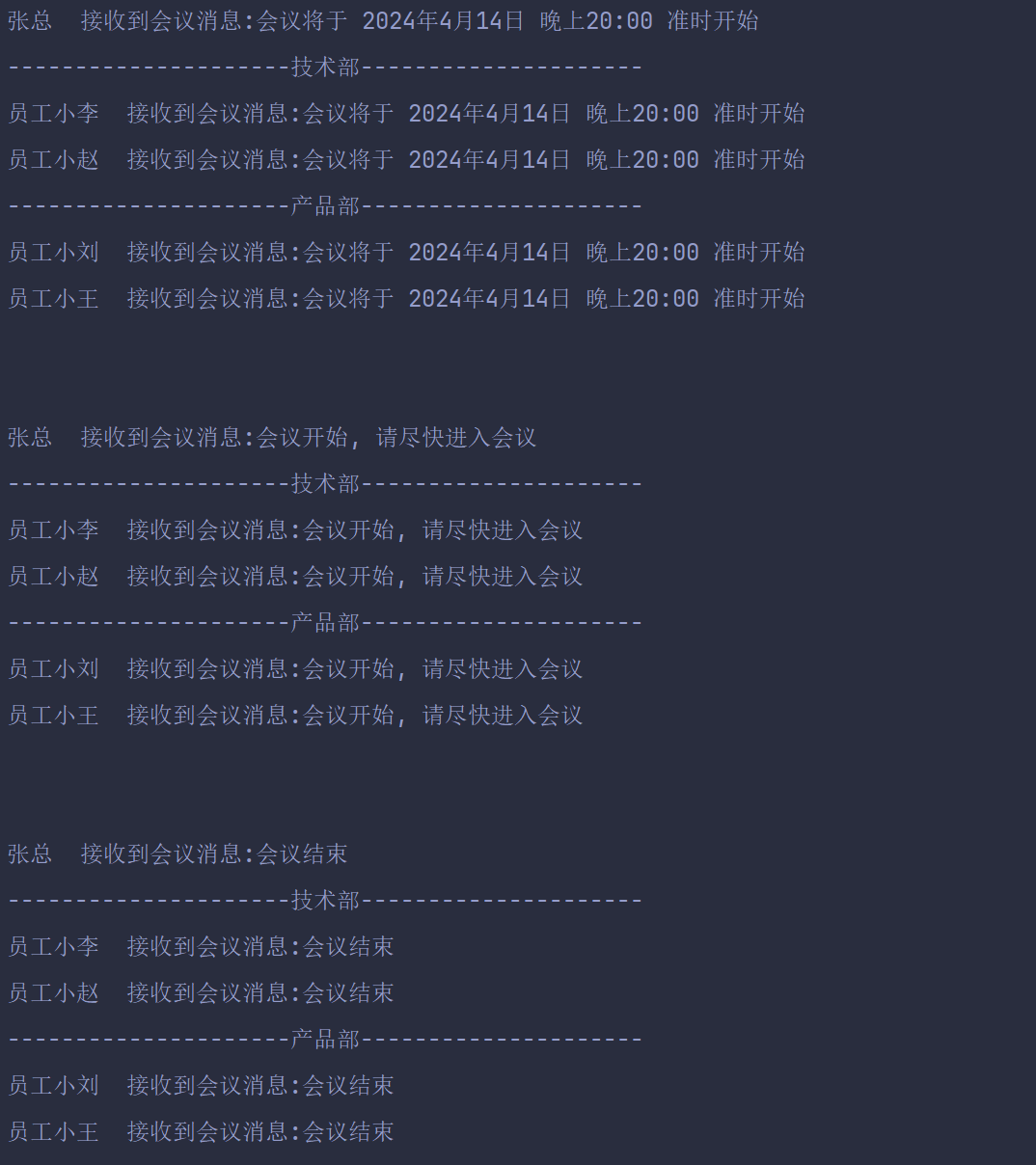
package edu.zhshio.two.informer;  
  
  
import edu.zhshio.two.support.Informer;  
  
import edu.zhshio.two.support.Observer;  
  
import java.util.LinkedList;  
import java.util.List;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午9:06  
 \*/  
  
public class MeetingList implements Informer {  
  
 private List<Observer> observers;  
  
 private String message;  
  
 public void attach(Observer observer) {  
 observers.add(observer);  
 }  
  
 public void datch(Observer Observer) {  
 observers.removeIf(observer -> Observer.equals(observer));  
 }  
  
 @Override  
  
 public void inform(String message) {  
 observers.stream()  
 .forEach(observer -> {  
 observer.accept(message);  
 });  
 }  
  
  
 public MeetingList() {  
 this.observers = new LinkedList<>();  
 }  
}

**用户测试类 Client：**

package edu.zhshio.two;  
  
import edu.zhshio.two.informer.MeetingList;  
import edu.zhshio.two.observers.Department;  
import edu.zhshio.two.observers.Employee;  
  
/\*\*  
 \* @description:  
 \* @author: zs  
 \* @time: 2024/4/14 下午9:14  
 \*/  
  
public class Client {  
 public static void main(String[] args) {  
  
 Employee employee1 = new Employee("员工小赵");  
 Employee employee2 = new Employee("员工小李");  
 Employee employee3 = new Employee("员工小刘");  
 Employee employee4 = new Employee("员工小王");  
 Employee employee5 = new Employee("张总");  
  
 Department developmentDepartment = new Department("技术部");  
 developmentDepartment.addEmployee(employee1);  
 developmentDepartment.addEmployee(employee2);  
 Department productDepartment = new Department("产品部");  
 productDepartment.addEmployee(employee3);  
 productDepartment.addEmployee(employee4);  
  
 MeetingList meetingList = new MeetingList();  
 meetingList.attach(employee5);  
 meetingList.attach(developmentDepartment);  
 meetingList.attach(productDepartment);  
  
 meetingList.inform("会议将于 2024年4月14日 晚上20:00 准时开始");  
 System.out.println("\n\n\n");  
 meetingList.inform("会议开始, 请尽快进入会议");  
 System.out.println("\n\n\n");  
 meetingList.inform("会议结束");  
  
 }  
}

**2.3 运行结果**

该应用程序使用了观察者模式和组合模式。



**四、心得体会**

最后两个实验是综合类应用，相较于前面的实验这个确实一眼看不出来怎么做，不过根据给出的类图，仔细思考之后还是可以做出来的，也就是多种模式的一个混用，重点是要熟练掌握前面所学的设计模式，然后带入进去就可以了。

以下是我对这两个小实验的总结：

第一个是媒体播放器，根据类图，不难看出播放器工厂这里是用了工厂方法模式，一个 MediaPlayerFactory，另一个 RealPlayerFactory，工厂里面要生产窗口和歌单，首先定义一个抽象窗口类和一个抽象歌单类，由于我们的窗口或歌单需要同时满足不同品牌的媒体播放器，不同品牌的播放器拥有不同的API结构和调用方法，此时就应该想到这里需要的是一个适配器类，让它去实现目标类与被适配者之间的联系。具体窗口类和具体歌单类作为适配器类，它们继承自目标类，并重写目标类中的抽象方法，API类作为被适配者类，里面定义了两种媒体播放器各自的特有方法，对于适配器我们可以选择类适配器模式，也可以选择对象适配器模式，因为我们对目标类用的是继承关系，而Java又不支持多继承，所以这里就只能用对象适配器模式了，在适配器中以成员变量的方式引入API类，并创建API对象，然后调用其方法，最后在测试类中通过辅助类选择要执行的工厂名称，调用方法创建对象就好了。

第二个是会议管理系统，根据题意，行政管理人员是要发布会议通知的，这里直接接收到会议通知的对象可以是员工也可以是部门，如果是部门，部门将逐个给它下面的每个员工再发送会议通知。首先发布通知和接收通知这两个动作应该用观察者模式实现，发布者为目标类，接收者为观察者类，观察者可以是员工也可以是部门，所以这里用一个抽象类来实现它，员工和部门之间的关系，像文件与文件夹之间的关系，即组合模式。部门类中提供了添加删除及查找员工的方法，并重写accept方法，再递归调用其员工的accept方法。测试类中添加meetingList并调用inform方法即可。