# 实验三：适配器模式

## 实验内容

被适配的类可以把一个Calendar类型的日期按照类似于“2020年11月25日”格式输出。

现输入类似“2020.11.2”的内容，使用对象适配器和类适配器完成按照格式的输出。

## 对象适配器模式

### UML设计



### 核心代码

ObjectAdapter.java

public class ObjectAdapter implements Target{

private calendarOp calendarOpDemo;

public ObjectAdapter(calendarOp calendarOpDemo) {

this.calendarOpDemo = calendarOpDemo;

}

@Override

public void op(String calendarString) {

// TODO Auto-generated method stub

Calendar calandar = calendarOpDemo.opCalender(calendarString);

Adaptee adaptee = new Adaptee();

System.out.println("对象适配器：");

adaptee.op(calandar);

}

}

Target.java

public interface Target {

void op(String calendarString);

}

Adaptee.java

public class Adaptee {

public void op(Calendar calendar) {

System.out.println("你好，我是被适配的类，你输入的日期是：" + calendar.get(Calendar.YEAR) + "年" + (calendar.get(Calendar.MONTH)+1) +"月" + calendar.get(Calendar.DATE) +"日");

}

}

calendarOp.java

public class calendarOp {

/\*

\* 该类的函数是负责把输入的表示年月日的字符串转化为日期对象

\*/

public Calendar opCalender(String calendarString) {

Calendar calendar = Calendar.getInstance();

String[] calendarStrings = calendarString.split("\\.");

int year = Integer.parseInt(calendarStrings[0]);

int month = Integer.parseInt(calendarStrings[1]) - 1;

int day = Integer.parseInt(calendarStrings[2]);

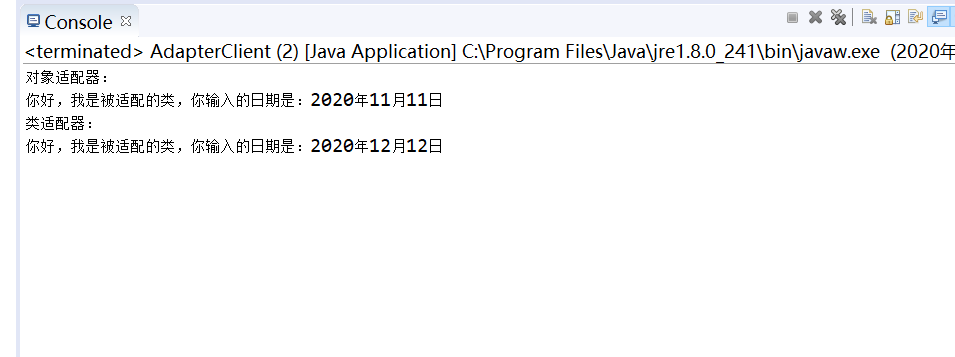
calendar.set(year,month,day);

return calendar;

}

}

### 实现效果



## 类适配器模式

### UML设计



### 核心代码

ClassAdapter.java

public class ClassAdapter extends Adaptee implements Target{

private calendarOp calendarOpDemo;

public ClassAdapter(calendarOp calendarOpDemo) {

this.calendarOpDemo = calendarOpDemo;

}

@Override

public void op(String calendarString) {

// TODO Auto-generated method stub

Calendar calendar = calendarOpDemo.opCalender(calendarString);

System.out.println("类适配器：");

op(calendar);

}

}

Target.java

public interface Target {

void op(String calendarString);

}

Adaptee.java

public class Adaptee {

public void op(Calendar calendar) {

System.out.println("你好，我是被适配的类，你输入的日期是：" + calendar.get(Calendar.YEAR) + "年" + (calendar.get(Calendar.MONTH)+1) +"月" + calendar.get(Calendar.DATE) +"日");

}

}

calendarOp.java

public class calendarOp {

/\*

\* 该类的函数是负责把输入的表示年月日的字符串转化为日期对象

\*/

public Calendar opCalender(String calendarString) {

Calendar calendar = Calendar.getInstance();

String[] calendarStrings = calendarString.split("\\.");

int year = Integer.parseInt(calendarStrings[0]);

int month = Integer.parseInt(calendarStrings[1]) - 1;

int day = Integer.parseInt(calendarStrings[2]);

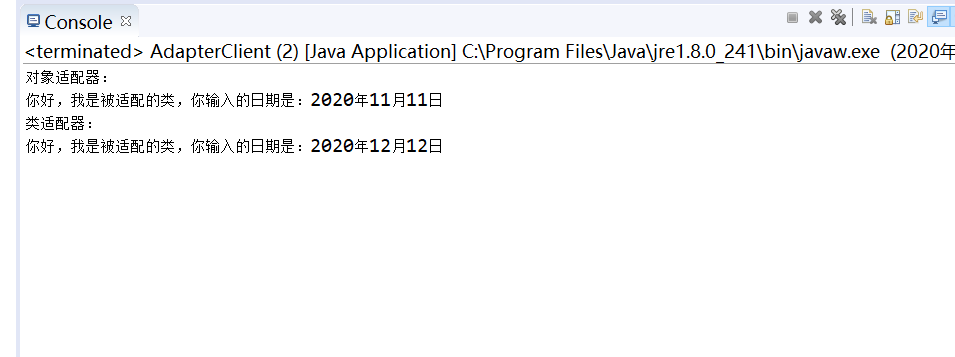
calendar.set(year,month,day);

return calendar;

}

}

### 实现效果



## 实验体会

这次的适配器模式，将一个类的接口变换成客户端所期待的另外一种接口，从而使原本因接口不匹配而无法在一起工作的两个类在一起工作。这次的实验学习到了一种新的设计模式，受益匪浅。