

Java程序设计实验报告

实验报告

【实验题目】 Java实验5

#### 【实验内容】

##### 1. 练习1

**实验目的**实现接口回调技术

**实验要求**巨型货轮需要运输一批贵重的货物，里面装有四种不同的物品：跑车、直升机、帆船以

及雕像。由于物品较为贵重，在出发前需要计算所有装载的重量以便安排存放空间。

(1) 设计一个名为 WeighCargo 接口，其包括如下方法： Public double weigh()

(2) 设计四个实现该接口的类 Roadster，Helicopter，Galleon 以及 Statue，通过以上四个类来

实现 WeighCargo 接口来计算出该类物品的重量。

(3) 设计一个 GiantFreighter 类，该类中采用 WeighCargo 接口类型的数组作为成员，那么该

数组的单元就可以存放 Roadster 对象的引用、Helicopter 对象的引用，Galleon 对象的引用，

或者 Statue 对象的引用。本程序最后将输出巨型货轮中运输物品的总重量.

运行效果示例：巨型货轮运送的货物重量为 5139.7 吨。

###### 代码：

**WeighCargo.java**

package Weight;

public interface WeighCargo {

    public double weigh();// 计算重量的抽象方法

}

**Roadster.java**

package Weight;

public class Roadster implements WeighCargo {

    public double weigh() {

        return 288.4;// 返回跑车重量

    }

}

**Helicopter.java**

package Weight;

public class Helicopter implements WeighCargo {

    public double weigh() {

        return 1288.4;// 返回直升机重量

    }

}

**Galleon.java**

package Weight;

public class Galleon implements WeighCargo {

    public double weigh() {

        return 988.56;// 返回帆船的重量

    }

}

**Statue.java**

package Weight;

public class Statue implements WeighCargo {

    public double weigh() {

        return 218.0;// 返回雕像重量

    }

}

**GiantFreighter.java**

package Weight;

public class GiantFreighter {

    WeighCargo[] wc;//定义WeighCargo数组

    GiantFreighter(WeighCargo[] a) {//传入数组参数

        wc = a;

    }

}

**Test1.java**

package Weight;

public class Test1 {

    public static void main(String[] args) {

        WeighCargo wc[] = new WeighCargo[4];// 定义WeighCargo类型的数组

        wc[0] = new Roadster();// 接收Roadster重量返回值

        wc[1] = new Helicopter();// 接收Helicopter重量返回值

        wc[2] = new Galleon();// 接收Galleon重量返回值

        wc[3] = new Statue();// 接收Statue重量返回值

        GiantFreighter gf = new GiantFreighter(wc);

        double sum;

        sum = wc[0].weigh() + wc[1].weigh() + wc[2].weigh() + wc[3].weigh();// 四个相加就是总和

        System.out.println("总工重量:" + sum + "吨");

    }

}

###### 输出结果：



###### 分析：

接口中的抽象方法在子类重写，分开计算每种物品的重量并返回，最后在main函数中加起来就可以求出总和

##### 2.练习2

**实验目的**掌握面向接口编程思想

**实验要求**青蛙的生长状态是随着环境的变化而变化的，首先是小尾蝌蚪，依次长出两条后腿和两条前腿，然后变成没有尾巴的小青蛙，最后变成青蛙等五个过程，要求用接口封装青蛙变化的状态。具体要求如下。

1）编写一个 FrogState 接口，该接口中有名为 void showState()的方法。

2）编写 Frog 类，该类中有一个 FrogState 接口申明的变量 state。另外，该类有一show()

方法，在该方法中让接口 state 回调 showState()方法

3）编写若干个实现 FrogState 接口的类（自己根据青蛙的状态确定类的名称），负责刻画

青蛙的各种时期的状态。

4）编写主类，在主类中测试青蛙的各种状态。

###### 代码：

**FrogState.java**

package Grow;

public interface FrogState {

    void showState();//打印状态的抽象函数

}

**Frog.java**

package Grow;

public class Frog {

    private FrogState state;//定义FrogState接口申明的变量

    public void show() {//重写方法

        state.showState();

    }

    public void setState(FrogState fs) {//设置参数

        state = fs;

    }

}

**Tadpole.java**

package Grow;

public class Tadpole implements FrogState {

    public void showState() {//重写showState方法

        System.out.println("现在只是一只小蝌蚪。。。");

    }

}

**Fleg.java**

package Grow;

public class Fleg implements FrogState {

    public void showState() {//重写showState方法

        System.out.println("现在长出了两条前腿。。。");

    }

}

**Bleg.java**

package Grow;

public class Bleg implements FrogState {

    public void showState() {//重写showState方法

        System.out.println("现在长出了两条后腿。。。");

    }

}

**BeForg.java**

package Grow;

public class BeFrog implements FrogState {

    public void showState() {//重写showState方法

        System.out.println("现在变成了小青蛙。。。");

    }

}

**Test2.java**

package Grow;

public class Test2 {

    public static void main(String[] args) {

        Frog f = new Frog();//new一个Frog对象

        f.setState(new Tadpole());//传入Tadpole类对象

        f.show();

        f.setState(new Fleg());//传入Fleg类对象

        f.show();

        f.setState(new Bleg());//传入Bleg类对象

        f.show();

        f.setState(new BeFrog());//传入BeFrog类对象

        f.show();

    }

}

###### 输出结果：

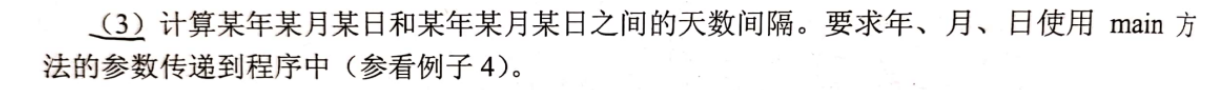
###### 

###### 分析：

先写一个接口FrogState，里面包含一个抽象方法，之后在每一个类里面实现接口中的抽象类，在回调方法的过程中，传入哪一个类的参数就会调用某一个类里面的showState方法，打印对应的状态

##### 练习3

**实验目的**熟练使用main方法的参数和日期的计算

**实验要求**

###### 代码：

**CountDays.java**

import java.util.Calendar;

import java.util.Scanner;

public class CountDays {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        args = new String[6];//创建对象，存储日期

        for (int i = 0; i < 6; i++) {

            args[i] = sc.next();//输入两次年月日

        }

        Calendar date1 = Calendar.getInstance();//在getInstance方法内部实例化了Calendar对象并返回了

        date1.set(Calendar.YEAR, Integer.parseInt(args[0]));//设置年

        date1.set(Calendar.MONTH, Integer.parseInt(args[1]));//设置月

        date1.set(Calendar.DATE, Integer.parseInt(args[2]));//设置日

        Calendar date2 = Calendar.getInstance();

        date2.set(Calendar.YEAR, Integer.parseInt(args[3]));//设置年

        date2.set(Calendar.MONTH, Integer.parseInt(args[4]));//设置月

        date2.set(Calendar.DATE, Integer.parseInt(args[5]));//设置日

        long delta = Math.abs(date1.getTimeInMillis() - date2.getTimeInMillis());//转换成微秒进行计算

        int days = (int) (delta / 1000 / 60 / 60 / 24);//换算成具体天数

        System.out.println("从" + date1.get(Calendar.YEAR) + "-" + date1.get(Calendar.MONTH) + "-"

                + date1.get(Calendar.DATE) + "到" + date2.get(Calendar.YEAR) + "-" + date2.get(Calendar.MONTH) + "-"

                + date2.get(Calendar.DATE) + "总共间隔：" + days + "天");//打印间隔天数

    }

}

###### 输出结果：



###### 分析：

由于自己计算日期间隔过于麻烦，还需要考虑是否闰年问题，但是java提供了Calendar抽象类，这个类可以进行日期的计算，知道这个就简单了，我们直接实例化对象之后，设置起始日期和终止日期就好了，直接转换成微秒进行计算，之后再转换成天数就好了

##### 练习4

###### 实验目的

学会Scanner类的使用

###### 实验要求

###### 

###### 代码：

import java.util.Scanner;

public class Replace {

    public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        System.out.println("请输入字符串：");

        String s1, s2;

        s1 = sc.nextLine();// 输入一个字符串

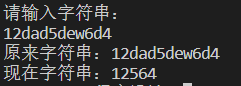
        s2 = s1.replaceAll("[a-zA-z]+", "");// 只保留数字，其中第一个“”中内容换成第二个“”中内容

        System.out.println("原来字符串：" + s1 + "\n" + "现在字符串：" + s2);

    }

}

###### 输出结果：



###### 分析：

直接使用Scanner类中的replaceAll方法进行无关字符串的替换

##### 练习5

###### 实验目的

学会Scanner类的使用

###### 实验要求

###### 代码：

import java.util.Scanner;

public class Score {

    public static void main(String args[]) {

        String cost = "数学87分，物理76分，英语96分";

        Scanner sc = new Scanner(cost);//传入值

        sc.useDelimiter("[^0123456789.]+");//设置分隔符

        double sum = 0, aver = 0;//总分；平均分

        while (sc.hasNext()) {//判断字符串是否结束

            double score = sc.nextDouble();//将数字输出给score

            sum += score;//累加求和

        }

        aver = sum / 3;//求平均

        System.out.println("总分:" + sum + "分");

        System.out.println(String.format("平均分：%.2f分", aver));

    }

}

###### 分析：

直接使用Scanner类中的useDelimiter方法只提取后面括号中指定的字符，本题就是提取数字

#### 【小结与讨论】

1.主要就是日期计算，如果真的要自己写方法，要考虑到是否是闰年，还要考虑月份相减可能还要考虑天数，毕竟不一定满一个月，就如7月24日和6月25日，是没办法直接用月份减月份的，还得看天数，这样实现的话可能需要定义两个数组，一个存闰年每个月的天数，另一个存非闰年每个月天数，这还得另外写一个函数判断是否为闰年，麻烦程度可想而知，为了能简化工作量，我们可以使用java中现成的Calendar类，其中有专门计算日期的方法，我们只需要进行时期设定就好了