# 四到六章作业

——周玉川 2017221302006

第4章：

1、创建一个类，它有一个接受一个String参数的构造器。在构造阶段，打印该参数。创建一个该类的对象引用数组，并通过以下两种方式对该数组初始化：1、构造函数，2、创建对象赋值给该引用数组。

|  |
| --- |
| **package** 第4章;  //导入包java.util的类接口  **import** java.util.\*;  //创建类Sample  **class** Sample{  //带参数的构造函数  Sample( String x ) {  System.***out***.println(x);  }  //不带参数的构造函数  Sample(){  **this**("No String");  }  }  **public** **class** 第一题 {  **public** **static** **void** main ( String[] args) {  Sample[] sam = **new** Sample[2];  Scanner x = **new** Scanner(System.***in***);  //利用new 创建对象赋值给引用数组 sam[2]  sam[0] = **new** Sample(x.nextLine());  sam[1] = **new** Sample();  x.close();  }  } |

2、以学生类为例，基本成员包括：学号、姓名、性别、年龄、爱好，其中爱好为枚举类型，为Football,Basketball,Dancing,Music的集合。

试在定义学生类基础上实现构造函数的过载，并实现不访问学生对象具体成员，用System.out.println()方法对学生对象的直接字符串输出显示功能。

|  |
| --- |
| **package** 第4章;  //爱好为枚举类型，为Football,Basketball,Dancing,Music的集合  **enum** Hobby {  ***Football***,***Basketball***,***Dancing***,***Music***  }  //学生类  **class** Student{  String idNumber;//学号  String name;//姓名  **int** year;//年龄  Hobby hobby;//爱好，枚举类型  //构造函数过载  Student(String idNumber, String name,**int** year,Hobby hobby){  **this**.idNumber=idNumber;**this**.name=name;**this**.year=year;**this**.hobby=hobby;  }  Student(String idNumber, String name,Hobby hobby){  **this**.idNumber=idNumber;**this**.name=name;**this**.hobby=hobby;  }  Student(String idNumber, String name,**int** year){  **this**.idNumber=idNumber;**this**.name=name;**this**.year=year;  }  Student(**int** year,Hobby hobby){  **this**.year=year;**this**.hobby=hobby;  }  Student(String idNumber, String name ){  **this**.idNumber=idNumber;**this**.name=name;  }  Student(**int** year){  **this**.year = year;  }  Student (Hobby hobby){  **this**.hobby = hobby;  }  **void** Print(){  System.***out***.println("idNumber:"+idNumber+"\nname:"+name+"\nyear:"+year+"\nhobby:"+hobby);  }  }  **public** **class** 第二题 {  **public** **static** **void** main(String[] args) {  Student test = **new** Student("2017221302006","周玉川",19,Hobby.***Dancing***);  test.Print();  }  } |

3、以下程序输出结果是什么？试解析原因。去掉注释语句后，输出结果是什么？试解析原因。

class Cups{

static Cup cup1;

static Cup cup2;

static {

cup1=new Cup(1);

cup2=new Cup(2);

}

Cups(){

System.out.println("Cups()");

}

}

class Cup{

Cup(int marker){

System.out.println("Cup("+marker+")");

}

void f(int marker){

System.out.println("f("+marker+")");

}

}

public class ExplicitStatic{

public static void main(String[] args) {

System.out.println("Inside main()");

Cups.cup1.f(99);

}

//static Cups cup1 = new Cups();

//static Cups cup2 = new Cups();

}

|  |
| --- |
| 输出:  Inside main()  Cup(1)  Cup(2)  f(99)  去掉注释后输出:  Cup(1)  Cup(2)  Cups()  Cups()  Inside main()  f(99)  原因:  去掉注释后，  在类ExplicitStatic中，先对cup1实例化，进入类Cups中，实例化(Cup)cup1,cup2,在类Cup中输出Cup(1)，Cup(2),然后在类Cups中输出Cup(),然后在类ExplicitStatic实例化cup2，因为  Static{}会在类被加载的时候执行且仅会被执行一次，初始化静态变量cup1,cup2近一次，所以在初始化cup2时，只会输出Cup(),在main中，输出Inside main()，同样不会执行类Cups中cup1，cup2的实例化，只输出f(99)。 |

4、P98-4、10

|  |
| --- |
| 第四题：  数据传递分为形式参数为简单类型，形式参数为引用类型。简单类型仅仅是将值赋给形式参数，不会影响实际参数的值，而引用类型，传递数据的地址，可能会改变实际参数的值。 |
| 第五题：  方法过载又称为方法重载，它是指在一个类中有多个方法重名。方法过载的方法具有相同的方法名，但是不同的参数。  举例：  void f(int i);void f(float i);void f(int a,int b); |
| 第六题:  类成员在编译时就有了确定的储存空间，而实例成员随着类的实力创建而实例化。  类成员使用 <类名>.<类成员名>  实例成员使用方法 <对象名>.<实例成员名> |
| 第七题:  对象是类的实例，对象变量是对类的实例引用，类为对象的创建提供了模板。  对象的引用即对象的创建，使用和回收。 |
| 第八题：  对象的创建是声明的对象对类的实例引用，它包含有三个方面的内容：对象的声明，类的实例创建和对象对实例引用。  对象实例化后就可以实现了对对象的使用。  当对象未被引用时被当作垃圾清理掉。 |
| 第九题：  **package** 第二次实验;  **class** MyDate{  **int** year;  String month;  **int** day;  //设置日期  **void** SetDate(**int** year, String month, **int** day){  **this**.year = year;  **this**.month = month;  **this**.day = day;  }  //增加日期  **void** AddDate(**int** add){  **this**.day+=1;  }  //打印日期  **void** PrintData(){  System.***out***.println("year: "+year+"\nmonth: "+month+"\nday: "+day);  }  }  **public** **class** TestDate {  **public** **static** **void** main(String[] args){  MyDate test = **new** MyDate();  test.SetDate(2017, "October", 22);  test.PrintData();  }  } |
| 第十题：  **package** 第二次实验;  **class** ArraySort{  **int**[] sim = **new** **int**[20];  //构造函数中利用随机数初始化数组  ArraySort(){  **for** ( **int** i = 0 ; i < 20 ; i++ ){  sim[i] = (**int**)(Math.*random*()\*100+1);  }  }  //对数组排序  **void** setOrder(){  **int** i,j;  **int** temp;  //冒泡排序  **for** ( i = 0 ; i < 20 ; i ++){  **for** ( j = i+1 ; j < 20 ; j++){  **if** (sim[i] > sim[j]){  temp=sim[i];  sim[i]=sim[j];  sim[j]=temp;  }  }  }  }  }  **public** **class** 第四章10 {  **public** **static** **void** main(String [] args){  ArraySort test = **new** ArraySort();  //排序前  System.***out***.println("Before sort: ");  **for** ( **int** i = 0 ; i < 20 ; i++ ){  System.***out***.print(test.sim[i]+",");  }  test.setOrder();  //排序后  System.***out***.println("\nAfter sort: ");  **for** ( **int** i = 0 ; i < 20 ; i++ ){  System.***out***.print(test.sim[i]+",");  }  }  } |

第5章：

1、P112-5、7、8

|  |
| --- |
| 第五题：  Java中的多态：子类对超类的隐藏和覆盖，方法过载， |
| 第七题：  Final类不能被继承，而abstract修饰的类必须被继承 |
| 第八题：  Equals()比较两个对象的类型和值是否相同,返回boolean类型。  举例:  String s1 = new String(“xx”);  String s2 = new String(“xx”);  s1.equals(s2)返回true，但是s1==s2返回false |

2、创建一个超类，它仅有一个非默认的构造器；再创建一个子类，它带有默认构造器和非默认构造器。在子类的构造器中调用超类的构造器。

|  |
| --- |
| **package** 第五章;  **class** father{  father(){  System.***out***.println("I love xinxin");  }  }  **class** son **extends** father{  son(){  **super**();  }  }  **public** **class** 第二题 {  **public** **static** **void** main(String[] args) {  son test = **new** son();  }  } |

3、创建一个抽象学生类，其中具有学生信息及print方法，创建学生类的子类分别为研究生类和本科生类，其中覆盖超类的print方法，通过超类引用实现多态。

|  |
| --- |
| **package** 第五章;  //超类student  **abstract** **class** Student{  String idNum;  String name;  **void** print() {  System.***out***.println("IdNum: "+idNum+"\nname: "+name);  }  }  //子类  **class** Undergrade **extends** Student{  **void** print() {  System.***out***.println("Undergrade:");  **super**.print();  }  }  //子类  **class** Aftergrade **extends** Student{  //覆盖超类print()函数  **void** print() {  System.***out***.println("Aftergrade:");  //多态  **super**.print();  }  }  **public** **class** 第三题 {  **public** **static** **void** main(String[] args) {  Undergrade test = **new** Undergrade();  test.idNum="2017221302006";test.name="周玉川";  test.print();  }  } |

第6章：

1、P129-1、2

|  |
| --- |
| 第一题：  包是一系列类的集合，也叫做类库。  直接import <包名>;放在package下面就可以使用。 |
| 第二题:  Package指明该文件中定于的类所在的包，缺省则制定为无名包。  Import语句来引入已经创建的所需要的类 |

2、P130-5、6

|  |
| --- |
| 第五题：  有四种:public,protected,缺省型，private  使用时直接写在类，变量，方法定义之前 |
| 第六题：  抽象类和接口的区别主要在:   1. java不支持子类多继承，但是可以实现多接口。 2. 在默认的方法实现上：抽象类可以有默认的方法实现，但是接口完全是抽象的。它根本不存在方法的实现。 3. 抽象类可以有构造器，接口不能有构造器。 4. 抽象方法可以有**public**、**protected**和**default**这些修饰符，接口方法默认修饰符是**public**。你不可以使用其它修饰符。 5. 如果基本功能在不断改变，那么就需要使用抽象类。如果不断改变基本功能并且使用接口，那么就需要改变所有实现了该接口的类。 |

3、以学生对象为基本元素，在P126-例6.3和P127-例6.4基础上，实现完整的先进先出队列代码。

|  |
| --- |
| **package** 第六章;  //学生对象  **class** Student{  String name;  String idNum;  }  //接口interface  **interface** Collection{  **int** ***MAX\_NUM*** = 100;  **void** add(Student objAdd);  **void** delete(Student objDelet);  **boolean** find(Student objFind);  **int** currentCount();  }  //队列  **class** FIFOQueue **implements** Collection{  **int** rear=-1,front=-1;  Student[] students = **new** Student[***MAX\_NUM***];  //添加  **public** **void** add(Student objAdd) {  **if** (rear ==***MAX\_NUM***-1) {  System.***out***.println("No space");  }  **else** {  rear++;  students[rear] = objAdd;  }  }  //删除  **public** **void** delete(Student objDelet) {  front ++;  }  //判断存在  **public** **boolean** find(Student objFind) {  **for** (**int** i = front+1 ; i < rear + 1 ; i++) {  **if** (students[i].equals(objFind)) {  **return** **true**;  }  }  **return** **false**;  }  //返回个数  **public** **int** currentCount() {  **return** rear-front;  }  }  //公共类  **public** **class** 队列 {  **public** **static** **void** main(String[] args) {  FIFOQueue test = **new** FIFOQueue();  System.***out***.println(test.currentCount());  }  } |