**每日作业卷**

就业班JavaSE--day04【内部类、static、包、访问修饰符、final】

1. **关卡1**
   1. **训练案例1**
      1. **训练考核知识点**

static关键字

* + 1. **训练描述**

请阐述对static关键字理解

* + 1. **操作步骤描述**

1. 请说出 static 可以修饰什么?

一般修饰成员

1. 请说出 static 关键字的特点 ?

static修饰的成员被多个对象共享。

static修饰的成员属于类，但是会影响每一个对象。

被static修饰的成员又叫类成员，不叫对象的成员。

1. 请说出 static 修饰的成员如何使用 ?

被static修饰的成员可以并且建议通过类名直接访问。也可以通过某个对象访到属于类的静态成员，原因即多个对象均属于一个类，共享使用同一个静态成员

* 1. **训练案例2**
     1. **训练考核知识点**

静态代码块

* + 1. **训练描述**

请阐述你对静态代码块的理解

* + 1. **操作步骤描述**

1. 请说出静态代码块的定义位置.

在类中的成员位置

1. 请说出静态代码块的定义格式.

用{}括起来，用static修饰，只执行一次,随着类的初始化而初始化

1. 请说出静态代码块执行时机和执行次数.

（1）类中的静态块会在整个类加载过程中的初始化阶段执行

（2）在某个工具类中使用static静态代码块，当第一次访问工具类时，就会先进行初始化(只会执行一次)，保存到静态全局属性中，当其他类再次访问时，将直接使用初始化数据。

1. 请说出静态代码块可用于做什么?

一 般情况下,如果有些代码必须在项目启动的时候就执行的时候,需要使用静态代码块,这种代码是主动执行的;需要在项目启动的时候就初始化,在不创建对象的情 况下,其他程序来调用的时候,需要使用静态方法,这种代码是被动执行的. 静态方法在类加载的时候 就已经加载 可以用类名直接调用

* 1. **训练案例3**
     1. **训练考核知识点**

final关键字

* + 1. **训练描述**

请阐述你对final关键字的理解

* + 1. **操作步骤描述**

1.请说出final关键字可以修饰什么?

可以修饰类、成员方法、变量。

2. 请说出被final关键字的特点.

final修饰的类无法被继承。

final修饰的方法无法被重写。

final修饰的变量无法被再次赋值。

* 1. **训练案例4**
     1. **训练考核知识点**

包

* + 1. **训练描述**

请阐述你对包的理解

* + 1. **操作步骤描述**

1. 请说出包的作用

分包管理是组织软件项目结构的基本方式。我们将同类功能放到一个包中，方便管理。并且日常项目的分工也是以包作为边界。

1. 请说出包命名规则

类中定义的包必须与实际class文件所在的文件夹情况相统一，即定义包后，如果类在a包下，则生成的.class文件必须在a文件夹下，否则找不到类。

使用公司域名反写，多级包全部小写，用”.”连接

1. 请说出包声明格式与位置

package 包名;

 package语句必须位于java文件的第一个位置

1. 请说出如何访问包中的类

在Java中提供了四种访问权限，使用不同的访问权限时，被修饰的内容会有不同的访问权限，以下表来说明不同权限的访问能力：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | public | protected | 空的(default) | private |
| 同一类中 | √ | √ | √ | √ |
| 同一包中(子类与无关类) | √ | √ | √ |  |
| 不同包的子类 | √ | √ |  |  |
| 不同包中的无关类 | √ |  |  |  |

1. 请说出导包的作用和导包的格式以及导包语句的位置

作用：（1）解决类名重复产生的冲突问题;（2）便于软件版本的发布

导包的格式：import 包名.类名;

导包语句的位置：java文件下位于包的声明之下，在类名之上。

* 1. **训练案例5**
     1. **训练考核知识点**

四种访问权限修饰符

* + 1. **训练描述**

请阐述你对四种访问权限修饰符的理解

* + 1. **操作步骤描述**

1. 请说出四种权限修饰符分别是什么 ?

Public、protected、defaul、privat

1. 请说出四种权限修饰符可以修饰什么?
2. public可以修饰类，数据成员，构造方法，方法成员。
3. protected可以修饰数据成员，构造方法，方法成员，不能修饰类（此处指外部类，不考虑内部类）。
4. 类，数据成员，构造方法，方法成员，都能够使用默认权限，即不写任何关键字。
5. private可以修饰数据成员，构造方法，方法成员，不能修饰类（此处指外部类，不考虑内部类）。
6. 请说出每种权限修饰符特点 ?
7. 被public修饰的成员，可以在任何一个类中被调用，不管同包或不同包，是权限最大的一个修饰符。
8. 被protected修饰的成员，能在定义它们的类中，同包的类中被调用。
9. 默认权限即同包权限，同包权限的元素只能在定义它们的类中，以及同包的类中被调用。
10. 被private修饰的成员，只能在定义它们的类中使用，在其他类中不能调用。
11. 请说出如何使用每种权限修饰符?

要想仅能在本类中访问使用private修饰；

要想本包中的类都可以访问不加修饰符即可；

要想本类的子类可以访问使用protected修饰

要想任意包中的任意类都可以访问使用public修饰。

* 1. **训练案例6**
     1. **训练考核知识点**

内部类

* + 1. **训练描述**

请阐述内部类的概念

* + 1. **操作步骤描述**

1. 请说出什么是内部类.

内部类即在类中定义另外一个类，是一个相对概念

1. 请说出什么时候使用内部类.

（1）实现事件监听器的时候；

 （2）编写事件驱动时（内部类的对象可以访问外部类的成员方法和变量，注意包括私有成员）；

 （3）在能实现功能的情况下，为了节省编译后产生的字节码（内部类可以减少字节码文件,即java文件编译后的.class文件）；

1. 请说出内部类的分类.

静态嵌套类和非静态嵌套类

* 1. **训练案例7**
     1. **训练考核知识点**

成员内部类

* + 1. **训练描述**

请阐述你对成员内部类的理解

* + 1. **操作步骤描述**

1. 请说出成员内部类定义位置.

定义在成员位置

1. 请说出成员内部类定义格式.

public class Outer{

class Inner{

//其他代码

}

}

1. 请说出如何通过成员内部类创建对象.

在外部类的内部可以直接使用inner s=new inner();（因为外部类知道inner是哪个类，所以可以生成对象。）

而在外部类的外部，要生成（new）一个内部类对象，需要首先建立一个外部类对象（外部类可用），然后在生成一个内部类对象。

Outer o=new Outer();

Outer.Inner in=o.new.Inner()。

* 1. **训练案例8**
     1. **训练考核知识点**

匿名内部类

* + 1. **训练描述**

请阐述你对匿名内部类的理解

* + 1. **操作步骤描述**

1. 请说出匿名内部类的两个含义分别是什么?.

 临时定义某一指定类型的子类

    定义后即刻创建刚刚定义的这个子类的对象

1. 请说出匿名内部类的作用.

可以使命名变得简洁

使代码更加紧凑，简洁,封装性比内部类更优

一个类用于继承其他类或是实现接口，无需增加其他的方法，只是对继承方法实现   覆盖。

* 1. **训练案例9**
     1. **训练考核知识点**

引用数据类型

* + 1. **训练描述**

请阐述你引用数据类型作为参数和返回值类型理解

* + 1. **操作步骤描述**

1. 请说出你对类作为参数和返回值类型理解.

表示可以传递本类对象的引用作为实参

可以传递本类子类的对象的引用

可以传递null，但会报NullPointerException异常

1. 请说出你对抽象类作为参数和返回值类型理解

(1)使用抽象类作为形参意味着可以接受一个他的子类的对象的引用,即将子类的对象的引用作为参数传到方法中

(2)可以传递null，但会报NullPointerException异常

1. 请说出你对接口作为参数和返回值类型理解

(1)使用接口作为形参意味着可以使用这个接口的实现类的对象的引用作为实参传递给方法

(2)可以传递null，但会报NullPointerException异常

## 训练案例10

### 训练考核知识点

静态成员变量

### 训练描述

请说出下面代码执行结果并绘制内存图

**public** **class** Student {

// 学校

**public** **static** String *school* = "清华大学";

// 姓名

**private** String name;

// 年龄

**private** **int** age;

// 构造方法

**public** Student() {

super();

}

**public** Student(String name, **int** age) {

**super**();

**this**.name = name;

**this**.age = age;

}

}

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Student s1 = **new** Student("景甜",22);

Student s2 = **new** Student("林思意",18);

System.***out***.println(s1.*school*);

s1.*school* = " 极客营";

System.***out***.println(s2.*school*);

}

}

### 操作步骤描述

1. 请说出上面代码的运行结果

清华大学

极客营

1. 请画出上面的代码的内存图
   1. 打开画图软件
   2. 在左上角写上:静态成员变量的内存图
   3. 在左边画一个竖长的文本框,把字体调整到10号,把上面的代码拷贝过去
   4. 在代码的右边绘制一个方块表示栈区,右边需要留出绘制堆内存区域,下面需要留出绘制方法区的区域
   5. 在栈内存的方法,绘制一个表示堆内存的方块,在该方块左上角表上堆
   6. 在栈内存和堆内存方块下面绘制一个方块表示方法区,在左上角标上方法区
   7. 在方法区左边绘制一个方块,表示为Test.class代码分配的内存,方法左上角标注Test.class; 方法内部写Test.class的部分代码.
   8. 在栈区最下面画一个方块,左上角标注main方法,表示main方法压栈执行
   9. 在方法区Test.class区域,右边绘制一个方块,左上角标注为Student.class 表示为Student.class代码分配的内存, 方块内部可以写Student类的部分代码
   10. 在方法区Student.class方法的右边绘制一个方块,左上角标注静态区
   11. 在静态区,画一个方块,左上角写上 Student.class 表示Student.class中的静态变量内存
   12. 在Student.class方块内,画一个方块,左上角标注 static school,表示为静态成员school分配的存储空间, 在这个方块内部写 “清华大学” ,表示默认初始化值为清华大学
   13. 在main方块中,上面绘制一个小方块,左上标注为Student s1 表示为s1分配的存储
   14. 在堆内存中,绘制一个方块,右上角标注new Student(“景甜”,22),表示创建对象分配的存储空间,在这个小方块内绘制绘制两个小方块一个左上标注为name,表示为成员变量name分配的存储空间,内部写上”景甜”,一个标注为age,内部写上22,表示为age成员变量分配的存储空间,初始化值为22
   15. 在这个对象的方块,左上角标注上 0X001,表示这个对象的地址;在栈内存中,方块s1中写上0X001,表示变量中存储是对象的地址;画一条从这里到堆内存的0X001的线,然后标注上箭头(两根线),表示s1指向堆内存.
   16. 在main方块中,上面绘制一个小方块,左上标注为Student s2 表示为s2分配的存储
   17. 在堆内存中,绘制一个方块,右上角标注new Student(“林思意”,18),表示创建对象分配的存储空间,在这个小方块内绘制绘制两个小方块一个左上标注为name,表示为成员变量name分配的存储空间,内部写上”林思意”,一个标注为age,内部写上18,表示为age成员变量分配的存储空间,初始化值为18
   18. 在这个对象的方块,左上角标注上 0X002,表示这个对象的地址;在栈内存中,方块s2中写上x003,表示变量中存储是对象的地址
   19. 使用红框把System.***out***.println(s1.*school*);中的s1.school 框起来,画一条从这里到线到静态区school的线条,标注上箭头(使用两根线),表示s1.school访问的是静态区school,所以此处输出的是: 清华大学
   20. 使用红框把s1.*school* = " 极客营"; 中的s1.school框起来,然后一个从这里到静态区school的箭头,然后划掉school静态变量的清华大学,在后面写上 极客营,表示把静态变量school的从原来的清华大学修改为极客营
   21. 使用红框把System.***out***.println(s2.*school*);中的s2.school 框起来,画一条从这里到线到静态区school的线条,标注上箭头(使用两根线),表示s2.school访问的是静态区school,所以此处输出的是: 极客营

## 训练案例11

### 训练考核知识点

静态代码块

### 训练描述

看下面的代码说出运行结果以及执行流程

**public** **class** Test {

**static** {

System.***out***.println("Test 静态代码块");

}

**public** Test(){

System.***out***.println("Test 构造方法");

}

**public** **static** **void** main(String[] args) {

System.***out***.println("Test main方法");

Test t = **new** Test();

}

}

### 操作步骤描述

1. 请说上面代码运行结果

Test 静态代码块

Test main方法

Test 构造方法

1. 请说出上面代码的执行流程

+ 静态代码块 main方法 构造方法

## 训练案例12

### 训练考核知识点

成员内部类

### 训练描述

请用代码描述:

汽车有两种状态,一种是停止,一种运行; 汽车的内部都一个发动机;发动机有一个工作的功能; 如果汽车的运行状态发动机就飞速旋转,如果汽车是停止状态,发动机停止工作

要求: 使用成员内部类.

### 操作步骤描述

1. 定义一个汽车类(C
2. ar)
   1. 成员变量: 状态(status),true表示运行状态一种,false表示停止状态.
   2. 成员内部类: 发动机(Engine)
      1. 成员方法:work()
         1. 如果汽车的运行状态打印发动机就飞速旋转
         2. 否则打印发动机停止工作
   3. 提供:状态(status)的getter和setter方法
3. 定义测试类(Test)
   1. 提供main方法
   2. 在main方法中
      1. 创建汽车对象 c
      2. 通过汽车对象c创建该汽车的发动机对象e
      3. 调用发动机e对象的work()方法
      4. 把汽车对象c的状态设置为true
      5. 调用发动机e对象的work()方法

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Car car = **new** Car();

Engine e= car.**new** Engine();

e.work();

car.setStatus(**true**);

e.work();

}

}

Car类：

**public** **class** Car {

**private** **boolean** status;

**public** **boolean** isStatus() {

**return** status;

}

**public** **void** setStatus(**boolean** status) {

**this**.status = status;

}

**public** **class** Engine{

**public** **void** work(){

**if** (status==**true**) {

System.***out***.println("发动机飞速旋转");

}**else**{

System.***out***.println("发动机停止工作");

}

}

}

}

# 关卡2

## 训练案例1

### 训练考核知识点

权限修饰符

static

final

abstract

### 训练描述

请简述abstract不能和那些关键一起使用,为什么.

### 操作步骤描述

1. abstract可以与private关键字一起使用吗? 为什么? 不可以，private子类看不到
2. abstract 可以与static关键字一起使用吗? 为什么? 不可以，static需要创建对象，abstract不需要创建对象
3. abstract 可以与final关键字一起使用吗? 为什么? 不可以，因为final修饰的方法不能被重写

## 训练案例2

### 训练考核知识点

静态代码块

### 训练描述

看下面的程序,说出的它的执行结果,以及执行流程

**public** **class** Student {

**static** {

System.***out***.println("Student 类的静态代码块");

}

**public** Student(){

System.***out***.println("Student 类的构造方法");

}

**public** Student(String methodName){

System.***out***.println("在"+methodName+"方法中创建Student对象调用的");

}

}

**public** **class** Test {

**static** {

System.***out***.println("Test 类的静态代码块");

}

**public** **static** **void** main(String[] args) {

System.***out***.println("执行main方法");

// 创建学生对象

Student s1 = **new** Student("main");

}

}

提示: 类只有被使用到的时候才会加载

### 操作步骤描述

1. 说出输出结果

Test 类的静态代码块

执行main方法

Student 类的静态代码块

在main方法中创建Student对象调用的

1. 阐述类的加载与代码的执行流程

主类的静态代码块，主类的main方法，Student类的静态代码块，Student的带参构造

## 训练案例3

### 训练考核知识点

final关键字

### 训练描述

请写代码验证final关键字的特点

### 操作步骤描述

1. 验证:final修饰类不可以被继承，但是可以继承其他类
   1. 定义类(Final\_1),用于验证第一个特点
   2. 在类上面使用注释写上: 验证:final修饰类不可以被继承，但是可以继承其他类
   3. 在Final\_1.java中,定义被修final修饰的类(FinalClass) 类
   4. 在Final\_1.java中,定义子类Sub继承FinalClass类,编译报错,把鼠标放在红色X上面,提示:The type Sub cannot subclass the final class FinalClass:
   5. 把类Sub,使用注释掉,然后在下面写上: 被final修饰的类不能被继承
   6. 让FinalClass继承Final\_1,编译通过; 在FinalClass上面使用注释写上: 验证:被final的修饰类可以继承其他类.

/\*

\*验证:final修饰类不可以被继承，但是可以继承其他类

\*/

**public** **class** Final\_1 {

}

/\*

\* 验证:被final的修饰类可以继承其他类.

\* \*/

**public** **final** **class** FinalClass **extends** Final\_1{

}

/\*public class Sub extends FinalClass{

}

被final修饰的类不能被继承

\*/

1. final修饰的方法不可以被覆盖,但父类中没有被final修饰方法，子类覆盖后可以加final
   1. 定义类(Final\_2),用于验证第二特点
      1. 成员方法:
         1. 被final修饰的成员方法 finalMethod
         2. 普通方法 normalMethod
   2. 在Final\_2.java中,定义子类(Sub2) 继承 Final\_2
      1. 覆盖 finalMethod编译报错,把鼠标放到红色x上面,提示:Cannot override the final method from Final\_2: 被final修饰方法不能被覆盖(重写);
      2. 注释调用这个方法,在上面使用注释写上: final修饰方法不能被覆盖(重写);
      3. 覆盖normalMethod 方法,在权限修饰符 和返回值直接加上final,编译通过,在这个方法上面写上: 父类中没有被final修饰方法，子类覆盖后可以加final

Finl\_2：

**public** **class** Final\_2 {

**public** **final** **void** finalMethod(){}

**public** **void** normalMethod(){}

}

Sub2类：

**public** **class** Sub2 **extends** Final\_2{

//final修饰方法不能被覆盖(重写);

//public void finalMethod(){}

//父类中没有被final修饰方法，子类覆盖后可以加final

**public** **final** **void** normalMethod (){}

}

1. final修饰的变量称为常量，这些变量只能赋值一次
   1. 定义Final\_3类,表示用于验证final的第三个特点
   2. 在这个类上面使用注释写上: 验证: final修饰的变量称为常量，这些变量只能赋值一次
   3. 提供main方法
   4. 在main方法中
      1. 定义被final修饰的整型变量a,初始化值为10
      2. 给a赋值为20,编译报错,
      3. 把鼠标放在红色x上面提示:The final local variable a cannot be assigned:被fianl修饰的局部变量不能被赋值;
      4. 注释掉这句话,在上面写上:final修饰的变量称为常量，这些变量只能赋值一次
      5. 定义被final修饰的整型变量b,没有初始化值
      6. 下一行,给变量b 赋值为5
      7. 然后再给变量b赋值为15,编译报错
      8. 把鼠标放到红色x上面提示:The final local variable b may already have been assigned: 被final修饰的局部变量已经赋值了
      9. 注释掉这个赋值语句,然后在上面使用注释写上:final修饰的变量称为常量，这些变量只能赋值一次

// 验证: final修饰的变量称为常量，这些变量只能赋值一次

**public** **class** Final\_3 {

**public** **static** **void** main(String[] args) {

**final** **int** a = 10;

// final修饰的变量称为常量，这些变量只能赋值一次

// a = 20;

**final** **int** b;

b = 5;

//final修饰的变量称为常量，这些变量只能赋值一次

//b = 15;

}

}

1. 引用类型的变量值为对象地址值，地址值不能更改，但是地址内的对象属性值可以修改
   1. 定义类(Final\_4),表示用于验证final的第四个特点,在这个类上面使用注释写上:验证:引用类型的变量值为对象地址值，地址值不能更改，但是地址内的对象属性值可以修改
   2. 在Fianl4.java中,定义辅助类,随便写,我这里使用猫类(Cat)
      1. 定义公共权限成员变量: 年龄 (age)
   3. 在Fianl\_4类中提供main方法
   4. 在main方法中
      1. 定义被final修饰的Cat类型的引用变量c,new Cat()赋值给它
      2. 然后,在new Cat() 赋值给变量c,编译报错
      3. 错误提示:The final local variable c cannot be assigned. 被fianl修饰的局部变量不能被赋值
      4. 注释掉这个赋值语句,然后在上面使用注释写上: 引用类型的变量值为对象地址值，地址值不能更改
   5. 给对象c的age属性赋值为2,编译通过; 表示: 其指向的对象属性值可以修改
   6. 在这话上面,使用注释写上:地址内的对象属性值可以修改

//验证:引用类型的变量值为对象地址值，地址值不能更改，但是地址内的对象属性值可以修改

**public** **class** Final\_4 {

**public** **static** **void** main(String[] args) {

**final** Cat c = **new** Cat();

//引用类型的变量值为对象地址值，地址值不能更改

//c = new Cat();

//地址内的对象属性值可以修改

c.age = 2;

}

}

1. 修饰成员变量，需要在创建对象前赋值，否则报错。(当没有显式赋值时，多个构造方法的均需要为其赋值。
   1. 定义类(Final\_5),表示用于验证final的第五个特点
   2. 在这个类上面使用注释写上:验证:修饰成员变量，需要在创建对象前赋值，否则报错。(当没有显式赋值时，多个构造方法的均需要为其赋值。
   3. 这里需要验证两点:
      1. 定义final成员变量的时候,显示赋值
      2. 否则每一个构造方法都需要给这个final修饰成员变量赋值
   4. 定义被final修饰的整型成员变量a,并显式赋值为10
   5. 定义被final修饰的整型成 员变量b,不显式赋值,编译报错,
      1. 提示The blank final field b may not have been initialized,被final修饰的成员变量可能没有被赋值
   6. 提供空参构造方法,上面成员变量b不报错了,但是这里开始报错了
      1. 提示:The blank final field b may not have been initialized
      2. 在空参构造方法内部写上: b = 20; 编译通过
   7. 提供满参构造,但是不给a和b赋值,报错
      1. 提示:The blank final field b may not have been initialized
      2. 在满参构造方法中写上this.b = b; 编译通过,说明: 当没有显式赋值时，多个构造方法的均需要为其赋值.
      3. 在写上 this.a = a; 编译报错,提示:The final field Final\_5.a cannot be assigned 表示被final修饰成员变量只能赋值一次
      4. 注释掉这句代码,在后面写上,被final修饰成员变量只能赋值一次

//验证:修饰成员变量，需要在创建对象前赋值，否则报错。

**public** **class** Final\_5 {

**public** **final** **int** a = 20;

**public** **final** **int** b;

**public** Final\_5() {

b=20;

}

//当没有显式赋值时，多个构造方法的均需要为其赋值.

**public** Final\_5(**int** b) {

**this**.b = b;

//this.b = a; 被final修饰成员变量只能赋值一次

}

}

## 训练案例4

### 训练考核知识点

抽象类

匿名内部类

### 训练描述

请用代码描述:

在某新型公司,每一个员工都有工号,姓名; 都要工作,但是每一个员工的工作内容都不一样;

要求:使用匿名内部类创建对象

### 操作步骤描述

1. 定义抽象类(Employee)
   1. 属性: 工号(id),姓名(name)
   2. 行为: 抽象方法work()
   3. 要求: 提供setters和gettters方法
2. 定义测试类Test
   1. 提供main方法
   2. 在main方法中
      1. 使用匿名内部类创建Employee的子类对象 e;并重写work()方法
         1. 输出格式:工号为001的员工范冰冰在努力的敲代码
      2. 使用setXxx方法给e的工号和名称赋值
      3. 调用e的work()方法

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Employee e = **new** Employee(){

**public** **void** work(){

System.***out***.println("工号为"+**this**.getId()+"的员工"+**this**.getName()+"在努力的敲代码");

}

};

e.setId("001");

e.setName("范冰冰");

e.work();

}

}

## 训练案例5

### 训练考核知识点

接口

接口名作为形式参数类型

匿名内部类

### 训练描述

请用代码描述：

某市举办一场运动会,只要是会跑的都可以参加.

要求: 使用到接口,接口作为参数,匿名内部类.

提示: 把跑功能抽取到接口中 ,在测试类中提供参加运动会的静态方法

### 操作步骤描述

1. 定义接口(Sport)
   1. 抽象方法: 跑(run)
2. 定义测试类Test
   1. 定义静态方法:进入运行会(enter(Sport s)),在方法中调用s的run方法
   2. 提供main方法
   3. 在main方法中
      1. 调用enter(Sport s),传入使用Sport接口的匿名内部类创建的对象;
      2. 在匿名内部类中重写run方法
         1. 输出: 参加运动会,奔跑吧

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

*enter*(**new** Sport() {

@Override

**public** **void** run() {

System.***out***.println("参加运动会,奔跑吧");

}

});

}

**public** **static** **void** enter(Sport s){

s.run();

}

}

Sport接口:

**public** **interface** Sport {

**public** **void** run();

}

## 训练案例6

### 训练考核知识点

接口

接口名作为形式参数类型

匿名内部类

### 训练描述

请用代码描述：

某高校举成立一个大球俱乐部,只要是会打篮球,踢足球,打排球的可以参加.

要求: 使用到接口,接口作为参数,匿名内部类.

提示: 把打篮球,踢足球,打排球功能抽取到接口中 ,在测试类中提供加入俱乐部的静态方法,参数为接口类;

### 操作步骤描述

1. 定义接口(Player)
   1. 抽象方法: 打篮球(playBasketball),踢足球(playFootball),打排球(playVolleyball)
2. 定义测试类Test
   1. 定义静态方法:进入运行会(join(Player p)),在方法中调用p的 打篮球,踢足球,打排球 方法
   2. 提供main方法
   3. 在main方法中
      1. 调用join(Player p),传入使用Player 接口的匿名内部类创建的匿名对象;
      2. 在匿名内部类中
         1. 重写playBasketball()方法
            1. 输出: 在打篮球
         2. .重写playFootball()方法
            1. 输出: 在踢足球
         3. .重写playVolleyball()方法
            1. 输出: 在打排球

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

*join*(**new** Player() {

@Override

**public** **void** playVolleyball() {

System.***out***.println("在打篮球");

}

@Override

**public** **void** playFootball() {

System.***out***.println("在踢足球");

}

@Override

**public** **void** playBasketball() {

System.***out***.println("在打排球");

}

});

}

**public** **static** **void** join(Player p){

p.playBasketball();

p.playFootball();

p.playVolleyball();

}

}

Player接口：

**public** **interface** Player {

**public** **abstract** **void** playBasketball();

**public** **abstract** **void** playFootball();

**public** **abstract** **void** playVolleyball();

}

# 关卡3

## 训练案例1

### 训练考核知识点

静态代码块

构造方法

成员变量

### 训练描述

看下面的程序说出执行结果并阐述程序的执行流程

**public** **class** Student {

**static** {

System.***out***.println("Student 类的静态代码块");

}

**public** Student(){

System.***out***.println("Student 类的构造方法");

}

**public** Student(String methodName){

System.***out***.println("在"+methodName+"创建Student对象调用的");

}

}

**public** **class** Test {

// 实例(对象)成员变量

Student s = **new** Student("成员变量显式初始化的时候");

// 静态代码块

**static** {

System.***out***.println("Test 类的静态代码块");

}

**public** **static** **void** main(String[] args) {

System.***out***.println("main方法");

// 创建学生对象

Student s1 = **new** Student("main方法中");

// 创建Test对象

Test t = **new** Test();

}

}

提示: 1. 类只有被使用就会加载到内存中

2.只有创建对象才会为成员变量分配存储空间才能对成员变量赋值

Test 类的静态代码块

main方法

Student 类的静态代码块

在main方法中创建Student对象调用的

在成员变量显式初始化的时候创建Student对象调用的

## 训练案例2

### 训练考核知识点

抽象类

匿名内部类

### 训练描述

请使用代码描述:

某人养了几只宠物,每个宠物都有名字,每一个宠物都会吃,但是每个宠物吃的东西都不一样;宠物1叫tom吃鱼,宠物2叫小黑,啃骨头.

要求: 用到抽象类和匿名内部类

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

Pet p1 = **new** Pet() {

@Override

**public** **void** eat() {

System.***out***.println(getName()+"吃鱼");

}

};

p1.setName("Tom");

p1.eat();

Pet p2 = **new** Pet() {

@Override

**public** **void** eat() {

System.***out***.println(getName()+"啃骨头");

}

};

p2.setName("小黑");

p2.eat();

}

}

Pet类：

**public** **abstract** **class** Pet {

**private** String name;

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **abstract** **void** eat();

}



## 训练案例3

### 训练考核知识点

接口

接口作为形式参数

匿名内部类

### 训练描述

请使用代码描述

某公司成立一个游泳俱乐部,只要是会游泳的都可以加入;

要求: 使用到接口,接口作为参数,匿名内部类.

提示: 把游泳功能抽取到接口中 ,在测试类中提供参加运动会的静态方法

测试类：

**public** **class** Test {

**public** **static** **void** main(String[] args) {

*join*(**new** Swim() {

@Override

**public** **void** swim() {

System.***out***.println("正在参加运动会游泳");

}

});

}

**public** **static** **void** join(Swim s){

s.swim();

}

}

Swim接口:

**public** **interface** Swim {

**public** **void** swim();

}

