JAVA 第一阶段—DAY11-JAVA作业答案

1. 简答题：简述什么多态，以及其优点。

**答案：**

多态：同一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果，这就是多态性。简单的说:就是用基类的引用指向子类的对象。

优点：

- 大大提高程序的可复用性。

- 可以提高可扩充性和可维护性。

1. 简答题：简述Java引用数据类型转换可能出现的问题，以及如何避免。

**答案：**

可能出现ClassCastException 类型转换异常。

为了避免ClassCastException的发生，Java提供了 instanceof 关键字，给引用变量做类型的校验。

1. 简答题：简述Java内部类的分类。

**答案：**

1. **成员内部类**

在一个类的内部方法为，与外部类的成员变量和成员方法同一级别。

1. **静态内部类**

就是使用static关键字修饰的内部类。

1. **局部内部类**

在类中的局部变量同级别位置定义的类。

1. **匿名内部类**

是内部类的简化写法，它的本质是一个 带具体实现的 父类或者父接口的 匿名的 子类对象。

1. 简答题：使用模板设计模式模拟完成动物类下的猫和狗的日常活动。

**答案：**

// 抽象类

public abstract class Animal {

public final void lifeTrack(){

sleep();

work();

eat();

}

private void sleep(){

System.out.println("睡觉");

}

private void eat(){

System.out.println("吃东西");

}

protected abstract void work();

}

// 子类

public class Cat extends Animal {

@Override

protected void work() {

System.out.println("猫抓老鼠");

}

}

// 测试类

public class Test1 {

public static void main(String[] args) {

Dog dog = new Dog();

dog.lifeTrack();

}

}

1. 编程题：阅读以下需求，请用代码完成程序。需求如下。

**答案：**

/\*

1.定义一个汽车类(Car)

a)成员变量: 状态(status),true表示运行状态一种,false表示停止状态.

b)成员内部类: 发动机(Engine)

i.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

c)提供:状态(status)的getter和setter方法

\*/

public class Car {

// a)成员变量: 状态(status),true表示运行状态一种,false表示停止状态.

private boolean status;

// b)成员内部类: 发动机(Engine)

public class Engine{

// i.成员方法:work()

public void work(){

// 1.如果汽车的运行状态打印发动机就飞速旋转

if(status){

System.out.println("发动机就飞速旋转");

}else{

// 2.否则打印发动机停止工作

System.out.println("发动机停止工作");

}

}

}

// 状态(status)的getter和setter方法

public boolean isStatus() {

return status;

}

public void setStatus(boolean status) {

this.status = status;

}

/\*

2.定义测试类(Test)

a)提供main方法

b)在main方法中

i.创建汽车对象 c

ii.通过汽车对象c创建该汽车的发动机对象e

iii.调用发动机e对象的work()方法

iv.把汽车对象c的状态设置为true

v.调用发动机e对象的work()方法

\*/

public class Test {

public static void main(String[] args) {

// i.创建汽车对象 c

Car c = new Car();

// ii.通过汽车对象c创建该汽车的发动机对象e

Car.Engine e = c.new Engine();

// iii.调用发动机e对象的work()方法

e.work();

// iv.把汽车对象c的状态设置为true

c.setStatus(true);

// v.调用发动机e对象的work()方法

e.work();

}

}

}

1. 编程题：阅读以下需求，请用代码完成程序。需求如下。

/\*

1.定义游泳者接口(Swimmer)

a)抽象方法: 游泳swim

\*/

public interface Swimmer {

//a) 抽象方法: 游泳swim

public abstract void swim();

}

/\*

2.定义测试类Test

a)定义加入游泳俱乐部的方法: join(Swimmer s)

i.在方法中调用s的swim方法

b)提供main方法

c)在main方法中

i.调用join(Swimmer s) 传入使用接口Swimmer的匿名内部类创建的对象

ii.在匿名内部类中重写swim方法: 输出:愉快的游泳

\*/

public class Test {

// a)定义加入游泳俱乐部的方法: join(Swimmer s)

public static void join(Swimmer s){

// i.在方法中调用s的swim方法

s.swim();

}

public static void main(String[] args) {

// i.调用join(Swimmer s) 传入使用接口Swimmer的匿名内部类创建的对象

join(new Swimmer() {

// ii.在匿名内部类中重写swim方法: 输出:愉快的游泳

public void swim() {

System.out.println("愉快的游泳");

}

});

}

}

1. 编程题：按照要求，补齐代码。

// 需求1

interface Inter {

void show();

}

class Outer {

//补齐代码

public static Inter method() {

return new Inter() {

@Override

public void show() {

System.out.println("HelloWorld");

}

};

}

}

public class Test {

public static void main(String[] args) {

Outer.method().show();

}

}

// 需求2

interface Inter {

void show();

}

class Outer {

//补齐代码

public static Inter method = new Inter() {

@Override

public void show() {

System.out.println("HelloWorld");

}

};

}

public class Test {

public static void main(String[] args) {

// Outer.method().show();

Outer.method.show();

}

}