### JAVA第一阶段—DAY09-JAVA 作业答案

1. 简答题：简述什么封装，以及其优点。

**答案：**

封装：就是将内部细节隐藏起来，若需要访问某个成员，提供公共方法对其访问或者是只允许内部访问。

优点：

- 良好的封装能够减少耦合。

- 类内部的结构可以自由修改。

- 可以对类成员进行更精确的控制。

- 隐藏信息，实现细节。

1. 简答题：简述Java权限修饰符有哪些以及对应的访问范围。

**答案：**

public：公共的。范围最大，同类、同包、不同包子类、不同包无关类都可以访问。

protected:收保护的。同类、同包、不同包子类所继承的属性和方法可以访问。

default：默认的。同类、同包可以访问。

private：私有的。访问权限最小，只能在同类可以访问。

1. 简答题：简述Java继承后的各个成员的特点。

**答案：**

1. **成员变量**

如果子类父类中出现不重名的成员变量，这时的访问是没有影响的；

如果子类父类中出现重名的成员变量，这时的访问是有影响的。

1. **成员方法**

如果子类父类中出现不重名的成员方法，这时的调用是没有影响的；

如果子类父类中出现重名的成员方法，可分为方法重载和方法重写两种情况。其中，如果子类中出现与父类一模一样的方法时（返回值类型，方法名和参数列表都相同），会出现覆盖效果，也称为重写或者复写。

1. **构造方法**

构造方法的名字是与类名一致的。所以子类是无法继承父类构造方法的。

子类的初始化过程中，必须先执行父类的初始化动作。子类的构造方法中默认有一个 super() ，表示调用父类的构造方法，父类成员变量初始化后，才可以给子类使用。

1. 简答题：简述Java中的final关键字及用法。

**答案：**

- final修饰类：表示最终类，也就是此类不能被继承；

- final修饰变量：表示常量，也就是一个变量的值始终如一，无法变更；

- final修饰方法：表示最终方法，也就是此方法不能被重写。

1. 简答题：以教师类为例，定义一个标准的JavaBean类。

**答案：**

public class Teacher {

private String name;

private int age;

public Teacher() {

}

public Teacher(String name, int age) {

this.name = name;

this.age = age;

}

public void teach() {

System.out.println(name+"老师教书");

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

}

1. 编程题：编程题：某公司要开发新游戏，请以面向对象的思想进行程序设计设计游戏中的蛇怪和蜈蚣精。

**答案：**

//妖怪类

public class Monster {

private String name;//怪物名

private int hp;//血量

private int aggressivity;//攻击力

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getHp() {

return hp;

}

public void setHp(int hp) {

this.hp = hp;

}

public int getAggressivity() {

return aggressivity;

}

public void setAggressivity(int aggressivity) {

this.aggressivity = aggressivity;

}

/\*\* 攻击 \*/

public void attack() {

System.out.println("怪物"+this.getName()+"展开攻击");

System.out.println("当前生命值："+this.getHp());

System.out.println("攻击力是："+this.getAggressivity());

}

/\*\* 移动 \*/

public void move() {

}

}

/\*\* 怪蛇类 \*/

public class Snake extends Monster {

/\*\* 加血 \*/

public void addHp(){

this.setHp(this.getHp()+20);

System.out.println("实施大蛇补血术......，当前的生命值是"+this.getHp());

}

/\*\* 重写父类的移动方法\*/

public void move() {

System.out.println("我是蛇，我走S线路");

}

}

/\*\* 蜈蚣类 \*/

public class Centipede extends Monster {

/\*\* 重写父类的移动方法\*/

public void move() {

System.out.println("我是蜈蚣精，我御风飞行");

}

}

// 测试类

public class Test {

public static void main(String[] args) {

Snake sn = new Snake();//生产一个蛇精

sn.setName("怪蛇甲");

sn.setHp(5);

sn.setAggressivity(20);

sn.attack();//调用攻击方法

if(sn.getHp()<10){//当生命少于10时，加20血

sn.addHp();

}

sn.move();//蛇移动

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

Centipede ct = new Centipede();//new一个蜈蚣

ct.setName("蜈蚣精乙");

ct.setHp(60);

ct.setAggressivity(15);

ct.attack();//蜈蚣攻击

ct.move();

}

}