# 继承、多态-作业

## 1.封装

**需求分析：**

描述什么是封装？

## 2.构造器

**需求分析：**

一个类中，是否总会存在默认的无参构造器？

## 3.封装考核

**需求分析：**

编写代码，实现封装一个学生的基本信息，实现对姓名、年龄、性别、学号属性的封装，确保属性不被外部直接访问，同时提供合适的方法访问和修改这些属性

测试代码：

```java
public class Test03 {
    public static void main(String[] args) {
        Student s = new Student("张三", 19, "男", "10000");
        System.out.println(s.getName()); //输出张三
        System.out.println(s.getAge()); //输出19
        s.setName("李四");
        System.out.println(s.getName()); //输出李四
    }
}
```

# 4.面向对象考核

**需求分析:**

请根据面向对象的思想使用代码实现张三开车去公司这件事。实体信息为人和交通工具

# 5.构造器考核

**需求分析:**

根据要求编写Duration(时长)类，编写完成后使用下面的测试类去测试，要求如下

`Duration`类:

1. 定义3个整型属性：hours、 minutes、seconds
2. 定义三参构造器，实现对hours、 minutes、seconds进行初始化
3. 定义一参构造器，实现对hours、 minutes、seconds进行初始化，参数为总的seconds，例如将x秒，转为a小时b分钟c秒，3661秒，就是1小时1分钟1秒
4. 定义每个属性对应的get方法，实现获取属性值

5. 定义`public int getTotalSeconds(){}`方法，实现返回总的秒数值
6. 定义`public String toString() {}`方法，方法返回内容为时分秒信息，格式为

测试类：

```
public class Test05_duration {
    public static void main(String[] args) {
        Duration d = new Duration(1, 1, 1);
        int totalSeconds = d.getTotalSeconds();
        System.out.println("totalSeconds = " + totalSeconds);
        //输出 totalSeconds = 3661

        Duration d2 = new Duration(3660);
        int hours = d2.getHours();
        System.out.println("hours = " + hours);//输出 hours = 1
        System.out.println(d2);// 输出 1:1:0
    }
}
```

# 6.程序分析

**需求分析：**

分析程序是否错误，如果错误，说明错误原因。如果正确，写出输出结果

```
//汽车类
class Car{
    public Car(){
        System.out.println("car");
    }
}
//奥迪类
class Audi extends Car{
    public Audi(){
```

```
10          System.out.println("audi");
11      }
12  }
13  public class Test06 {
14      public static void main(String[] args){
15          Audi a = new Audi();
16          Car c = new Car();
17      }
18  }
```

# 7.一个类最多可以继承多少个类?

# 8.封装和继承有什么区别?

# 9.继承考核

**需求分析:**

编写程序，实现汽车租赁公司汽车出租方案:

1. 所有车辆（Automobile）都具有品牌（brand）和车牌号（plateNumber）信息和可以计算租金（getRent）的功能
2. 所有车主要分为卡车（Truck）和巴士（Bus）2种类型
3. 卡车租金方案:

| 车辆类型 | 每天租金（元） |
| --- | --- |
| 小型 | 300 |
| 中型 | 350 |
| 大型 | 500 |

4. 巴士租金方案：

| 座位数 | 每天租金 |
| --- | --- |
| <=16 | 400 |
| >16 | 600 |

测试类：

```
public class Test09_automobile {
    public static void main(String[] args) {
        Truck t = new Truck("北汽", "苏U12345", "中型");
        int rent = t.getRent(3);
        System.out.println(t.getBrand() + "\t" +
t.getPlateNumber() + " 租金为：" + rent);
        //输出 北汽 苏U12345 租金为：1050

        Automobile b = new Bus("宇通", "苏A11111", 30);
        int busRent = b.getRent(10);
        System.out.println(b.getBrand() + "\t" +
b.getPlateNumber() + " 租金为：" + busRent);
        //输出 宇通 苏A11111 租金为：6000
    }
}
```

# 10.多态考核

**需求分析：**

编写代码表示多态概念，并说明什么是多态?

测试类：

```
public class Test10_ball {
    public static void playBall(Ball ball){
        ball.play();
    }
    public static void main(String[] args) {
        Ball b = new Ball();
        Ball b2 = new Basketball();
        Ball b3 = new Football();
        Test10_ball.playBall(b);
        Test10_ball.playBall(b2);
        Test10_ball.playBall(b3);
    }
}
```

# 11.重载与重写

**需求分析：**

描述方法重载和方法重写的语法要求

# 12.程序分析

**需求分析：**

分析程序是否错误，如果错误，说明错误原因。如果正确，写出输出结果

```
class Super{
    protected int test(){
        return 1;
```

```
    }
}

class Sub extends Super{
    public long test(){
        return 0L;
    }
}
public class Test12 {
    public static void main(String[] args) {
        Super s = new Sub();
        System.out.println(s.test());
    }
}
```

# 13.程序设计

**需求分析:**

假设要为某个公司编写雇员工资支付程序:

1. 工人（Worker）按每月工作的天数计算工资
2. 销售人员（Salesman）在基本工资基础上每月还有销售提成
3. 经理（Manager） 每月按固定工资支付
4. 临时工（Floater ）按每小时50元支付
5. 所有员工都有共同特性（如姓名，性别，出生日期，员工类别）

测试类:

```
public class Test13_employee {
    public static void main(String[] args) {
        //21表示工作天数
        Employee worker = new Worker("张三", "男",
"2001.01.01", 21);
```

```
        //3000 表示基本工资 2000表示销售提成
        Employee salesman = new Salesman("赵六", "男",
"2000.03.07", 3000, 2000);
        //6000 表示基本工资
        Employee manager = new Manager("李四", "女",
"2003.02.09", 6000);
        // 168表示工作小时
        Employee floater = new Floater("王五", "女",
"2002.10.23", 168);
        worker.computeSalary();//输出：工人 张三 本月工资为
4200.0 元。
        salesman.computeSalary();//输出：销售员 赵六 本月工资为
5000.0 元。
        manager.computeSalary();//输出：经理 李四 本月工资为
6000.0 元。
        floater.computeSalary();//输出：临时工 王五 本月工资为
8400.0 元。
    }
}
```

# 14.图形类

1. 定义一个圆类 Circle
   属性：半径radius
   功能：1.计算表面积、2.计算周长

   重写功能：1.toString方法会输出对象信息

2. 定义圆的子类：圆柱体 Cylinder
   属性：高 height
   重写功能：1.计算表面积、2.toString方法会输出对象信息

   新增功能：计算体积

当Circle类型的引用指向Cylinder类型的对象时，能否调用到它的计算体积的方法？如果能，如何编写代码？

# 15.程序分析

**需求分析:**

分析程序是否错误，如果错误，说明错误原因。如果正确，写出输出结果

```
class Father {
    void show() {
        System.out.println("A");
    }
}

class Son extends Father {
    void show() {
        super.show();
        System.out.println("B");
    }
}
public class Test15 {
    public static void main(String[] args) {
        Father f = new Son();
        f.show();
    }
}
```

# 16.程序分析

**需求分析:**

分析程序是否错误，如果错误，说明错误原因。如果正确，写出输出结果

```
class A {
    int x = 10;
    void show() {
        System.out.println("A: " + x);
```

```
    }
}
class B extends A {
    int x = 20;
    void show() {
        System.out.println("B: " + x);
    }
}
public class Test16 {
    public static void main(String[] args) {
        A a = new B();
        a.show();
        System.out.println("A: " + a.x);
        System.out.println("B: " + ((B)a).x);
    }
}
```

# 17.程序分析

**需求分析:**

分析程序是否错误，如果错误，说明错误原因。如果正确，写出输出结果

```
class Animal {
    public void speak() {
        System.out.println("I am an animal.");
    }
}

class Dog extends Animal {
    @Override
    public void speak() {
        System.out.println("I am a dog.");
    }

```

```
13      public void wagTail() {
14          System.out.println("I am wagging my tail.");
15      }
16  }
17
18  public class Test17 {
19      public static void main(String[] args) {
20          Animal a = new Dog();
21          a.speak();
22          a.wagTail();
23      }
24  }
```