继承、多态-答案

1.封装

需求分析：

描述什么是封装？

答案：

封装是一种将数据和行为包装在类中的机制，以保护数据的安全性和可靠性。它是一种面向对象编程的基本概念，通过隐藏类的实现细节和访问限制，使得代码更易于维护和理解。

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？

答案：

如果用户没有显示声明一个构造器那么就会存在一个默认的无参构造器，但是当用户自己编写任意一个构造器之后则不会提供默认无参构造器。

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：

```
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
    }
}
```

答案：

```
class Student {
    private String id;
    private String name;
    private int age;
    private String gender;

    public Student(String name, int age, String gender, String id) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.gender = gender;
    }

    public String getId() {
        return id;
    }

```

```
18      public void setId(String id) {
19          this.id = id;
20      }
21
22      public String getName() {
23          return name;
24      }
25
26      public void setName(String name) {
27          this.name = name;
28      }
29
30      public int getAge() {
31          return age;
32      }
33
34      public void setAge(int age) {
35          this.age = age;
36      }
37
38      public String getGender() {
39          return gender;
40      }
41
42      public void setGender(String gender) {
43          this.gender = gender;
44      }
45  }
```

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

答案:

```
//人
class Person {
    private String name;

    public Person(String name) {
        this.name = name;
    }
    //开车
    public void drive(Vehicle vehicle, String destination) {
        System.out.printf("%s开%s去%s", this.name, vehicle,
destination);
    }
}
//交通工具
class Vehicle {
    private String type;

    public Vehicle(String type) {
        this.type = type;
    }

    @Override
    public String toString() {
        return this.type;
    }
}
public class Test04 {
    public static void main(String[] args) {
        Person p = new Person("张三");
        Vehicle v = new Vehicle("汽车");
        p.drive(v, "公司");
```

```
31          }
32      }
```

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：

```java
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
    }
}
```

答案:

```java
//时长类
class Duration {
    private int hours;
    private int minutes;
    private int seconds;

    public Duration(int hours, int minutes, int seconds) {
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
    }

    public Duration(int totalSeconds) {
        this.hours = totalSeconds / 3600;
        this.minutes = totalSeconds % 3600 / 60;
        this.seconds = totalSeconds % 60;
    }

    public int getHours() {
        return hours;
```

```
    }

    public int getMinutes() {
        return minutes;
    }

    public int getSeconds() {
        return seconds;
    }

    public int getTotalSeconds() {
        return hours * 3600 + minutes * 60 + seconds;
    }

    public String toString() {
        return hours + ":" + minutes + ":" + seconds;
    }
}
```

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{
```

```
9       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  }
```

答案:

```
1  car
2  audi
3  car
```

7.一个类最多可以继承多少个类?

答案:

Java中一个类最多只能继承一个类，这也称为单继承。

8.封装和继承有什么区别?

答案:

封装和继承都是面向对象编程中的基本概念。封装强调数据和行为的组合，使得类的实现细节对外隐藏；而继承强调代码的重用，可以从现有类派生出新类，并继承原有类的属性和方法。封装和继承可以协同工作，使得代码更加模块化和易于维护。

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：

```java
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
    }
}
```

答案：

```java
//汽车类
class Automobile {
    //品牌
    private String brand;
    //车牌号
    private String plateNumber;

    public Automobile(String brand, String plateNumber) {
        this.brand = brand;
        this.plateNumber = plateNumber;
    }

    public int getRent(int days) {
        return 0;
    }

    public String getBrand() {
        return brand;
```

```
19      }
20
21      public String getPlateNumber() {
22          return plateNumber;
23      }
24  }
25  //卡车类
26  class Truck extends Automobile {
27      //车型
28      private String carType;
29
30      public Truck(String brand, String plateNumber, String
    carType) {
31          super(brand, plateNumber);
32          this.carType = carType;
33      }
34
35      @Override
36      public int getRent(int days) {
37          int rent = 0;
38          switch (carType){
39              case "小型":
40                  rent = 300 * days;
41                  break;
42              case "中型":
43                  rent = 350 * days;
44                  break;
45              case "大型":
46                  rent = 500 * days;
47                  break;
48          }
49          return rent;
50      }
51  }
52  //巴士车
53  class Bus extends Automobile {
```

```
54      //座位数
55      private int seatNumber;
56
57      public Bus(String brand, String plateNumber, int
    seatNumber) {
58          super(brand, plateNumber);
59          this.seatNumber = seatNumber;
60      }
61
62      @Override
63      public int getRent(int days) {
64          return seatNumber <= 16 ? 400 * days : 600 * days;
65      }
66  }
```

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:

```java
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);
    }
}
```

答案：

多态是指同一操作作用于不同的对象，可以有不同的结果，产生不同的执行结果。在Java中，多态可以通过继承和接口实现。例如，我们可以定义一个动物类，然后派生出狗类和猫类，它们都重写了动物类中的speak方法，但是它们的speak方法具体实现不同，因此在调用speak方法时，会根据对象的不同而产生不同的行为。

```java
//球类
class Ball {
    public void play() {
        System.out.println("球类运动");
    }
}
//篮球类
class Basketball extends Ball {
    @Override
    public void play() {
        System.out.println("篮球运动");
    }
}
//足球类
```

```java
class Football extends Ball {
    @Override
    public void play() {
        System.out.println("足球运动");
    }
}
```

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。

答案：

重载，必须是在同一个类中，并且要求如下：

- 方法的名字必须相同
- 方法的参数列表必须不同
- 方法的修饰符、返回类型、抛出异常这些地方没有限制（可以相同，也可以不同，但一般都是相同的）

重写，在子父类中的方法：

- 方法名必须相同
- 参数列表必须相同
- 访问控制修饰符可以被扩大或不变，但是不能被缩小
- 方法抛出异常类型的范围可以被缩小或不变，但是不能被扩大
- 返回类型可以相同，也可以不同

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());
    }
}
```

答案：

编译错误，因为类 B 试图重写类 A 中的 test() 方法，但返回类型不同

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。
    }
}
```

答案：

```java
class Employee {
    private String name;
    private String gender;
    private String birthdate;
    private String type;

    public Employee(String name, String gender, String birthdate, String type) {
        this.name = name;
        this.gender = gender;
        this.birthdate = birthdate;
        this.type = type;
    }

    public String getName() {
        return name;
    }

    public String getGender() {
        return gender;
    }

    public String getBirthdate() {
        return birthdate;
    }

    public String getType() {
        return type;
    }

    public void computeSalary() {
        //空实现，待子类实现
    }
}

class Worker extends Employee {
```

```
36      private int workDays;
37
38      public Worker(String name, String gender, String
    birthdate, int workDays) {
39          super(name, gender, birthdate, "一般工人");
40          this.workDays = workDays;
41      }
42
43      @Override
44      public void computeSalary() {
45          double salary = workDays * 200;
46          System.out.println("工人 " + getName() + " 本月工资为 "
    + salary + " 元。");
47      }
48  }
49
50  class Salesman extends Employee {
51      private double baseSalary;
52      private double commission;
53
54      public Salesman(String name, String gender, String
    birthdate, double baseSalary, double commission) {
55          super(name, gender, birthdate, "销售人员");
56          this.baseSalary = baseSalary;
57          this.commission = commission;
58      }
59
60      @Override
61      public void computeSalary() {
62          double salary = baseSalary + commission;
63          System.out.println("销售员 " + getName() + " 本月工资为
    " + salary + " 元。");
64      }
65  }
66
67  class Manager extends Employee {
```

```
68      private double fixedSalary;
69
70      public Manager(String name, String gender, String
    birthdate, double fixedSalary) {
71          super(name, gender, birthdate, "经理");
72          this.fixedSalary = fixedSalary;
73      }
74
75      @Override
76      public void computeSalary() {
77          double salary = fixedSalary;
78          System.out.println("经理 " + getName() + " 本月工资为 "
    + salary + " 元。");
79      }
80  }
81
82  class Floater extends Employee {
83      private int hours;
84
85      public Floater(String name, String gender, String
    birthdate, int hours) {
86          super(name, gender, birthdate, "临时工");
87          this.hours = hours;
88      }
89
90      @Override
91      public void computeSalary() {
92          double salary = hours * 50;
93          System.out.println("临时工 " + getName() + " 本月工资为
    " + salary + " 元。");
94      }
95  }
```

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？

答案：

```
class Cylinder extends Circle {
    private double height;

    public Cylinder(double radius, double height) {
        super(radius);
        this.height = height;
    }

    public double getHeight() {
        return height;
    }

    @Override
    public double getArea() {
        return 2 * super.getArea() + super.getCircumference()
* height;
    }

```

```
18      public double getVolume() {
19          return super.getArea() * height;
20      }
21
22      @Override
23      public String toString() {
24          return "Cylinder{radius=" + getRadius() + ", height="
    + height + "}";
25      }
26  }
27  class Circle {
28      private double radius;
29
30      public Circle(double radius) {
31          this.radius = radius;
32      }
33
34      public double getRadius() {
35          return radius;
36      }
37
38      public double getArea() {
39          return Math.PI * radius * radius;
40      }
41
42      public double getCircumference() {
43          return 2 * Math.PI * radius;
44      }
45
46      @Override
47      public String toString() {
48          return "Circle{radius=" + radius + "}";
49      }
50  }
51  public class Test14_Circle {
52      public static void main(String[] args) {
```

```
53          //创建圆对象
54          Circle circle = new Circle(10);
55          System.out.println("圆的面积" + circle.getArea());
56          System.out.println("圆的周长" +
    circle.getCircumference());
57          //创建圆柱对象
58          circle = new Cylinder(20, 10);
59          System.out.println("圆柱的面积" + circle.getArea());
60          System.out.println("圆柱的体积" + ((Cylinder)
    circle).getVolume());
61      }
62  }
```

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

```
1   class Father {
2       void show() {
3           System.out.println("A");
4       }
5   }
6
7   class Son extends Father {
8       void show() {
9           super.show();
10          System.out.println("B");
11      }
12  }
13  public class Test15 {
14      public static void main(String[] args) {
```

```
        Father f = new Son();
        f.show();
    }
}
```

答案:

```
A
B
```

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);
```

```
    }
}
```

答案：

```
B: 20
A: 10
B: 20
```

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.");
    }

    public void wagTail() {
        System.out.println("I am wagging my tail.");
    }
}

public class Test17 {
    public static void main(String[] args) {
```

```
20          Animal a = new Dog();
21          a.speak();
22          a.wagTail();
23      }
24  }
```

答案：

代码 `a.wagTail();` 编译时会报错，父类引用无法调用子类特有的方法。