Day10/11 - 三大特性

百知教育 Baizhi Education

应用知识点：

- **封装**
- **继承的基础语法**
- **访问修饰符**
- **对象创建的过程**
- **super的应用**
- **多态的语法**
- **多态的应用**

习题：

1. (继承)关于继承描述错误的是()

   A. 继承体现的是类与类之间的"is-a"关系

   B. 通过继承，子类可以直接访问父类中所有的属性和方法

   C. Java 中的继承是单继承的关系

   D. 父类是子类共性的提取

2. (修饰符)下列关于访问修饰符访问权限描述错误的是()

   A. private :只能在本类中使用　　B. default :本类+同包

   C. protected :本类+同包+不同包　　D. public: 本类+同包+不同包

3. (覆盖)下列关于方法覆盖描述错误的是()

   A. 子类中的方法名必须和父类中方法名相同

   B. 子类中的方法参数列表和父类的不同

   C. 子类中的方法返回值类型和父类相同

D. 父类的方法访问修饰符和子类中的相同或是更宽

4. (继承)关于 Java 中的继承，以下说法正确的是()

A. 一个子类可以有多个直接的父类，一个父类也可以有多个直接的子类

B. 一个子类可以有多个直接的父类，但是一个父类只可以有一个直接的子类

C. 一个子类只能有一个直接的父类，但是一个父类可以有多个直接的子类

D. 以上说法都不对

5. (覆盖)仔细阅读以下代码，写出代码执行的结果：

```
class Super {
    public Super() {
        System.out.println("Super()");
    }
    public Super(String str) {
        System.out.println("Super(String)");
    }
}
class Sub extends Super {
    public Sub() {
        System.out.println("Sub()");
    }
    public Sub(int i) {
        this();
        System.out.println("Sub(int)");
    }
    public Sub(String str) {
        super(str);
        System.out.println("Sub(String)");
    }
}
public class TestSuperSub {
    public static void main(String args[]) {
        Sub s1 = new Sub();
        Sub s2 = new Sub(10);
        Sub s3 = new Sub("hello");
    }
}
```

6. (super)仔细阅读以下程序，请问如何修改代码才能通过？

```
class Super {}
class Sub extends Super {
    public Sub() {}
    public Sub(String str) {
        super(str);
    }
}
```

7. (super)仔细阅读以下代码，写出代码执行的结果。

```
class Super {
    public void m1() {
        System.out.println("m1() in Super");
    }

    public void m2() {
        System.out.println("m2() in Super");
    }
}

class Sub extends Super {
    public void m1() {
        System.out.println("m1() in Sub");
        super.m1();
    }
}

public class TestSuperSub {
    public static void main(String args[]) {
        Sub s = new Sub();
        s.m1();
        s.m2();
    }
}
```

8. (多态)仔细阅读以下代码，编译是否通过，如果通过，写出输出结果；如果不能通过，则如何修改？

```
class Super {
    public void method() {
        System.out.println("method() in Super");
    }
    public void method(int i) {
        System.out.println("method(int) in Super");
    }
}
class Sub extends Super {
    public void method() {
        System.out.println("method() in Sub");
    }
    public void method(String str) {
        System.out.println("method(String) in Sub");
    }
}
public class TestSuperSub {
    public static void main(String args[]) {
        Super s = new Sub();
        s.method(10);
        s.method();
        s.method("hello");
    }
}
```

9. (多态)仔细阅读以下代码，写出程序运行之后输出的结果。

```
class Super {
    public void m() {
        System.out.println("m() in Super");
    }
}

class Sub extends Super {
    public void m() {
        System.out.println("m() in Sub");
    }
}

public class TestSuperSub {
    public static void foo(Super s) {
        s.m();
    }

    public static void main(String args[]) {
        Sub sub = new Sub();
        Super sup = new Super();
        foo(sup);
        foo(sub);
    }
}
```

10. (访问修饰符)仔细阅读以下代码，描述正确的是()

```
1)      //MyClass.java
2)      package corejava.chp7;
3)      public class MyClass{
4)          int value;
5)      }
6)
7)      //MySubClass.java
8)      package corejava.temp;
9)      import corejava.chp7.MyClass;
10)     public class MySubClass extends MyClass{
11)         public MySubClass(int value){
12)             this.value = value;
13)         }
14)     }
```

A. 编译通过

B. 编译不通过，应把第 12 行改成 super.value = value;

C. 编译不通过，应把第 12 行改成 super(value);

D. 编译不通过，可以为 MySubClass 增加一个 value 属性

E. 编译不通过，把第 4 行改为 protected int value; 把第 12 行改为 super.value = value;

11. (访问修饰符)仔细阅读以下代码，以下代码有哪些地方编译出错？假设不允许修改 MyClass 类，那应该如何修改？

```
//MyClass.java
package corejava.chp7;
public class MyClass{
    private int value;
    public MyClass(){}
        MyClass(int value){
        this.value = value;
    }
    public int getValue(){
        return value;
    }
    public void setValue(int value){
        this.value = value;
    }
}

//TestMyClass1.java
package corejava.chp7;
public class TestMyClass1{
    public static void main(String args[]){
        MyClass mc1 = new MyClass();
        MyClass mc2 = new MyClass(10);
        System.out.println(mc1.value);
        System.out.println(mc2.value);
    }
}
//TestMyClass2.java
package corejava.temp;
import corejava.chp7.*;
public class TestMyClass2{
    public static void main(String args[]){
        MyClass mc1 = new MyClass();
        MyClass mc2 = new MyClass(10);
        System.out.println(mc1.value);
        System.out.println(mc2.value);
    }
}
```

12. (覆盖)阅读以下代码，哪些代码写在//1 处，程序编译能通过()

```
class Super {
    int method() {
        return 0;
    }
}
class Sub extends Super {
    // 1
}
```

A. public int method(){return 0;}　　B. void method(){}

C. void method(int n){}　　D. protected void method(){}

13. (对象创建过程)仔细阅读以下代码，写出代码的执行结果：

```
class Meal {
    public Meal() {
        System.out.println("Meal()");
    }
}
class Lunch extends Meal {
    public Lunch() {
        System.out.println("Lunch()");
    }
}
class Vegetable {
    public Vegetable() {
        System.out.println("Vegetable()");
    }
}
class Potato extends Vegetable {
    public Potato() {
        System.out.println("Potato()");
    }
}
class Tomato extends Vegetable {
    public Tomato() {
        System.out.println("Tomato()");
    }
}
class Meat {
    public Meat() {
        System.out.println("Meat()");
    }
}
class Sandwich extends Lunch {
    Potato p = new Potato();
    Meat m = new Meat();
    Tomato t = new Tomato();

    public Sandwich() {
        System.out.println("Sandwich()");
    }
}
public class TestSandwich {
    public static void main(String args[]) {
        Sandwich s = new Sandwich();
    }
}
```

14. (覆盖)阅读以下代码，哪些代码写在//1 处，程序编译能通过()

```
class Super {
    private void method() {
    }
}
class Sub extends Super {
    // 1
}
```

A. public int method(){return 0;}　　B. void method(){}

C. void method(int n){}　　D. private void method(){}

15. (多态)仔细阅读以下代码，下列几个选项中，有哪几个放在//1 位置能够编译通过()

```
class Animal {}
class Dog extends Animal {}
class Cat extends Animal {}
public class TestAnimal {
    public static void main(String args[]) {
        // 主方法代码省略
    }
    public static Animal getAnimal() {
        // 1
    }
}
```

A. return null;　　B. return new Animal();

C. return new Dog();　　D. return new Cat();

16. (封装)编程：定义一个 Dog 类，类中属性有名字、年龄、性别(true-公)，要求如下：

(1) 对类进行封装，并提供 get/set 方法

(2) 提供一个无参数的构造方法和一个带有三个参数的构造方法

(3) 定义一个测试类，创建对象，并对属性赋值，并将对象的信息打印在控制台上

17. (封装)编程：定义一个网络用户类(User 类)，需要处理的信息有用户 ID、用户密码 password、邮箱地址(email)，要求如下：

(1) 对类进行封装，提供 get/set 方法

(2) 提供带有两个参数的构造方法，为用户 ID 和用户密码赋值，此时 email 采用默认的：用户名加上”@zparkhr.com.cn”；同时提供带有三个参数的构造方法

(3) 定义一个测试类，创建对象，并展示用户的信息

18. (封装)编程：定义一个 Book 类(代表教材)，具有属性名称(title)、页数(pageNum)，要求如下：

(1) 对类进行封装，属性私有化，并提供公开的 get/set 方法；其中要求页数不能少于 200 页，否则输出”错误信息”，并赋予默认值 200

(2) 提供无参数和有参数的构造方法

(3) 编写一个测试类，创建对象并为属性赋值，将对象的信息展示在控制台上

19. (封装)编程：已知一个 Student 类，代码如下：

```
class Student{
    String name;
    int age;
    String address;
    String zipCode;
    String mobile;
}
```

(1) 将 Student 类进行封装，即属性均私有化，并提供 get/set 方法

(2) 为 Student 类添加一个 getPostAddress 方法，要求返回 Student 对象的地址和邮编

(3) 定义一个测试类，创建对象并为属性赋值，将用户的信息进行展示

20. (继承+封装)编程：定义一个人类(Person)，包括属性：姓名、性别、年龄、国籍；

包括的方法:吃饭、睡觉，工作

(1) 根据人类，定义一个子类，增加属性：学校、学号；重写工作方法(实现内容为学习)

(2) 根据人类，定义一个工人类，增加属性：单位，工龄；重写工作方法

(3) 根据学生类，定义一个学生干部类(StudentLeader)，增加属性：职务；增加方法：开会

(4) 定义一个测试类，分别创建上述 3 类具体人物的对象并将信息打印在控制台上。

21. (多态)在上一个题目的基础上，定义一个 Person 类型的数组，存储多个不同类型的子类型对象，

(1) 统计并打印输出数组中所有学生干部的个数

(2) 打印输出所有学生的信息

22. (继承+封装)编程：定义一个交通工具类(Vehicles)，该类的属性为：商标(brand)、颜色(color)；功能方法为：run 方法(行驶功能，控制台输出“车已经启动”)、showInfo(显示信息，控制台输出商标和颜色)

(1) 编写一个小汽车类(Car)继承于 Vehicles 类，添加属性座位(seats)；成员方法 showCar(显示小汽车的所有信息)

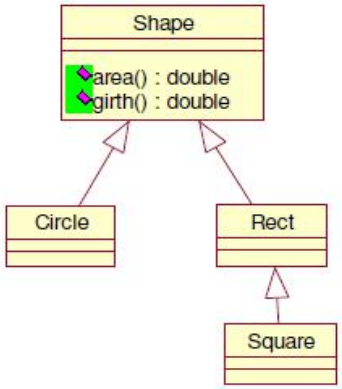
(2) 编写一个卡车类(Truck)继承于 Vehicles 类，添加属性载重(load)；成员方法 showTruck(显示卡车的所有信息)

(3) 定义测试类，分别创建 Car 对象和 Truck 对象，控制台打印输出的信息如下：

商标：奔驰，颜色：白色，座位：5

商标：福田，颜色：红色，载重：6.5 吨

23. 编程：有以下几个类，根据下面的继承关系，用 Java 代码实现：



(1) Circle 类（圆形），属性：半径；方法：求周长、求面积

(2) Rect 类（矩形），属性：长、宽；方法：求周长、求面积

(3) Square 类（正方形），属性：边长；方法：求周长、求面积

提示： ① 这三个类均具有求周长和面积的方法　　② 正方形是特殊的矩形

24. 编程：在上一题的基础上，创建一个长度为 3 的数组，里面有三个不同类型的对象，分别打印这三个对象的周长和面积。

25. 编程：阅读以下代码，根据要求完成程序功能。

```
class Animal {
    private String name;
    // 1
}
class Dog extends Animal {
    // 2
}
class Cat extends Animal {
    // 3
}
public class TestAnimal {
    public static void main(String args[]) {
        Animal[] as = new Animal[] { new Dog("Pluto"), new Cat("Tom"),
                new Dog("Snoopy"), new Cat("Garfield") };
        Dog[] dogs = getAllDog(as);
        for (int i = 0; i < dogs.length; i++) {
            System.out.println(dogs[i].getName());
        }
    }
    public static Dog[] getAllDog(Animal[] as) {
        // 4
    }
}
```

(1) 在程序的 1、2、3 处填上适当的 构造方法或 get/set 方法

(2) 完成 4 处的填空：getAllDog 方法从一个 Animal 数组中挑选出所有的 Dog 对象，并把这些对象放在一个 Dog 数组中返回

26. 编程：某公司的雇员分为以下若干类：

(1) Employee：这是所有员工总的父类。

① 属性：员工的姓名,员工的生日月份

② 方法：getSalary(int month) 根据参数月份来确定工资，如果该月员工过生日，则公司会额外奖励 100 元。

(2) SalariedEmployee：Employee 的子类，拿固定工资的员工。

① 属性：月薪。

(3) HourlyEmployee：Employee 的子类，按小时拿工资的员工，每月工作超出 160 小时的部分按照 1.5 倍工资发放。

① 属性：每小时的工资、每月工作的小时数。

(4) SalesEmployee：Employee 的子类，销售，工资由月销售额和提成率决定。

① 属性：月销售额、提成率。

(5) BasePlusSalesEmployee：SalesEmployee 的子类，有固定底薪的销售人员，工资由底薪加上销售提成部分。

① 属性：底薪。

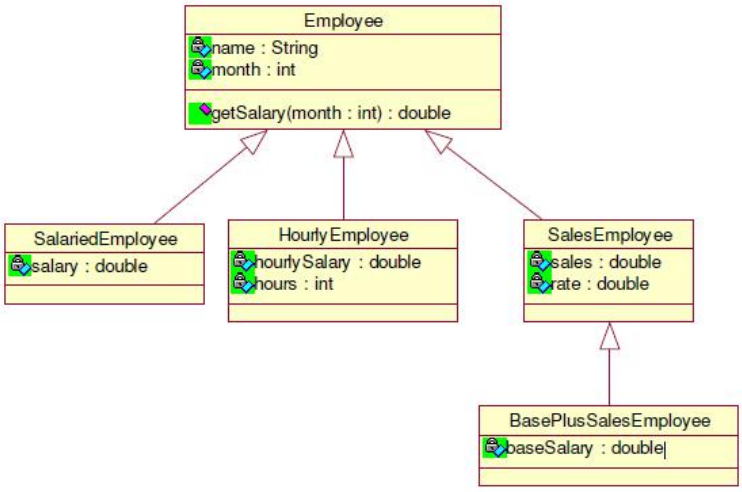
要求：

(1) 创建 SalariedEmployee、HourlyEmployee、SaleEmployee、BasePlusSalesEmployee 四个类的对象各一个

(2) 并调用父类 getSalary(int money)方法计算某个月这四个对象各自的工资

注意：要求把每个类都做成完全封装，不允许非私有化属性。

类图如下：



27. 编程：在上一题的基础上，创建一个 Employee 数组，分别创建若干不同的 Employee 对象，并打印某个月的工资。