Chapter 06 面向对象三大特性

Key Point：

- 封装/数据隐藏
- 继承的基本语法
- 访问修饰符
- 对象创建过程
- super 关键字
- 方法覆盖
- 多态的基本语法和使用
- instanceof
- 多态用在参数和返回值上

问题：

1. （继承、this 和 super 关键字）有以下代码

```
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");
    }
}
```

写出该程序运行的结果。

2. （super）看下面代码，写出程序运行的结果

```
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();
        }
    }
```

3. (多态) 有如下代码

```
    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");
    }
}
```

问：该程序是否能编译通过？如果可以，输出结果是什么？如果不可以，应该如何 修改？

4. （多态）有如下代码：

```
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);
    }
}
```

写出该程序编译运行后输出的结果。

5. （多态）有如下代码

```
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
    }
}
```

问：下列几个选项中，有哪几个放在//1 位置能够编译通过？

A. return null;

B. return new Animal();

C. return new Dog();

D. return new Cat();

6. (访问修饰符)有如下代码

```
//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);
    }
}
```

以上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？

7. （继承、访问修饰符）有如下代码

```
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) }
```

选择正确答案：

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;

8. （继承、对象构造过程）有以下代码

```
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();
```

```
    }
}
```

写出这段代码的输出结果。

9. （默认构造函数）有以下代码

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

问：该程序应该如何修改才能编译通过？

10. （方法覆盖）有如下代码

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

在//1 处，能编译通过的代码为：

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

B. void method(){}

C. void method(int n){}

11. （方法覆盖）有如下代码

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

在//1 处，能编译通过的代码为：

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

B. void method(){}

C. void method(int n){}

D. private void method(){}

12. （多态，instanceof）有如下代码

```
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
    }
}
```

程序填空：

I. 在 //1, //2, //3 处填上适当的 get/set 方法和构造方法

II. 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。

13. （封装）将 Chp5 中的 Worker、Address 类中所有的属性都改成私有，并提供相 应的 get/set 方法。

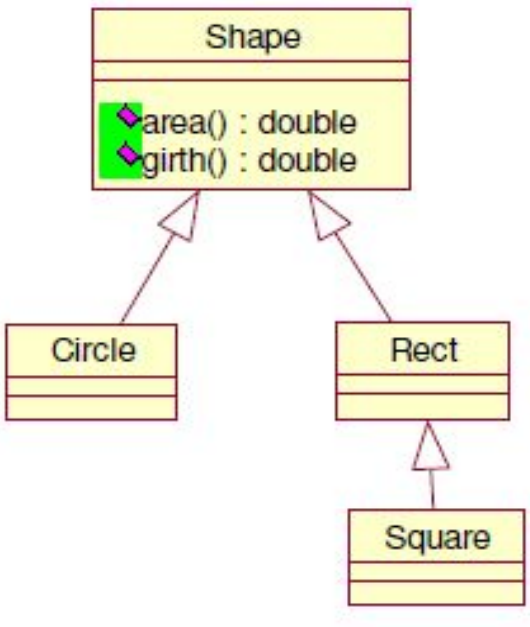
14. （封装）已知一个类 Student 代码如下：

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

I. 把 Student 的属性都作为私有，并提供 get/set 方法以及适当的构造方法。

II. 为 Student 类添加一个 getPostAddress 方法，要求返回 Student 对象的地址和邮编。

15. （封装、继承）有以下几个类，根据下面的继承关系，用 Java 代码实现。



I. Circle 类（圆形），属性：半径；方法：求周长、求面积。

II. Rect 类（矩形），属性：长、宽；方法：求周长、求面积。

III. 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。

16. （多态）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。

17. （封装、继承、super）某公司的雇员分为以下若干类：

I. 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。

II. SalariedEmployee：Employee 的子类，拿固定工资的员工。

1). 属性：月薪。

III. 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。

IV. 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。

V. 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。

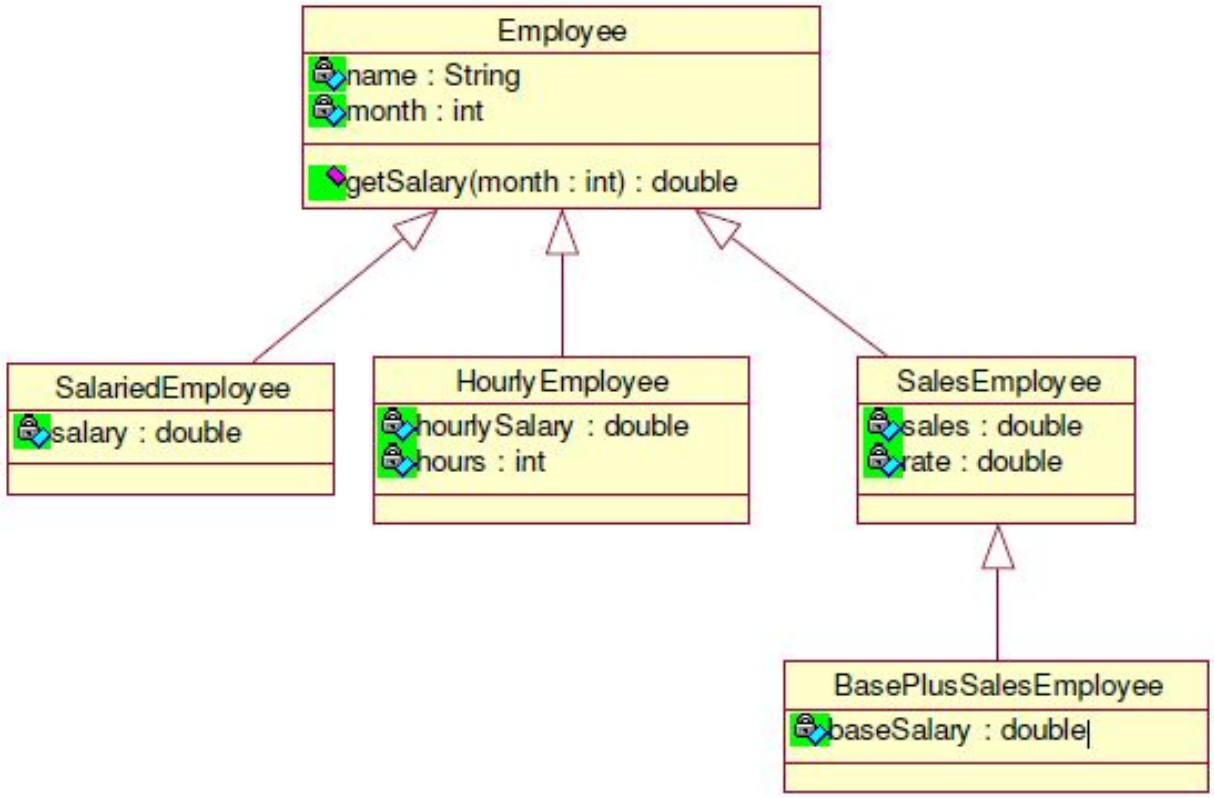
要求：

I. 创建 SalariedEmployee、HourlyEmployee、SaleEmployee、BasePlusSalesEmployee 四个类的对象各一个。

II. 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：



18. （多态）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。