

Chp5 面向对象基础

Key Point

- 类和对象的概念
- 实例变量
- 方法重载
- 构造方法
- 引用的概念
- this 关键字

练习

1. （重载，实例变量）有以下代码：

```
class ClassA{
    public void method(int value){
        System.out.println(value);
    }
    public void method(){
        System.out.println(value);
    }
    int value;
}

class TestClassA{
    public static void main(String args[]){
        ClassA classA = new ClassA();
        classA.value = 10;
        classA.method();
        classA.method(20);
    }
}
```

请选择正确结果：

- A. 编译不通过
 - B. 输出10 10
 - C. 输出 10 20**
 - D. 输出0 20
2. （方法重载，函数返回值）有以下代码

```
class ClassA{
    void method(){
        System.out.println("method()");
    }
    int method(int i){
        System.out.println("method(int)");
    }
    public static void main(String args[]){
        ClassA a = new ClassA();
    }
}
```

int类型 缺少返回值
或者 把int改为 void

```

        a.method();
        a.method(10);
    }
}

```

该程序是否能编译通过？如果可以，写出该程序运行结果。如果不能，请说明理由，以及如何修改。

3. （构造方法）关于构造方法，下列说法正确的是：

- A. 每个类中都有至少一个构造方法
- B. 一个类中可以有多个构造方法
- C. 构造方法可以有返回值
- D. 构造方法可以有多个参数



4. （引用）有以下代码

```

class MyClass{
    int value;
}

public class TestRef{
    public static void main(String args[]) {
        int a = 10;
        int b = a;
        b ++ ;
        System.out.println(a);
        MyClass mc1 = new MyClass();
        mc1.value = 10;
        MyClass mc2 = mc1;
        mc2.value ++;
        System.out.println(mc1.value);
    }
}

```

m2和m1指向的是同一个对象

10
11

请写出编译运行后的结果。

5. （引用）有以下代码

```

class ClassA{
    int value = 10;
}

public class TestReturnRef{
    public static void main(String args[]) {
        ClassA ca = new ClassA();
        ca = getObject();
        ca = getObject();
        ca = getObject();
        System.out.println(ca.value);
    }

    public static ClassA getObject() {
        ClassA newObject = new ClassA();
        newObject.value += 10;
    }
}

```

out: 20

//每调用一次，创建一个新对象

```

        return newObject;
    }
}

```

编译运行TestReturnRef 程序，结果为：

- A. 编译出错
- B. 输出10
- C. 输出20**
- D. 输出40

6. （构造函数）有以下代码

```

class MyClass{
    int value;
}

```

```

public class TestMyClass{
    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);
    }
}

```

不能编译，缺少两个构造函数

```

class MyClass(){
class MyClass(int value){
    this.value=value;
}
}

```

问：这个程序能否编译通过？如果可以，输出结果是什么？如果不可以，则应该如何修改？

7. （面向对象基础）根据注释，把下面代码补充完整

//定义一个Dog 类

```

class Dog{
    //定义一个name 属性，该属性为String 类型
    String name;
    //定义一个age 属性，该属性为int 类型
    int age;
    //定义一个sexual 属性，该属性为boolean 类型
    //true 表示为公，false 表示为母
    boolean sexual;
    public Dog() {}
    public Dog(String name, int age, boolean sexual){
        //分别根据参数，设置Dog 类的属性
    }
    public void play(){
        System.out.println(name + “ play” );
    }
    public void play(int n){
        System.out.println(name + “ play ” + n + “ minutes” );
    }
}
public class TestDog{

```

```

this.name=name;
this.age=age;
this.sexual=sexual;

```

```

public static void main(String args[]) {
    Dog d;
    //创建一个Dog 对象，利用带参数的构造函数
    //名字为joy，年龄为2 岁，性别为母
    d=new( "joy", 2, false);
    //调用Dog 对象无参的play 方法。
    d.paly;
    //调用Dog 对象有参的play 方法，参数为30
    d.paly(30);
}
}

```

8. * (对象创建过程) 有以下代码

```

class ClassA{
    public ClassA() {
        System.out.println("ClassA()");
    }
}
class ClassB{
    public ClassB() {
        System.out.println("ClassB()");
    }
}
class ClassC{
    ClassA a = new ClassA();
    ClassB b;
    public ClassC() {
        System.out.println("ClassC()");
        b = new ClassB();
    }
}
public class TestConstructor{
    public static void main(String args[]) {
        ClassC cc = new ClassC();
    }
}

```

here

请选择正确答案:

- A. 编译不通过
 - B. 输出ClassA() ClassB() ClassC()
 - C. 输出 ClassA() ClassC() ClassB()**
 - D. 输出 ClassC() ClassB() ClassA()
9. * (引用，方法参数传递) 有以下代码

```

class ClassA{
    int value;
}

```

```

public class TestClassA{
    public static void main(String args[]) {
        int value = 10;
        changeInt(value);
        System.out.println(value);
        ClassA ca = new ClassA();
        ca.value = 10;
        changeObject(ca);
        System.out.println(ca.value);
    }
    public static void changeInt(int value) {
        value++;
    }
    public static void changeObject(ClassA ca) {
        ca.value++;
    }
}

```

Handwritten annotations: Blue arrows show the flow of value updates. In `main`, `value` is 10, then `changeInt` increments it to 11, then `println` prints 11. Then a new `ClassA` object `ca` is created with `value = 10`, then `changeObject` increments `ca.value` to 11, and finally `println` prints 11. To the right, the output "10 11" is handwritten.

编译运行TestClassA 时，结果是

- A. 编译出错
- B. 输出 10 11
- C. 输出 10 10
- D. 输出 11 11

10. * (构造函数，this 关键字) 程序改错

```

public class Student{
    public void Student() {}
    void init() {
        age = 10;
        name = "limy";
    }
    public Student(String name) {
        this.init();
        this.name = name;
    }
    public Student(String name, int age) {
        this.init();
        this(name);
        this.age = age;
    }
    int age;
    String name;
}

```

Handwritten annotations: A blue squiggle is under the first `public`. A blue box highlights `this.init();` and `this(name);` in the third constructor. A blue arrow points from `this(name);` to the right.


`this(name);`
`this.init();`

11. (面向对象基础) 写一个Worker 类，并创建多个Worker 对象。

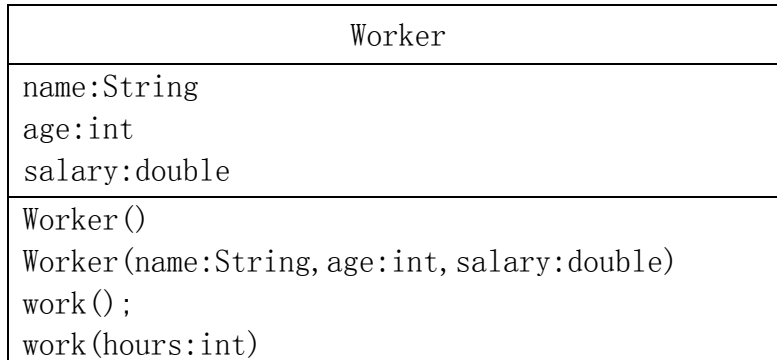
- 1) 为Worker 类添加三个属性，1) String 类型的name，表示工人的姓名；
- 2) int 类型的age，表示工人的年龄；
- 3) double 类型的salary，表示工人

的工资。

2) 为Worker 类添加两个构造方法，1) 公开无参构造方法；2) 接受三个参数的构造方法，三个参数分别为字符串、int 和double 类型。

3) 为Worker 类添加两个work 方法，一个无参，另一个带整数参数，表示工人工作的时间（为多少小时）。 

类图如下：

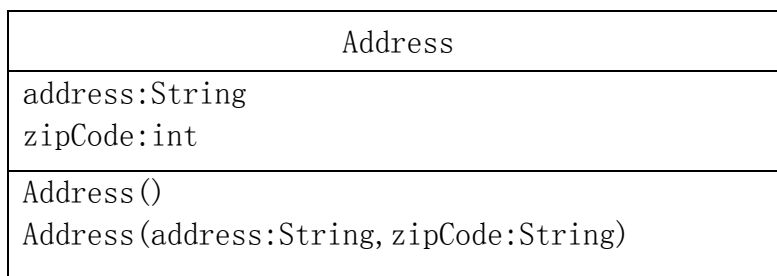


12. （面向对象基础）创建一个Address 类，描述如下：

1) 该类有两个属性，1) String 类型的address，表示地址；2) String 类型的zipCode，表示邮编。

2) 该类有两个构造方法，一为无参构造方法，一为带三个参数的方法。

类图如下：



13. *（面向对象基础）为第1 题中的Worker 类添加一个属性addr，该属性为Address 类型。

创建一个Worker 对象，其姓名为"zhangsan"，年龄为25，工资为2500，家庭住址为“北京市海淀区清华园1 号”，邮政编码为100084。

14. **（引用，方法参数传递）有以下代码

```
class ClassA{
    int value;
}
public class ChangeRef{
    public static void main(String args[]){
        ClassA ca = new ClassA();
        changeValue(ca);
        System.out.println(ca.value);
        changeRef(ca);
        System.out.println(ca.value);
    }
}
```

```

    }
    public static void changeValue(ClassA ca) {
        ca.value = 100;
    }
    public static void changeRef(ClassA ca) {
        ca = new ClassA();
        ca.value = 200;
    }
}

```

又new了一个对象，和原来的ca就没有关系了

编译运行ChangeRef，结果为

- A. 编译不通过
- B. 输出100 200
- C. 输出100 100
- D. 输出0 200

15. （面向对象基础）**复数概念如下：

每个复数都有实部和虚部。例如， $3 + 5i$ ，3 为实部， $5i$ 为虚部。其中， i 称为虚数单位，有 $i*i = -1$ 。

两个复数进行加法运算，运算时实部与实部相加，虚部与虚部相加。例如：
 $(1.5 - 3i) + (2.3 + 2.4i) = (1.5+2.3) + (-3 + 2.4)i = 3.8 - 0.6i$

两个复数进行减法运算，与加法运算类似。

两个复数进行乘法运算，其过程如下：

$$(a+bi) * (c + di) = ac + adi + bci + bd(i*i) = (ac-bd) + (ad+bc)i$$

例如：

$$(3+5i) * (4+6i) = (3*4-5*6) + (3*6+4*5)i = -18 + 38i$$

写一个类Complex，用来表示复数。这个复数类具有两个属性：double real，表示实部；double im，表示虚部。并为Complex 类增加add、sub、mul 方法，分别表示复数的加法、减法和乘法运算。其中，add 方法的声明如下：

```

public Complex add(Complex c) //表示当前Complex 对象与参数c 对象相加
public Complex add(double real) //表示当前 Complex 对象与实数real 相加

```

