1. 什么是方法的重载？

2. 说明 java 方法中的参数传递机制的具体体现？

值传递。

3. 类的方法内是否可以定义变量？是否可以调用属性？是否可以定义方法？是否可以调用方法？

4.提供如下代码的内存解析

```
class Value {
    int i = 15;
}
```

```
class Test {
    public static void main(String args[]) {
        Test t = new Test();
        t.first();
    }

    public void first() {
        int i = 5;
        Value v = new Value();
        v.i = 25;
        second(v, i);
        System.out.println(v.i);
    }
    public void second(Value v, int i) {
        i = 0;
        v.i = 20;
        Value val = new Value();
        v = val;
        System.out.println(v.i + " " + i);
    }
}
```

4.面向对象中两个重要的概念：类、对象(或实例)

类：对一类事物的描述、抽象的、概念上的定义
对象：由类所派生出来的，真实存在的一个结构、实例。

二者的关系：对象，是由类派生(new)出来的。

5.面向对象思想落地实现的规则一

```
* 1. 创建类及提供类的成员：属性、方法
* 2. 类的实例化（创建类的对象
* 3. 调用对象的相关结构:"对象.属性" 或 "对象.方法"
```

补充：几个概念的使用说明

```
* 属性 - 成员变量 - field - （域 - 字段）
* 方法 - 成员方法 - 函数 - method
* 创建类的对象 - 实例化类 - 类的实例化
```

4.7 面向对象特征之一：封装和隐藏

- 为什么需要封装？封装的作用和含义？
  - 我要用洗衣机，只需要按一下开关和洗涤模式就可以了。有必要了解洗衣机内部的结构吗？有必要碰电动机吗？
  - 我要开车，…
- 隐藏对象内部的复杂性，只对外公开简单的接口。便于外界调用，从而提高系统的可扩展性、可维护性。
- 我们程序设计要追求“高内聚，低耦合”。
  - 高内聚 ：就是类的内部数据操作细节自己完成，不允许外部干涉；
  - 低耦合 ：仅暴露少量的方法给外部使用。

```
/*
 * 面向对象的特征一：封装与隐藏
 *
 * 1.封装性设计的由来？
 * 创建了类的对象以后，我们可以使用“对象.属性”方式调用或设置属性的值。在赋值时，要求考虑到变量的数据类型和储值范围。
 * 但是，我们在实际问题，需要额外的给属性赋值时，加入限制条件。这些限制条件不可能在变量声明时做添加，我们只能通过在方法中
 * 给变量赋值，同时添加限制条件（setXxx()体现）。同时，我们应该禁止直接通过“对象.属性”的方式给属性赋值。
 *
 *    此外，为了能调用此属性，我们再提供获取属性的方法（getXxx()）
 *
 * 2. 封装性的体现（狭义上）
 * 将类的属性私有化，同时，提供公共的get()和set()方法来获取和设置此属性
 *
 * 3. 封装性的体现（广义上）
 * java规范的4种权限修饰：private < 缺省 < protected < public
 *
 *
 *
```

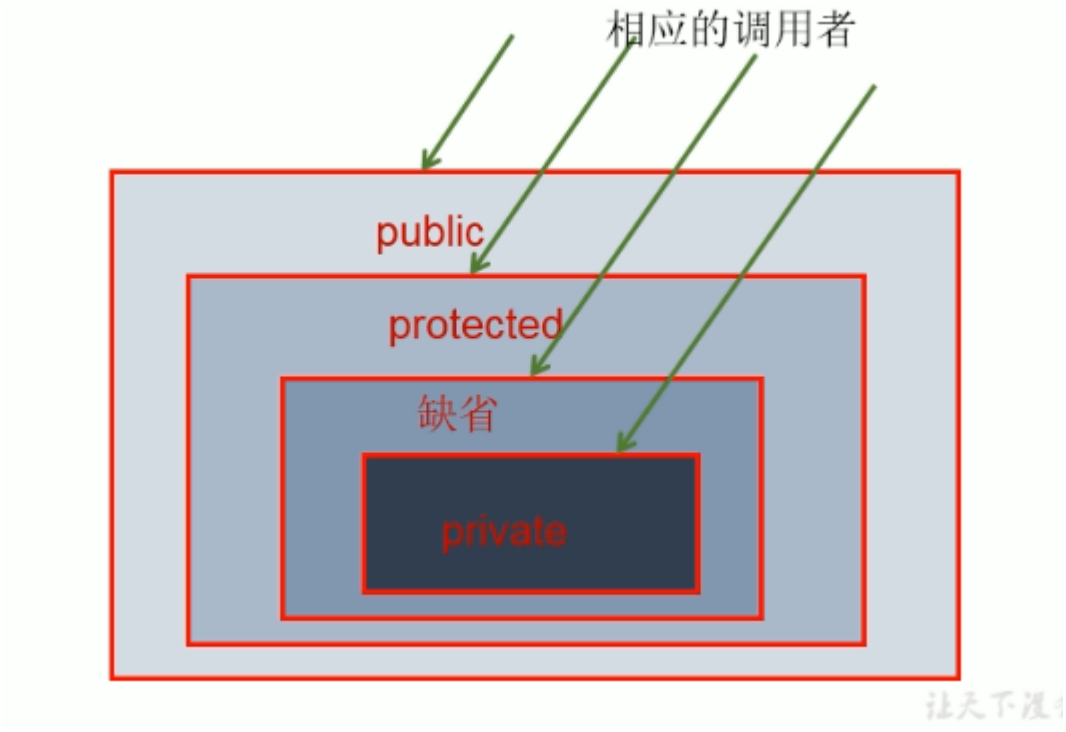
四种访问权限修饰符

Java权限修饰符public、protected、private置于类的成员定义前，用来限定对象对该类成员的访问权限。

修饰符	类内部	同一个包	不同包的子类	任何地方
private	Yes			
(缺省)	Yes	Yes		
protected	Yes	Yes	Yes	
public	Yes	Yes	Yes	Yes

对于class的权限修饰只可以用public和default(缺省)。

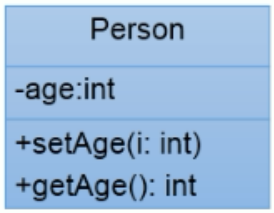
- public类可以在任意地方被访问。
- default类只可以被同一个包内部的类访问。



练习4

1.创建程序,在其中定义两个类：Person和PersonTest类。定义如下：

用setAge()设置人的合法年龄(0~130)，用getAge()返回人的年龄。在PersonTest类中实例化Person类的对象b，调用setAge()和getAge()方法，体会Java的封装性。



实验 1：

1、将对象作为参数传递给方法。

题目要求：

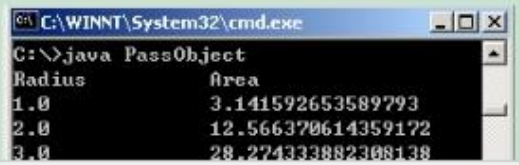
（1）定义一个 Circle 类，包含一个 double 型的 radius 属性代表圆的半径，一个 findArea() 方法返回圆的面积。

（2）定义一个类 PassObject，在类中定义一个方法 printAreas()，该方法的定义如下：

public void printAreas(Circle c, int time)

在 printAreas 方法中打印输出 1 到 time 之间的每个整数半径值,以及对应的面积。例如,times 为 5，则输出半径 1，2，3，4，5，以及对应的圆面积。

在 main 方法中调用 printAreas()方法，调用完毕后输出当前半径值。程序运行结果如图所示。

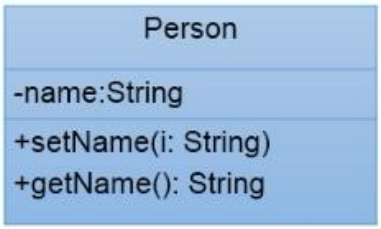


```java
/*
 * 类的成员之三：构造器(或构造方法),Constructor
 *       construct:建设、建造  CCB   ICBC
 *
 * 1.构造器的作用：创建类的对象；初始化对象的属性
 *
 *
 * 2. 说明：
 * ① 如果我们没有显式的提供类的构造器的话，则系统会默认给一个类提供默认的构造器：无参数的。
 * ② 我们如果显式的声明类的构造器的话，格式为：权限修饰符 类名(形参列表){}
 * ③ 同一个类中的多个构造器之间构成重载。
 * ④ 如果我们显式的提供了类的构造器，则系统不再提供默认的空参的构造器
 * ⑤ java的类中一定存在构造器。
 */
```

练习5

1. 在前面定义的Person类中添加构造器，利用构造器设置所有人的age属性初始值都为18。

2. 修改上题中类和构造器，增加name属性,使得每次创建Person对象的同时初始化对象的age属性值和name属性值。




```java
/*
 * 类的属性的赋值的先后顺序：
 *
 * ① 属性的默认初始化
 * ② 属性的显式初始化
 * ③ 构造中给属性初始化
 * ④ 通过"对象.方法" 或"对象.属性"的方法，给属性赋值
 *
 *
```

```
/*
 * 类的属性的赋值的先后顺序：① - ② - ③ - ④
 *
 * ① 属性的默认初始化
 * ② 属性的显式初始化
 * ③ 构造器中给属性初始化
 * ④ 通过"对象.方法" 或 "对象.属性"的方法，给属性赋值
 *
 * 说明：上述的操作① ② ③ 在执行中只调用一次。④ 可以根据用户需求多次调用
 *
 */
```