笔记总链接：http://bbs.itheima.com/thread-200600-1-1.html

3、面向对象

3.4 构造函数

特点：

1、函数名与类名相同。

2、不用定义返回值类型。

3、没有具体的返回值。

P.S.

在构造函数前面加上返回值就只是一般函数了。

作用：给对象进行初始化。

示例：

```java
class Person{
        private String name ;
        private int age ;

        //定义一个Person类的构造函数
        //构造函数，而且是空参数的
       Person(){
             System.out.println("person run");
       }

        public void speak(){
             System.out.println(name + ":" + age);
       }
}

class ConsDemo{
        public static void main(String[] args){
              //构造函数：构建创造对象时调用的函数
              //作用：可以给对象进行初始化
             Person p = new Person();
             p.speak();
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac ConsDemo.java

D:\code\day07>java ConsDemo
person run
null:0
```

P.S.

1、一般函数和构造函数什么区别呢？

构造函数：对象创建时，就会调用与之对应的构造函数，对对象进行初始化。

一般函数：对象创建后，需要函数功能时才调用。

构造函数：对象创建时，会调用并且只调用一次。

一般函数：对象创建后，可以被调用多次。

2、创建对象都必须要通过构造函数初始化。

一个类中如果没有定义过构造函数，那么该类中会有一个默认的空参数构造函数。

如果在类中定义了指定的构造函数，那么类中的默认构造函数就没有了。

3、多个构造函数是以重载的形式存在的。

示例：

```java
class Person{
          private String name ;
          private int age ;

          Person(){
                   name = "baby" ;
                   age = 1;
                   System.out.println("person run");
          }

          //如果有的孩子一出生就有名字
          Person(String n){
                 name = n;
          }

           //如果有的孩子一出生就有名字和年龄
          Person(String n, int a){
                 name = n;
                 age = a;
          }

          public void speak(){
              System.out.println(name + ":" + age);
          }
}

class ConsDemo{
        public static void main(String[] args){
             Person p1 = new Person();
             p1.speak();
             Person p2 = new Person("旺财" );
             p2.speak();
             Person p3 = new Person("小强" ,10);
             p3.speak();
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac ConsDemo.java

D:\code\day07>java ConsDemo
person run
baby:1
旺财:0
小强:10
```

3.5 this关键字

this代表其所在函数所属对象的引用。换言之，this代本类对象的引用。

当成员变量和局部变量重名，可以用关键字this来区分，this就是所在函数所属对象的引用。

简单说，哪个对象调用了this所在的函数，this就代表哪个对象。一般方法调用默认加this。

什么时候使用this关键字呢？

当在函数内需要用到调用该函数的对象时，就用this。

示例：

```java
class Person{
          private String name ;
          private int age ;

          Person(String name){
             //通过this区分成员变量和局部变量
             this.name = name;
          }

          Person(String name, int age){
             //this也可以用于在构造函数中调用其他构造函数
             this(name);
             this.age = age;
          }

          public void speak(){
             System.out.println(name + ":" + age);
          }
}

class ConsDemo{
        public static void main(String[] args){
             Person p1 = new Person("旺财" );
             p1.speak();
             Person p2 = new Person("小强" ,10);
             p2.speak();
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac ConsDemo.java

D:\code\day07>java ConsDemo
旺财:0
小强:10
```

P.S.

通过this在构造函数中调用其他构造函数的时候，只能定义在构造函数的第一行，因为初始化动作要先执行，否则就会报错。

示例：

```java
class Person{
          private String name ;
          private int age ;

          Person(String name, int age){
             this.name = name;
             this.age = age;
          }

          /*
          判断是否是同龄人
          */
          public boolean compare(Person p){
                return this.age == p.age;
          }
}

class ConsDemo{
        public static void main(String[] args){
             Person p1 = new Person("旺财" ,10);
             Person p2 = new Person("小强" ,10);
             System.out.println(p1.compare(p2));
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac ConsDemo.java

D:\code\day07>java ConsDemo
true
```

3.6 static关键字

static关键字：用于修饰成员（成员变量和成员函数）。

被修饰后的成员具备以下特点：

1、随着类的加载而加载。

2、优先于对象存在。

3、被所有对象所共享。

4、可以直接被类名调用。

示例：

```java
class Person{
         //成员变量，实例变量
        String name;
        //静态变量，类变量
        //所有对象共享的属性用static修饰
        static String country = "CN";
        public void show(){
             System.out.println(country + ":" + name);
              //等效语句：System.out.println(Person.country + ":" + this.name);
       }
}

class StaticDemo{
        public static void main(String[] args){
             Person p = new Person();
             System.out.println(p.country);
              //可以用类名直接调用
             System.out.println(Person.country);
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac StaticDemo.java

D:\code\day07>java StaticDemo
CN
CN
```

成员变量和静态变量的区别？

1. 两个变量的生命周期不同

成员变量随着对象的创建而存在，随着对象被回收而释放。

静态变量随着类的加载而存在，随着类的消失而消失。

2. 调用方式不同

成员变量只能被对象调用。

静态变量可以被对象调用，还可以被类名调用。

3. 别名不同

成员变量也称为实例变量。

静态变量也称为类变量。

4. 数据存储位置不同

成员变量存储在堆内存的对象中，所以也叫对象的特有数据。

静态变量数据存储在方法区（共享数据区）的静态区，所以也叫对象的共享数据。

使用注意：

1. 静态方法只能访问静态成员，如果访问非静态成员，就会报错！

原因：静态方法和变量存在的时候，对象还不存在，非静态变量也不存在，肯定无法访问。

示例：

```java
class Person{
        String name;
        static String country = "CN";
        //静态方法
        public static void show(){
             System.out.println(country + ":" + name);
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac Person.java
Person.java:6: 无法从静态上下文中引用非静态 变量 name
            System. out.println(country + ":" + name);
                                                ^
1 错误
```

P.S.

非静态方法既可以访问静态成员，又可以访问非静态成员。

2、静态方法中不可以写this，super关键字。

原因：静态方法存在的时候，对象还不存在，this代表的就是调用方法的那个对象，既然不存在，肯定不可以写在静态方法中。

3、主函数是静态的，主函数特殊之处：

1. 格式是固定的。

2. 被jvm所识别和调用。

主函数各成分解析：

public：权限必须是最大的，被jvm调用。

static：不需要对象调用，直接用主函数所属类名调用即可。 命令行窗口中输入：java StaticDemo，实际上就是在执行StaticDemo.main();。

void：主函数不需要具体的返回值。

main：函数名，不是关键字，只是一个jvm识别的固定的名字。

String[] args：这是主函数的参数列表，是一个数组类型的参数，而且元素都是字符串类型。

示例：

```java
class MainDemo{
        public static void main(String[] args){
              for(int x = 0; x < args.length; x++){
                   System.out.println(args[x]);
             }
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac MainDemo.java

D:\code\day07>java MainDemo hehe haha xixi
hehe
haha
xixi
```

静态什么时候用？

1. 静态变量

当分析对象中所具备的成员变量的值都是相同时，这时这个成员就可以被静态修饰。

只要数据在对象中都是不同的，就是对象的特有数据，必须存储在对象中，是非静态的。

如果是相同的数据，对象不需要做修改，只需要使用即可，不需要存储在对象中，定义成静态的。

2. 静态函数

函数是否用静态修饰，就参考一点，就是该函数功能是否需要访问到对象中的特有数据。

简单点说，从源代码看，该功能是否需要访问非静态的成员变量，如果需要，该功能就是非静态的。

如果不需要，就可以将该功能定义成静态的。当然，也可以定义成非静态，但是非静态需要被对象调用。

如果没有访问特有数据的方法，该对象的创建是没有意义。

示例：

```java
class Demo{
        int age ;
        static int num = 9;

        public static void speak(){
             System.out.println(num);
       }

        public void show(){
             System.out.println(age);
       }
}

class StaticDemo{
        public static void main(String[] args){
             Demo.speak();
             new Demo().show();
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac StaticDemo.java

D:\code\day07>java StaticDemo
9
0
```

静态代码块

随着类的加载而执行，而且只执行一次。

作用：用于给类进行初始化。

示例：

```java
class StaticCode{
        static int num;
        static{
              num = 10;
       }
        static void show(){
             System.out.println(num);
       }
}

class StaticCodeDemo{
        public static void main(String[] args){
             StaticCode.show();
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac StaticCodeDemo.java

D:\code\day07>java StaticCodeDemo
10
```

构造代码块

作用：可以给所有对象进行初始化。

示例：

```java
class Person{
        private String name ;

        //构造代码块，可以给所有对象进行初始化的
       {
            System.out.println("person run");
       }

        //是给对应的对象进行针对性的初始化
       Person(){
             name = "baby" ;
       }

       Person(String name){
             this.name = name;
       }

        public void speak(){
            System.out.println("name:" + name);
       }
}

class StaticCodeDemo{
        public static void main(String[] args){
            Person p1 = new Person();
            p1.speak();
            Person p2 = new Person("旺财");
            p2.speak();
       }
}
```

复制代码

运行结果：

```
D:\code\day07>javac StaticCodeDemo.java

D:\code\day07>java StaticCodeDemo
person run
name:baby
person run
name:旺财
```

~END~

~爱上海，爱黑马~