一、面向对象基础

1、什么是面向对象：

面向对象(Object Oriented)是软件开发方法，面向对象是一种对现实世界理解和抽象的方法。

2、面向对象和面向过程的对比

2.1 优点

- 易维护：代码可读性高
- 易扩展：高内聚、低耦合
- 开发效率高：开发时可使用抽象

2.2 不足

开销比较大，比较消耗资源，所以当性能是最重要的考量因素的时候不适合使用面向对象语言，比如单片机、嵌入式开发、Linux/Unix等一般采用面向过程开发。

3、成员变量和局部变量

4.1 成员变量：对象的属性也称为成员变量

4.2 局部变量：在成员方法中定义一个变量，那么这个变量就被称为局部变量。

4.3 两者之间区别：

- 定义的位置不同：成员变量在方法的外部，直接写在类当中。局部变量在方法的内部。
- 作用域不同：成员变量在整个类可以通用。局部变量只有在方法中才可以使用，出了方法就不可以使用。
- 默认值不一样：成员变量如果没有进行赋值，会有默认值。局部变量没默认值，需要赋值才可以使用。
- 内存位置不一样：成员变量位于堆当中，局部变量位于栈当中。
- 生命周期不同：成员变量随着对象创建而诞生，随着对象被垃圾回收而消失。局部变量随着方法进栈而诞生随着方法出栈而消失。

4、成员方法

4.1.1 实例方法

- 格式

```
public void testA()
{
        System.out.printf("testA");
}
```

- 运行时机：

属于实例对象，实例化后才会分配内存，必须通过类的实例来引用。不会常驻内存，当实例对象被JVM 回收之后，也跟着消失

4.1.2 静态方法

- 格式

```
public static void testB()
{
     System.out.printf("testB");
}
```

- 运行时机：

  属于类本身，在类装载的时候被装载到内存，不自动进行销毁，会一直存在于内存中，直到VM关闭。

- 作用：

  如果一个方法与他所在类的实例对象无关，那么它就应该是静态的，静态方法不能滥用。

4.1.3 构造方法

- 格式：

```
public 类名()
{
    System.out.printf("构造方法");
}
```

- 运行时机：

构造方法是专门用来创建对象的方法，当我们通过关键字new来创建对象时，起始就是调用构造方法，如果没有编写构造方法,编译器会默认一个无参构造方法，一旦编写了构造方法，那么编译器将不再创建。

4.1.4 静态代码块（构造代码块、普通代码块）

- 格式

```
static{
        System.out.println("静态代码块");
}
```

- 执行时机

  静态代码块在类被加载的时候就运行了，而且只运行一次，并且优先于各种代码块以及构造函数

- 静态代码块的作用

  一般情况下，如果有些代码需要在项目启动的时候就执行，这时候就需要静态代码块。比如一个项目启动需要加载的很多配置文件等资源，我们就可以都放入静态代码块中。

4.1.5 几种方法执行顺序比较

```
public class Person {
    static {
        System.out.println("Person静态代码块1");
    }
    public Person() {
        System.out.println("Person构造函数2");
```

```
    }
    private int numB;
    public void test(int numA)
    {
        int numC;
        System.out.println(String.valueOf(numA));
        System.out.println(String.valueOf(numB));
        //System.out.printf(String.valueOf(numC));
    }
    public static void testA()
    {
        System.out.println("testA");
    }
}
```

```
public class Main {

    static {
        System.out.println("Main静态代码块3");
    }
    public Main() {
        System.out.println("Main构造函数4");
    }
    public static void main(String[] args) {

       Person p=  new Person();
       p.test(11);
    }
}
```

执行顺序：312，4是不会执行的

5、类、引用、实例和对象原理