**面向对象三大特征：**

1. 封装性：一是指把对象的属性和行为看成一个密不可分的整体，将两者封装在对象 之中；二是指信息隐蔽，把不需要让外界知道的信息隐蔽起来。
2. 继承性：被继承的类称为父类或超类，而经继承产生的类称为子类或派生类；根据 继承机制，子类继承了父类的所有成员，并相应地增加了自己的一些新 的成员。若一个子类只允许继承一个父类，称为单继承，若一个子类允许 继承多个父类，称为多继承。Java语言通过接口(interface)的方式来弥补 单继承的缺点。
3. 多态性：多态是允许程序中出现重名现象，Java语言中含有对象多态和方法重载两 种形式的多态。对象多态指的是子类对象与父类对象可以通过向上转型或 向下转型进行相互的转换，根据不同子类或父类实例化所完成的功能也不 同。方法重载是指在一个类中，允许多个方法使用同一个名字，只是方法 的参数个数或类型不同。

**类与对象的关系：**

类表示的是客观世界中某一群体的一些基本特征抽象。对象是表示一个个具体的东西。

例如，人是一类，拥有姓名，年龄等属性，而学生也是人。

**类的定义：**

类是由属性和方法组成的，属性就是变量，有时也称成员(Field)，方法是一些操作的行 为。

**对象的创建与访问：**

1. 声明对象开辟栈内存空间，栈内存空间存放的是堆内存空间的访问地址；实例化对 象开辟堆内存空间，堆内存空间保存对象的具体内容。
2. 一个对象要想被使用，则必须被实例化，否则会出现异常。

**封装性：**

封装性是指对外不可见，使用private关键字进行封装，封装后的属性必须通过setter 和getter方法设置和取得。对于类中的属性必须全部封装。

**构造方法：**

1）构造方法是一种特殊的方法，主要作用是为属性赋值并返回一个实例，当调用new 实例化对象时才会调用构造方法。

2）构造方法名与类名称一致，并且没有返回值。

3）只要是类就必须存在构造方法，如果一个类中没有明确声明一个构造方法则在 编译时会自动生成一个无参数的、什么都不做的构造方法，如果类中存在构造 方法则不会自动生成。即一个类中至少存在一个构造方法。

4）构造方法可以重载，在一个类中对构造方法重载时，所有的构造方法最好按照参数 个数从多到少或者从少到多进行排列。并且在一个类中，最好存在一个无参构造方 法和满参构造方法。

**匿名对象：**

匿名对象是指没有名字的对象，一般匿名对象只使用一次，并且只会在堆内存开辟 空间，不存在栈内存的引用。

**类的设计思路：**

1. 根据要求创建类并写出类所包含的属性；
2. 将所有的属性进行封装并添加setter和getter方法；
3. 根据需要添加若干构造方法；
4. 根据要求添加其他的方法；
5. 最好覆写Object类中的toString()、equals()、hashCode()方法；
6. 类中的方法都不要直接输出，而是交给被调用出输出。

**String类对象的实例化：**

String类对象有两种实例化的方式，一是直接赋值(String str=”hello”)，二是通过构造方 法赋值(String str=new String(“hello”)。需要明白的是，一个字符串就是一个匿名对象， 匿名对象是已经开辟堆内存空间的并可以直接拿来使用的对象。

1）对于直接赋值，只会开辟一块堆内存空间，并且会自动保存在对象池中，如果下次 通过直接赋值声明了同样的字符串，则它们会共享一块堆内存空间，不会开辟新的 内存空间。如果使用“==”进行比较其结果为true。

2）对于使用构造方法赋值，会开辟两块堆内存空间，因为一个字符串本身就是一个匿名对象，而匿名对象的实质本身是块堆内存空间。其中有一块会成为垃圾，并且不会自动入池。

所以，String类对象的实例化都是采用直接赋值的方式完成。

**String类对象内容的比较：**

对于基本数据类型可以通过“==”进行比较，“==”是用来进行地址值的比较的。而对 于字符串来讲，使用“==”比较两个字符串只是判断两个字符串是否使用着同一块堆内 存空间。而要想判断两个字符串的内容是否相等，需要依靠String类中的equals( )方 法。

**字符串的不可改变性：**

1. 字符串的内容一旦声明则不可改变
2. 一个String类对象的内容的改变实际上是通过内存地址的“断开-连接”变化完成的， 而本身字符串中的内容并没有任何的变化。

**String类中常用的方法：**

1. 字符串可以使用toCharArray( )方法变成一个字符数组，也可以使用String类的构造 方法把一个字符数组的全部或部分的变为一个字符串。
2. 字符串可以使用getBytes( )方法变为一个byte数组，也可以通过String类的构造方 法将一个字节数组全部或部分的变为字符串。
3. 可以直接使用String类中的CharAt( )方法取出字符串指定位置的字符。
4. 在String类中使用length( )方法取得字符串的长度。
5. 在String类中使用indexOf( )方法，可以返回指定字符串的位置，如果不存在则返回 -1。或者使用contains( )方法判断字符串是否存在，结果返回布尔值。
6. 在String类中提供了substring( )方法可以截取指定位置的字符串或截图从指定位置 到字符串结尾的字符串。
7. 在String类中通过split( )方法可以进行字符串的拆分操作，拆分的数据将以字符数 组的形式返回。但对于特殊的字符需要进行转义操作。
8. 可以使用toUpperCase( )方法和toLowerCase( )方法完成字符串大小写的转换。
9. 在String类中使用startsWith( )方法可以判断字符串是否以指定的字符串开头，或使 用endsWith( )方法可以判断字符串是否以指定的字符串结尾，结果返回布尔值。
10. 在String类中可以使用equals( )方法进行字符串内容的比较。如果要完成不区分字 母大小写的比较则可以使用equalsIgnoreCase( )方法进行比较。
11. 使用String类中的replaceAll( )方法，可以将字符串中指定的字符串进行替换。

**引用传递：**

1. 引用数据类型指的是一段堆内存空间可以同时被多个栈内存指向。
2. 所谓的引用传递，就是将一个堆内存空间的使用权交给多个栈内存空间。每个栈内 存空间都可以修改堆内存空间的内容。
3. 引用传递的本质就是在于不同的栈内存空间将指向同一块堆内存空间，并且都可以 对堆内存空间的数据进行修改。
4. 一个栈内存空间只能指向一个堆内存，而一个堆内存空间可以被多个栈内存指向。
5. 引用传递就是将堆内存空间的使用权交给多个栈内存空间，每个栈内存空间都可以 对堆内存空间修改。

（基本数据类型引用传递后原先变量的值并不会发生改变）

**字符串的引用传递：**

字符串的引用传递修改内容后，原先的字符串并不会发生改变，因为字符串的内容一旦

声明则不可改变，改变的只是其内存地址的指向。

但如果字符串是作为某个类中的属性存在的话，则引用传递后会发生改变。

**this关键字：**

1. 表示本类中的方法或属性。
2. 调用本类中的构造方法。使用this调用本类中的构造方法必须放在构造方法的第一 行，并且类中至少存在一个构造方法是不使用this调用其他构造方法的，否则将构 成调用死循环。一般将无参构造方法作为出口。
3. 表示当前对象。当前对象指的是正在调用类中方法的对象。

**对象的比较：**

要想判断两个对象是否相等，可以先使用“==”判断两个对象的引用地址是否相等，如 果相等，则对象相等。，然后判断两个对象的全部属性是否一一全部相等，如果全部一 一相等，则对象相等。

**static关键字：**

使用static关键字可以用来声明属性和方法。使用static关键字声明的属性和方法可以 由类名称直接调用，并且可共享。

1. 使用static关键字声明的属性称为全局属性或静态属性，可以被对象共享。
2. 使用static关键字声明的方法称为全局方法或静态方法，可以使用类名称直接调用， 但需要注意的是，非static方法可以调用static或非static方法，但static声明的方 法只能调用static声明的方法，不能调用非static声明的方法，即static类型的方法 中只能调用static类型的方法。（属性也是）

**Java常用的内存区域：**

1. 栈内存空间：保存引用的堆内存空间的地址
2. 堆内存空间：保存每个对象的具体属性内容
3. 全局数据区：保存static类型的属性
4. 全局代码区：保存所有的方法定义

**代码块：**

代码块是使用“{ }”括起来的一段代码，根据使用的位置不同，代码块分为4种：普通 代码块、构造代码块、静态代码块、同步代码块。

1. 普通代码块是指在语句或方法中定义的代码块，主要作用是进行程序分割。
2. 构造代码块是直接写在类中的代码块。构造代码块优先于构造方法执行，并且每次 实例化对象都会执行构造代码块。
3. 静态代码块是使用static声明的代码块，定义在类中。在主类中定义的静态代码块 优先于主方法执行，在类中定义的代码块优先于构造代码块和构造方法执行，但是， 静态代码块只执行一次。

**单例设计模式（Singleton）：**

使用private可以对属性或方法进行封装，在对方法的封装中也包含了对构造方法的封 装。但构造方法封装后，在其他类中就无法实例化该类对象。此时就可以该类的内部声 明一个public static类型的本类对象，然后通过类名称调用此对象给在其他类中的该类 对象实例化。

但是此对象也是作为属性存在的，所以也需要进行封装处理。这时就可以通过定义一个 static方法返回该属性。这种设计模式称为单例设计模式，即将类中的构造方法封装， 在类的内部实例化一个本类的对象作为本类的一个属性，然后通过调用static方法返 回该属性给其他类中的该类对象进行、实例化。只要将构造方法实例化了，就可以 控制实例化对象的产生。

**对象数组：**

1. 对象数组是指包含了一组相同类型的对象，对象数组中的每一元素都是一个对象。
2. 对象数组中的每一个对象都必须分别进行实例化操作，并且对象数组也分静态初始 化和动态初始化。

**内部类：**

内部类是指在类的内部声明的类，例如在A类中定义B类，则B类称为内部类，A类称 为外部类。

1. 在内部类中可以方便的访问外部类中的私有属性，在外部类中也可以实例化内部 类对象访问内部类中的私有属性。
2. 使用static声明的内部类不能访问外部类中的非static属性或方法。
3. 一个内部类除了可以通过外部类访问，也可以在其他类中进行调用，其语法格式为：

**外部类.内部类 内部类对象=外部类实例.new 内部类( );**

1. 在方法中也可以定义内部类。

内部类定义后，生产的class文件是以”Outer$Inner.class”的形式存在的。