复习：

1. static：可以修饰变量、方法、内部类、代码块。

静态变量---随着类的加载而加载到方法区中的静态区，并赋予了默认值。静态变量先于对象存在。所有的对象存储的静态变量所对应的地址

可以用this调用静态变量吗？---可以

静态方法---在类加载的时候加载到了方法区中的静态区，只是存储在了静态区没有执行，在方法被调用的时候到栈内存中执行。静态方法先于对象存在。静态可以重载，不可以被重写，但是在父子类中可以存在方法签名完全一致的静态方法(隐藏)。---如果父子类中存在了方法签名完全一致的方法，要么都是静态，要么都是非静态。

可以在静态方法中使用this吗？---不能

可以在静态方法中直接调用本类的非静态方法吗？---不可以

public void m1(){}

public static void m2(){

m1(); // 错--- 在Java中，所有的非静态方法和非静态属性都需要通过对象来调用

}

静态代码块---在类第一次被真正使用的时候---创建对象，调用方法

执行顺序：父类静态-子类静态-父类构造-子类构造

2. final：修饰数据、方法、类

常量---定义好之后值不可改变---基本类型是实际值不可变，引用类型是地址不可变，其中的属性值或者元素值可以改变的。---如果定义了一个成员常量，需要在对象创建完成之前给值；如果是静态常量，需要在类加载完成之前给值。

最终方法---不能被重写---可以被重载，可以被继承

子类可以继承父类的所有的属性和方法，有一部分父类的功能对子类不可见

子类通过继承可以使用父类中一部分的方法和属性

最终类---不能被继承

3. abstract：修饰类、方法

抽象类---不能被实例化，但是有构造方法，抽象类中可以定义一切属性和方法；子类在继承了抽象类之后必须重写抽象类中的所有抽象方法，除非这个子类也是抽象类。抽象类一定不是最终类

抽象方法---可以被重载，必然被重写---static/final/private不能修饰抽象方法。如果一个抽象方法是使用了默认权限修饰符来修饰的，那么这个时候要求父子类同包。

4. interface：用implements关键字与类产生实现关系---多实现。接口可以继承多个接口---类实现了接口之后必须重写接口中的所有的抽象方法。接口不能定义构造方法。---接口中所有的方法都是抽象方法，默认用public abstract修饰。接口中定义的属性默认是用public static final修饰

## 内部类

### 方法内部类

定义在方法中的类

Outer1$1$Inner1.class---外部类$编号$内部类---习惯上，如果class文件名中出现了$,一般会认为这是一个内部类

### 成员内部类

定义在类内方法外的类

Outer2.Inner2 oi2 = new Outer2().new Inner2();

### 静态内部类

用static修饰的内部类

Outer3.Inner3 oi3 = new Outer3.Inner3();

### 匿名内部类

没有名字的内部类

任何一个可以被继承的类以及任何一个接口都可以存在匿名内部类的形式---匿名内部类本质上是继承了对应的类或者是实现了对应的接口

## 包

用package声明包，用import导包

\* 表示一个通配符---导入当前包下的所有的类，但是不包括子包下的类---不推荐使用

java---Java小组提供的原生包

javax---Java的扩展包

org---第三方厂商提供的包

java.lang---核心包---包含了Java程序要运行的基本类---在JVM启动的时候就自动加载到内存中了---所以这个包中的所有的类在使用的时候不需要手动导包

考虑：什么情况下可以不用手动导包？---java.lang， 同包类

java.util---工具包---提供了很多便捷的操作数据的类

java.math---和数学运算有关的包

java.text---和格式化有关的类

java.io---和数据传输有关的包

java.sql---对数据库进行操作的包

java.net---和网络通信有关的包

java.security---和数据安全有关的包

java.nio---支持高并发的包

java.applet---GUI操作相关的包---现在几乎不用

javax.swing---GUI的扩展包

## 垃圾分代回收机制

回收针对的是堆内存。

将堆内存划分为了2块：新生代和老生代。新生代有划分为伊甸园区和幸存区。对象刚创建的时候是放在新生代的伊甸园区，在伊甸园区经过一次回收之后如果这个对象依然存在，则会挪到幸存区。如果在幸存区经过多次回收，会挪到老生代。老生代的扫描频率是远低于新生代。老生代的对象一旦回收，往往会导致程序的卡顿甚至于崩溃。

发生在新生代的回收---初代回收(minor gc --- garbage collector)

发生在老生代的回收---完全回收(full gc)

如果一个对象在新生代存放不开，会试图直接放入老生代---OutOfMemoryError

### 扩展：类加载机制

加载（将class文件转换为字节码放入内存）---校验（检查这段字节码是否合法）---准备（为所有的类变量分配空间并默认初始化）---解析（引用替换过程）---初始化（类变量赋值，静态代码块执行）

类加载器---ClassLoader

启动/引导类加载器 --- Bootstrap ClassLoader---加载Java的原生类

扩展类加载器 --- Extension ClassLoader --- 加载扩展类

应用类加载器 --- Application ClassLoader --- 加载自定义类

在Java中，加载类的时候双亲委派机制

API --- Application Programming Interfaces --- 应用程序接口

Object String Pattern 数学类 包装类 日期类

异常 集合 泛型 映射

File IO 线程 套接字 反射 注解

JDK1.8的特性，（有时间就扩展）jvm的运行参数