复习：

1. static

静态变量---在类加载的时候加载到方法区，并且在方法区中赋予了默认值。静态变量是先于对象存在，因此习惯上是通过类名来调用。所有的对象存的是静态变量的地址，所以它是被共享的。

静态方法---随着类的加载而加载到方法区，只是存储在方法区并不执行，在方法被调用的时候到栈内存中执行。静态方法也是先于对象存在，所以一般是通过类名来调用静态方法。静态方法可以重载，可以继承，但是不能重写，然而父子类中可以存在方法签名一致的静态方法---隐藏

静态代码块---在类加载的时候执行一次 --- 执行顺序

2. final

常量---定义好之后值不可变

最终方法 --- 可以重载，可以继承，但是不能被重写

最终类 --- 不能被继承

## abstract

如果一个类的所有的子类都重写了父类的某个方法，那么可以把父类的这个方法声明为一个抽象方法，抽象方法所在的类必须是抽象类。

抽象类是不能用Java语言创建对象

抽象类中可以存在实体方法，也可以定义属性。

抽象类中不一定有抽象方法。

子类在继承抽象类之后必须重写其中的抽象方法，除非子类也是抽象类

抽象方法是否可以重载？---可以

抽象方法是否可以用static/final修饰？---不可以

抽象方法是否可以是private的？---不可以

如果抽想方法用的是默认权限，那么对子类有什么要求？---父子类必须同包

抽象类中的构造方法能否私有化？---可以

抽象类能否用final修饰？ --- 不可以

## 接口

用interface定义一个接口，利用implements让类来实现一个接口。

类在实现接口之后必须重写其中的所有的抽象方法

接口中只能定义抽象方法。

接口中不能定义构造方法 --- 接口不是类

在Java，支持单继承，多实现。 --- 一个类只能继承一个父类，一个类可以实现多个接口

接口之间也可以继承，接口之间是多继承

interface A {

public int m();

}

interface B {

public int m();

}

class C implements A, B {

public int m(){} // 如果一个类实现了多个接口，并且这多个接口中存在了方法签名一致的方法就可能会导致方法的重写产生歧义

}

接口中的属性默认用public static final修饰

练习：定义一个接口表示形状(Shape)，提供两个方法：周长和面积。给这个接口提供三个实现类：矩形(Rectangle),圆形(Circle),三角形(Triangle),给矩形提供一个子类正方形(Square)

圆形：周长2πr 面积πr2 r表示半径

三角形：周长a+b+c 面积√(p(p-a)(p-b)(p-c)) p=(a + b + c)/2 --- Math.sqrt(); Math.pow(a,0.5)

## 内部类/嵌套类

### 方法内部类

定义在方法中的类。

// 定义在方法中的类---方法内部类/局部内部类

// 只能在定义它的方法中使用

// 方法内部类中不能定义静态变量和静态方法，可以定义静态常量

// 方法内部类可以使用外部类的一切属性和方法

// 方法内部类不能使用当前方法中的局部变量,但是可以使用当前方法中的局部常量

// 可以继承别的类可以实现别的接口

// 可以使用final或者是abstract来修饰

class Inner1 {

int i = 5;

static final int j = 7;

public void m() {

System.out.println("Inner m");

System.out.println(j);

System.out.println(oi);

System.out.println(m);

}

}

### 成员内部类

定义在类内方法外的类

总结：

可以修饰类的修饰符：public、abstract、final、默认

可以修饰方法的修饰符：权限修饰符、static、final、abstract、strictfp、synchronized、native

可以修饰属性的修饰符：权限修饰符、static、final、volatile

### 静态内部类

用static修饰的内部类

总结：

1. 除了静态内部类以外，其余的内部类中只能定义静态常量而不能定义静态变量和静态方法

2. 除了静态内部类以外，其余的内部类都可以使用当前外部类中的非静态属性和非静态方法

### 匿名内部类

只要一个类可以被继承，那么就可以存在匿名内部类形式

任何一个接口都一定可以存在匿名内部类形式

扩展：在Java中，类中可以定义类，类中定义接口。接口中可以类，接口中定义接口。

类中定义的接口以及接口中定义的类或者是接口都默认是静态的。

## 包

定义包用的package，导入包用的import

package语句必须放在整个Java文件首行。

\* 表示通配符，可以导入当前包下的所有的类而不包括子包下的类

java --- 原生包

java.lang --- 核心包，包含了一个Java程序能够运行的基本类。程序在启动的时候会首先默认加载核心包，所以核心包中的类在使用的时候不需要导包。

java.util --- 工具包，提供了很多的能够便捷的操作数据的方式

java.applet --- GUI --- 图形用户信息 --- 界面

java.math --- 数学运算

java.sql --- 和数据库交互的包

java.io --- 用于数据的传输

java.nio --- 高并发

java.net --- 网络通信

java.text --- 格式化

javax --- 扩展包

org --- 第三方厂商提供的包

## 垃圾分代回收机制

垃圾回收针对的是堆内存。

在Java中内存是自动回收和管理的机制。Java为了防止内存问题，所以将各种数据类型的大小固定，也因此导致在Java中一旦出现内存问题，程序员无法处理。在Java中，所有的数据使用完成之后由Java自动管理释放。

对象在堆内存中存储，用完之后由GC(garbage collector)回收。GC回收时间是不确定的。 --- 对象用完之后不一定会立即释放，因为GC可能没有启动。--- 当堆内存的使用率超过70%的时候，GC会启动回收

堆内存划分为：新生代和老生代。新生代划分为伊甸园区和幸存区。一个对象刚创建的时候放入伊甸园区，在伊甸园区经过依次扫描，如果这个对象依然存在，则挪到幸存区。在幸存区中经过多次扫描，如果依然存在则挪到老生代。老生代的扫描频率要远低于新生代。所以老生代的对象如果产生回收可能会导致程序的卡顿甚至于崩溃。

发生在新生代的回收称之为初代回收 --- minor gc

发生在老生代的回收称之为完全回收 --- full gc