复习：

1. 多态

A. 编译时多态：方法的重载

B. 运行时多态：向上造型、方法的重写

C. 如果使用了向上造型来创建一个对象，那么这个对象能够调用什么方法看的是父类，如何执行看的是实现类

2. 静态

A. 静态变量：随着类的加载而加载到方法区，并且方法区被赋予默认值。静态变量先于对象出现，所以习惯上是通过类名来调用。所有的对象存的是这个变量的地址，所以他是被共享的。

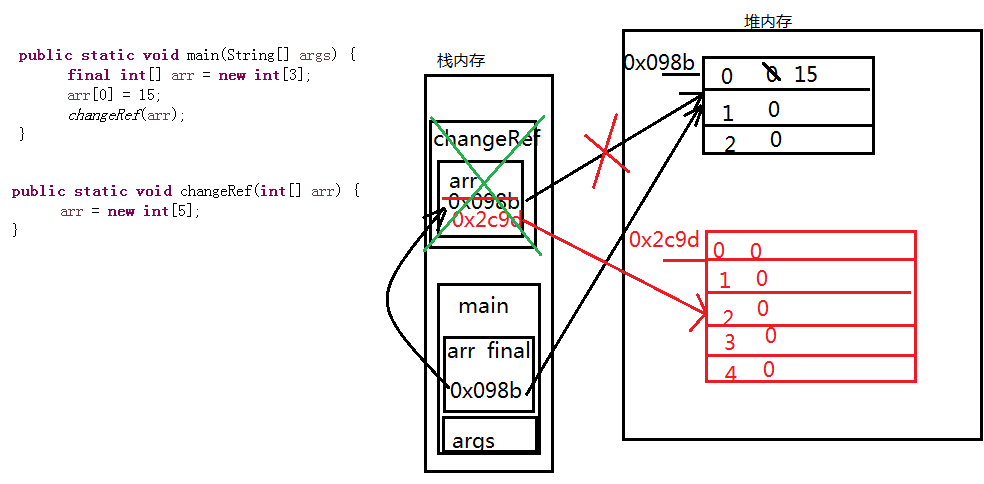
B. 静态方法：在类加载的时候加载到方法区，存储在方法区而没有执行，被调用的时候去栈内存中执行。习惯上是通过类名来调用静态方法。静态方法可以重载，可以被继承，但是不能重写，而在父子类中可以存在方法签名一致的静态方法，此时构成的是隐藏。静态方法中不能使用this/super，也因此不能直接在静态方法中使用本类的非静态方法。---只要父子类中存在了方法签名一致的方法，要么都是静态，要么都是非静态。

C. 静态代码块：在类第一次被真正使用的时候执行。---执行顺序：父类静态->子类静态->父类构造->子类构造

## final

用于数据、方法、类

常量---定义好之后值不可变的。对于基本类型而言值得是实际值不可变；对于引用类型而言，指的是地址不可变。常量属性必须在对象创建完成之前给值。静态常量要求在类加载完成之前给值



public class Test {

private final int i ;

{i = 7;}

public Test(){}

public Test(int i){this.i = i;} // 报错，二次赋值

}

Test t = new Test(5);

final修饰方法称之为最终方法---不能被重写/隐藏。可以被重载，可以被继承

final修饰类称之为最终类---不能被继承

注意：子类会继承父类全部的方法和属性，但是部分的方法和属性被继承过来之后不可见

## abstract

如果所有的子类对父类中的某个方法都做了重写，那么此时这个方法可以省略掉方法体不写，用abstract修饰，此时这个方法称之为抽象方法。抽象方法所在的类一定是抽象类。

抽象类不能被java语言实例化，有构造方法，底层会以C语言的形式来创建实例。

抽象类中可以定义一切属性和方法

抽象类中可以有抽象方法，也可以没有

子类继承了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

抽象类一定不能被final修饰

抽象方法可以重载

抽象方法不能被static/final/private修饰

如果抽象方法是使用默认权限，那么对子类有什么要求？---同包

练习：定义一个类表示形状(Shape),定义两个方法求周长(girth)和面积(area)， 然后提供两个子类---矩形(Rectangle)和圆形(Circle)， 矩形有一个子类表示正方形(Square)

## interface

用interface定义接口---接口中只能定义抽象方法（jdk1.8以前）

接口不能创建对象，也没有构造方法

利用implements关键字让类去实现一个接口

在java中，类和接口之间是多实现的关系。---多实现在一定程度上会导致重写混乱

interface A {

public abstract String m();

}

interface B {

public abstract void m();

}

class C implements A, B {

public void m(){}

public String m(){}

}

C c = new C();

c.m();

接口与接口之间是多继承的

interface A {

void m(); // 默认用public abstract修饰

}

class B implements A {

public void m(){}

}

总结：

能够用于修饰数据的修饰符：static、final、权限修饰符、volatile

能够用于修饰类的修饰符:pulic/默认、final、abstract

能够用于修饰方法的修饰符：权限修饰符、static、final、abstract、strictfp、native、synchronized、default

接口中的属性默认是用public static final修饰

## 内部类

### 方法内部类

定义在方法中的类---能够使一段或者多段逻辑被重复利用，并且保证这段逻辑只能属于某一个方法而不会被其他方法使用

### 成员内部类

定义在类内方法外。

Outer2.Inner2 oi2 = new Outer2().new Inner2(); --- 表示在类外创建内部类对象