Java

1. 局部变量与成员变量

Java类中不会给一个局部变量默认赋值（方法中定义的变量），java类只会默认的给成员变量默认的附加初始值，一般为0。

当成员变量与局部变量重名时，只会调用局部变量，局部变量会覆盖成员变量的值

2.构造方法（有参数和无参数）

使用new+构造方法 创建一个对象

定义在java类中一个用来初始化对象的方法

与类同名且无返回值

3.静态成员（static）

它属于整个类，而不是某个对象所有，被类的所有对象共享，静态成员可以使用类名直接访问，也可以使用对象名进行访问（一般用类名进行访问）。

4.静态方法（static）

与静态变量一样，也可以使用static修饰方法，称为静态方法或者静态类

静态方法中可以直接调用同类中的静态成员，但是不能直接调用非静态成员

如果过想在静态方法中调用非静态成员，那就必须创建对象，通过对象进行访问

在普通成员方法中，则可以直接访问同类的静态变量和非静态变量

静态方法中不能直接调用非静态方法，需要通过对象来访问非静态方法

5.静态初始化块(static)

Java中可以通过初始化块进行数据赋值

静态初始化块只在类加载时执行，且只会执行一次，同时静态初始化块只能给静态变量赋值，不能初始化普通的成员变量

6. 初始化块:

在类的声明中，可以包含多个初始化块，当创建类的实例时，就会依次执行这些代码块

总结：程序运行时，静态初始化块最先被执行，然后执行普通初始化块，最后才执行构造方法。由于静态初始化块只在类加载时执行一次，所以当再此创建对象时并未执行静态初始化块

7.封装

将类的某些信息隐藏在类的内部，不允许外部程序直接访问，而是通过该类提供的方法来实现对隐藏信息的操作和访问

好处：

1. 只能同通过规定的方法访问数据

2.隐藏类的实例细节，方便修改和实现

封装的实现步骤：

1. 修改属性的可见性（public private protected）

2. 创建getter/setter方发（属于属性的读写）

3.在get和set方法中加入属性的控制语句（对属性值的合法性进行约束）

8.java中的访问修饰符

修饰符 本类 同包 子类 其他(包外)

Private yes no no no

Public yes yes yes yes

Protected yes yes yes no

默认 yes yes no no

9.this关键字

This关键字类似于当前类的对象，可以调用类中的属性或值

10.内部类

内部类就是定义在类的内部的类，与之对应，包含内部类的类被称为外部类

作用：

提供了更好的封装，可以把内部类隐藏在外部类之内，不允许同一个包中的其他类进行访问

内部类的方法可以直接访问外部类的所有数据，包括私有数据(**直接给数据赋值，不需要通过对象访问然后赋值**)

内部类所实现的功能，使用外部类同样可以实现，只是有时使用内部类更加方便

内部类：

1.成员内部类

定义了成员内部类之后，必须使用外部类对象创建内部类的对象，二不能直接去new一个内部类的对象

外部类是不能直接使用内部类的成员和方法:可以先创建内部类的对象，然后通过内部类的对象来访问其成员变量和方法

2.静态内部类

不能直接访问外部类的非静态成员，但是可以通过new外部类().成员的方式访问

如果外部类的静态成员与内部类的静态成员名称相同，可通过 类名.静态成员 访问外部类的静态成员；

如果外部类的静态成员与内部类的静态成员名称不相同，可通过成员名直接访问外部类的静态成员；

3.方法内部类

内部类定义在外部类的方法之中，方法内部类只在该方法的内部可见，即在该方法内可以使用

**不能在外部类的方法以外使用，因此方法内部类不能使用访问控制符和static修饰**

匿名内部类

11.继承

继承的好处：子类拥有父类的所有属性和方法（private修饰的无效）

实现代码的复用

**//注意修饰符为private的属性不可以继承**

格式：…..extends………

继承的初始化顺序：初始化父类再初始化子类

先执行初始化对象中的属性，再执行构造方法中的初始化

12.final关键字

可以修饰类，属性，方法，变量

1.修饰类：该类不可以被继承

2.修饰方法：该方法不允许被覆盖(重写)

3.修饰属性：该属性不会进行隐式的初始化(类的初始化属性必须有值)（该类属性不会自动初始化）

或在构造方法中赋值

4.修饰变量：该变量的值只能被赋值一次，即变为常量

13.super关键字

在对象的内部使用，可以代表父类对象

1.访问父类的属性

super.age;

2.访问父类的方法

Super.eat();