

Continue：跳出本次循环，继续下次循环

Break：终止当前循环，继续执行后面的代码

栈：保存局部变量的值，包括：1、用来保存基本的数据类型的值；2、保存类的实例即堆区对象的引用（指针）。也可以用来保存加载方法时的帧。

堆：用来存放动态产生的数据，比如new出来的对象。注意创建出来的对象只包含属于各自的成员变量，并不包括成员方法。因为同一个类的对象拥有各自的成员变量，存储在各自堆中，但是他们共享该类的方法，并不是每创建一个对象就把成员方法复制一次。

常量池：JVM为每个已加载的类型维护一个常量池，常量池就是这个类型用到的常量的一个有序集合。包括直接常量（基本类型，string）和对其他类型、方法、字段的符号引用。池中的数据和数组一样通过索引访问。由于常量池包含了一个类型所有的对其他类型、方法、字段的符号引用，所有常量池在java的动态链接中起了核心作用。常量池存在于堆中。

代码段：用来存放从硬盘上读取的源程序代码

数据段：用来存放static定义的静态成员

面向对象：

如果需要调用一个类中的属性和方法，必先创建该类的对象。

构造函数：

1. 方法名和类名相同
2. 没有返回值类型
3. 总是和new关键字一起使用，用来对象初始化
4. 构造函数是可以有任意多个任意的类型的参数 注意： 如果将代参的构造函数显示的写出那么默认的无参的构造函数就不存在了
5. 构造函数可以有一个或者多个。如果没有显示的写构造函数，那么将存在一个默认无参构造函数

构造函数重载：

构造函数的方法名相同，参数列表不同（个数，类型，顺序）

根据传入的参数列表来决定调用的是哪一个构造函数

方法重载：

在同一个类中，方法名相同，传入的参数列表不同（参数个数，类型，顺序），与返回值无关

面向对象的三大特征:封装、继承、多态

封装：将设计者和使用者分开，从而保护数据不被破坏

通过private关键字对属性进行私有化，如果需要访问数据，那么需要另外提供公用方法

继承：

* 一个类继承另外一个类，被继承的类称之为父类或者基类或超类，继承的类叫做子类或者派生类
* 子类继承父类，将继承父类的所有属性和方法包含私有，但是不能访问，其他的都可以直接访问，
* 子类得到构造过程中必须调用分类的构造方法
* 只能是直接的单继承，但是可以间接继承

减少代码的冗余，提高代码的重复利用性

是方法重写的前提

多态：一种表现形似，多种形态叫做多态

多态的种类：

1. 编译时多态 如方法重载
2. 运行时多态 如方法重写 1）要有继承2）要有重写3）父类对子类对象的应用

方法重写

在继承的关系中的，子类与父类的方法名、返回值类型、参数列表相同

Super：

1. 子类调用父类的方法、变量
2. 如果是在构造方法中使用super，那么表示调用父类的构造方法，必须是第一行
3. 必须在子类构造方法中才可以调用父类的构造方法

This：

1. 调用当前类中的方法
2. 如果是调用构造方法，那么必在构造方法中调用
3. 与super不能同时显示调用

设计模式之工厂模式：作用以及目的是为了解耦（耦合度）

简单工厂模式

工厂方法模式

抽象工厂模式

抽象类

Abstract修饰的类叫做抽象类

抽象 抽取了子类和父类共有的属性和方法

抽象类和普通类的区别

抽象类可以有实例变量 实例方法 可以有抽象方法

抽象类不能被实例化，需要通过子类进行实例化

抽象方法

有abstract关键字修饰的方法，没有方法体，只有方法声明，他就是用来被重写的

如果子类继承父类，那么子类必须重写父类的抽象方法

Static：静态关键字(变量名大写，“\_”符链接)

Static修饰属性：属于类而不是属于某个成员。直接通过类名来调用

由static修饰的变量叫类变量。在使用静态变量时往往直接通过类名来进行调用，而不建议使用成员来调用

静态方法中只能访问静态变量和静态方法

静态代码块 代码块 静态变量 静态方法 构造方法执行的优先级

静态修饰的东西(按照书写顺序执行)>普通代码块>构造函数

接口

一个特殊的类，类中的所有变量都是 最终的 静态的 变量，所有的方法都是公共的抽象方法

接口就是用来被实现的，子类必须重写父类接口中的所有抽象方法

接口可以继承接口，如果一个类实现了接口，必须实例化接口以及所有的抽象方法

抽象类可以实现接口，可以实例化接口中的抽象方法，也可以不实例化。如果不实例化的话将交给后面的类来执行

接口不能继承抽象类。接口不能实例化，只能通过子类实现

接口和抽象类和普通类的区别以及关系

1. 普通类由实例变量 实例方法
2. 抽象类可以有实例变量，实例方法，也可以有抽象方法。抽象类只能被单继承可以简接继承
3. 接口中的所有变量都是最终的 静态的，所有方法都是抽象方法
4. 接口和抽象类都不能实例化，需要通过子类来进行实例化
5. 抽象类可以实现接口，不一定要实现接口中的抽象方法，接口可以多实现

Package

作用就是用来区分管理文件，体现为文件夹

创建一个新的java文件 在第一行会显示 包的路径

在同一个包路径下面的文件之间相互可以直接访问

1. 在调用这个类的时候自动生成
2. 手动导入 \*
3. 快捷键 shift + ctrl + o （阻止导入）

访问控制符：用来指定被修饰的内容的被访问权限

Public（公共的）：

可以修饰类、变量、方法，只要在同一个工程下public所修饰的类、变量、方法任何类中都可以访问

Protect（保护的）：

可以修饰变量、方法，同包/不同包的只有在继承的情况下子类才可以访问

默认的：

可以修饰的类、变量、方法，同一个包可以访问

Private：

可以修饰变量、方法，同一个类中可以访问

在继承的方法重写关系中：重写的方法限制性不能高于父类

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 类内部 | 同一个包 | 子类 | 任何 |
| Private | √ |  |  |  |
| 默认的 | √ | √ |  |  |
| Protect | √ | √ | √ |  |
| Public | √ | √ | √ | √ |

常用系统类

String <==>StringBuffer

String：存在于常量池，字符串不可变，如果对字符串进行改变那么将返回一个新的字符串，而字符串本身不会变

StringBuffer：字符串缓存类，可变长度的字符串，对它的操作就是对内存的操作，字符串本身和返回的结果一样

性能比较，字符串拼接的方法测试 StringBuffer的性能要比string的性能好，StringBuffer（append）要比String（+）性能好

Math：提供算术的方法，都是静态方法

Random：随机数

System：

Date：日期类

Object：对象类 所有类的父类

（java的垃圾回收机制，当外面创建一个类这个类会是object的子类，会监控类的使用状态，当对象==null的时候或者使用完之后不再使用的情况下对象就会被垃圾回收机制回收

造成内存溢出的几种可能性

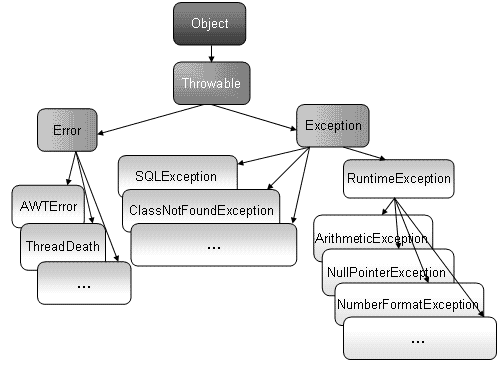
1. 不停的频繁的申请内存空间
2. 静态修饰的数据尽量不要频繁使用

）

包装类（提供了对于基本数据类型的操作方法）

字符串可以转成格式符合对应数据类型的 数据类型

Java的异常



1. 区分异常和错误

错误：语法问题

异常：程序可以通过编译 在执行的过程中发生问题，异常一旦发生将控制权交给了JVM

1. 异常的分类

运行时异常：代码编译没有问题，在程序执行的时候可能由于用户的操作造成，又叫做非检查异常 是RuntimeException和RuntimeException的子类

算术异常

空指针异常

检查异常：程序本身没有问题，由于外部原因导致（大部分是外部jar包）

ClassNotFoundException sqlException

比如数据库连接 需要依赖与第三方jar包ojdbc14.jar

1. 异常处理

1）捕获：

try{

需要处理异常的代码

}catch（exception e）{

发生异常后执行的部分

}

finally{

无论程序是否正确执行，都要执行的部分

}

将需要处理异常的代码段放到try中。如果异常发生程序将执行到对应异常种类的catch块，本身不解决异常，只是对异常的发生进行提示

2）throws 抛出 在方法后通过 throws 关键字抛出对应的异常，本身不解决异常，也不告诉你异常，而是将异常的处理交给了后面的调用