Java vs. C++

http://zhidao.baidu.com/question/2135937.html

When to use, scenarios

What is polymorphism,

So polymorphism is the ability (in programming) to present the same interface for differing underlying forms (data types).

For example, given a base class shape, polymorphism enables the programmer to define different area methods for any number of derived classes, such as circles, rectangles and triangles.

Ability to process objects differently depending on their [data type](http://www.webopedia.com/TERM/D/data_type.html) or[class](http://www.webopedia.com/TERM/C/class.html)..

Overload vs override,

Overload, just two method with the same name, but input, output, functionality is different,

Override, overwrite the method of sub class when inherited from super class.

Static key word etc. ,不用初始化对象的方法 或者变量，static表示不要实例化就可以使用， 不用consturct，initialize一次就一直用

http://developer.51cto.com/art/201204/332155.htm

static表示“全局”或者“静态”的意思，用来修饰成员变量和成员方法，也可以形成静态static代码块，但是Java语言中没有全局变量的概念。

被static修饰的成员变量和成员方法独立于该类的任何对象。也就是说，它不依赖类特定的实例，被类的所有实例共享。 在 stack上面而不是heap上面， instance vaiable 在heap上，member variable指初始化之前

用public修饰的static成员变量和成员方法本质是全局变量和全局方法，当声明它类的对象市，不生成static变量的副本，而是类的所有实例共享同一个static变量。

**2、静态方法**

静态方法可以直接通过类名调用，任何的实例也都可以调用，

因此静态方法中不能用this和super关键字，不能直接访问所属类的实例变量和实例方法(就是不带static的成员变量和成员成员方法)，只能访问所属类的静态成员变量和成员方法。

因为实例成员与特定的对象关联！这个需要去理解，想明白其中的道理，不是记忆！！！

因为static方法独立于任何实例，因此static方法必须被实现，而不能是抽象的abstract。

例如为了方便方法的调用，Java API中的Math类中所有的方法都是静态的，而一般类内部的static方法也是方便其它类对该方法的调用。

静态方法是类内部的一类特殊方法，只有在需要时才将对应的方法声明成静态的，一个类内部的方法一般都是非静态的

就行String.valueOf, Integer.xxxx

**static代码块**

static代码块也叫静态代码块，是在类中独立于类成员的static语句块，可以有多个，位置可以随便放，它不在任何的方法体内，JVM加载类时会执行这些静态的代码块，如果static代码块有多个，JVM将按照它们在类中出现的先后顺序依次执行它们，每个代码块只会被执行一次。例如：

，都一个static的main方法，这样JVM在运行main方法的时候可以直接调用而不用创建实例。

static final用来修饰成员变量和成员方法，可简单理解为“全局常量”！

encapsulate: wraps the data and function into single private unit, keep it safe from external manipulation.

FINAL, FINALLY

# final

final can be used to mark a variable "unchangeable"

private final String name = "foo"; //the reference name can never change

final can also make a method not "overrideable"

public final String toString() { return "NULL"; }

final can also make a class not "inheritable". i.e. the class can not be subclassed.

public final class finalClass {...}

public class classNotAllowed extends finalClass {...} // Not allowed

# finally

# 一般用来close or release resource

finally is used in a try/catch statement to [execute code "always"](http://docs.oracle.com/javase/tutorial/essential/exceptions/finally.html)

lock.lock();

try {

//do stuff

} catch (SomeException se) {

//handle se

} finally {

lock.unlock(); //always executed, even if Exception or Error or se

}

Java 7 has a [new try with resources statement](http://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html) that you can use to automatically close resources that explicitly or implicitly implement [java.io.Closeable](http://docs.oracle.com/javase/7/docs/api/java/io/Closeable.html) or [java.lang.AutoCloseable](http://docs.oracle.com/javase/7/docs/api/java/lang/AutoCloseable.html)

# finalize

finalize is called when an object is garbage collected. You rarely need to override it. An example:

protected void finalize() {

//free resources (e.g. unallocate memory)

super.finalize();

}

Jvm一遍自己完成，这个是手动回收

工程模式

单例模式