# Java基础

## 类型长度：

byte：1

boolean：1

short：2

char：2

int：4

float：4

long：8

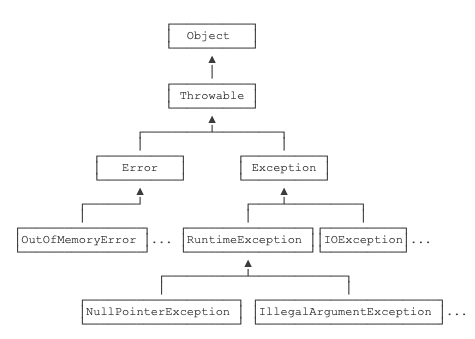
double：8

float：需要加f

long：需要加L

常量：加final

## 异常：



子线程的异常处理：

<https://www.cnblogs.com/jpfss/p/10272066.html>

1:在子线程中正确处理异常

2:给线程（所有的线程）设置默认的异常处理器

3:使用ExecuterService.submit提交任务，future.get会抛出子线程的异常

## String：

### String/StringBuffer/StringBuilder:

首先：他们三个都是final的，不可以被继承

String是不可变的

StringBuffer和StringBuilder是可变的

StringBuffer：线程安全

StringBuilder：非线程安全

String每次在对String进行改变的时候都会生成一个新的对象

而StringBuffer和StringBuilder不会生成新的对象

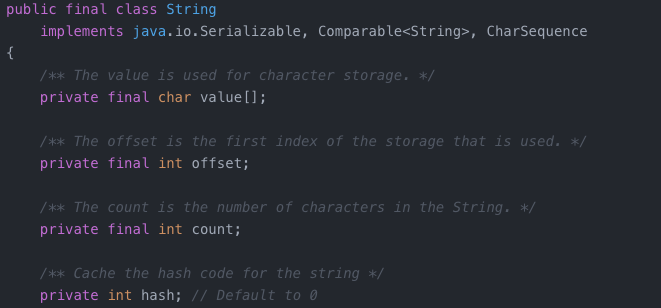
所以：经常改变的场景使用StringBuffer/StringBuilder，不改变的场景使用String

### String为啥不可变：

<https://www.zhihu.com/question/20618891>

String使用final修改，语言层面上限制String不可变

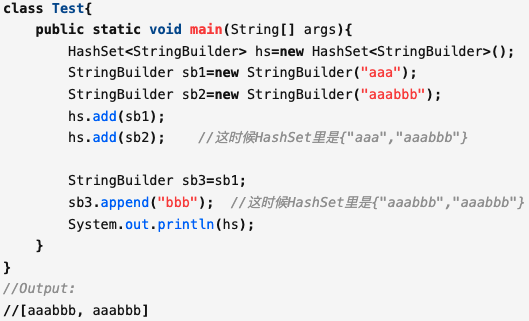
String底层使用char数组实现的，其实通过反射也是可以改变内容的。



设计成不可变的思考：

1:String 在java中大量被使用，设计成不可变的常量，可以节约空间

2:在安全方便，比如我们绝大多数场景使用String作为key使用HashSet，如果可变，那么会出现如下不可预知的错误。



### String创建多少个对象的问题：

<https://cloud.tencent.com/developer/article/1686226>

前提：字符串常量池中没有 adc这个常量

#### 1:new String(“123”)场景



会创建2个对象

一个放到了常量池里

一个在堆中

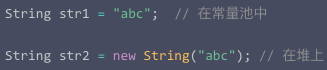


会创建3个对象

一个放到常量池中

2个在堆中

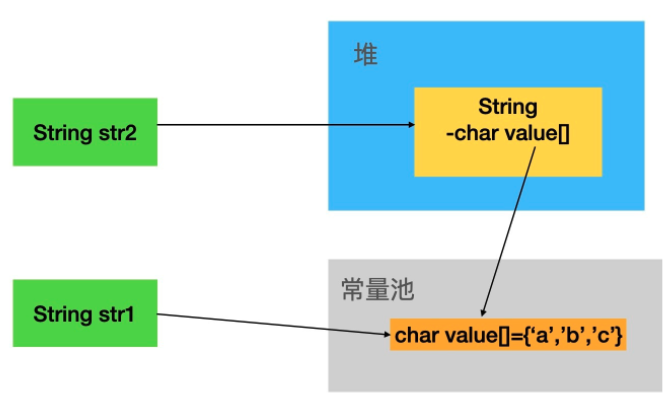
原理：

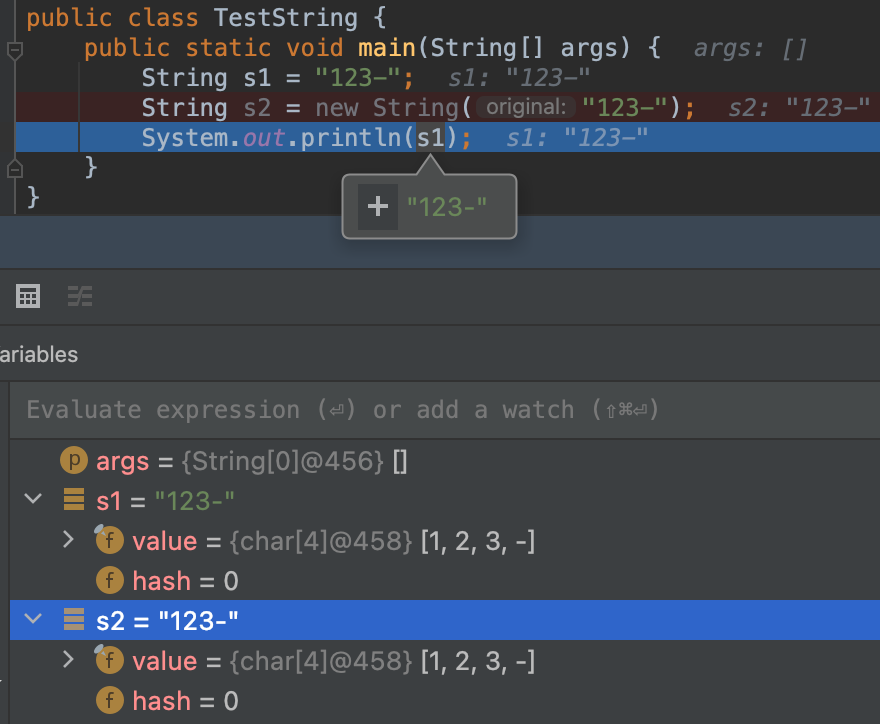


1:上面这个例子，str1在常量池中创建一个 String常量（下图的常量池应该外层有一个String对象的）

2:str2在new的时候，先查看常量池是否存在abc常量

3:如果有那么在堆中创建String对象，chars指向常量池中的cahrs





#### String a = “a” + “bc”问题：

这个等于 String s = “abc”只会在常量池创建一个对象

#### String a = “a” + new String(“bc”)问题：

创建5个

1:a（常量池）1个

2:bc（常量池+堆）2个

3:abc（堆）1个

4:StringBuffer 辅助类

#### s.intern()：

把s如果在堆中，那么返回常量池中对应的String 对象

## finally return问题：

<https://www.jianshu.com/p/ebb13ac70f5d>

结论：所有的场景都以最后一个执行return 的为准。

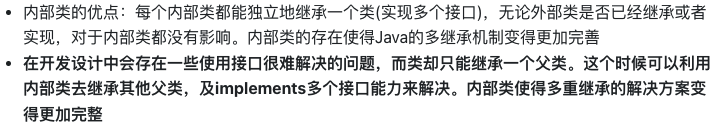
执行顺序，try/catch/finally

所以：return并不是方法的最终出口，如果有finally，那么会执行finally的，如果finally有return，那么此时才会退出

## 内部类：

### 普通内部类：

存在于类的内部，其他类想要创建这个类的对象，先创建这个类的outer类。内部类持有外部类的引用，并且可以访问外部类的字段;



从上面可以看出，内部类可以用来解决多重继承的问题。

### 匿名内部类：

比如new Runnable或者new Thread，说白了就是一次性的类的实现

### 静态内部类：

存在于类的内部，使用static 关键字修饰。两个类之间比较独立，除了static内部类可以访问外部类的 static private的区域。

静态内部类和外部类之间很独立，更像是体现一种包含的关系。

### 抽象类和接口的区别：

#### 思想方面：



接口是：先定义规范/约束，然后其他类实现规范

抽象类是现有一批有共同特性的类，然后再弄出一个抽象类

#### 语法方面：

1:接口 可以实现多重继承，不能有普通属性

2:抽象类就是有抽象函数的类（也可以没有），其他方面和普通类没有区别

### 重写重载

重写：

1:参数和返回类型不能变

2:返回类型可以是 原返回值的 派生类

3:权限不能缩小

4:final 和static 不能重写

重载：

1:返回类型 和 参数都可以变（必须）



### 并发hashmap。1.7位啥不用segment模式了？？？

1.8使用了synchronized代替了Segment

1:segment模式比较臃肿，如果数据量很大Segment锁数量多，占空间

2:1.8对与synchronized做了很多的优化，性能也不会特别低