目录

[一、 JAVA基础教程 2](#_Toc497328411)

[1. Eclipse中查看JDK源码遇到source not found问题 2](#_Toc497328412)

[2. 类和对象 5](#_Toc497328413)

[3. 构造函数 6](#_Toc497328414)

[4. 方法的重载 7](#_Toc497328415)

[5. static关键字 7](#_Toc497328416)

[6. public/protected/private/this介绍 7](#_Toc497328417)

[7. 继承 8](#_Toc497328418)

[8. Final关键字 8](#_Toc497328419)

[9. 方法重写 9](#_Toc497328420)

[10. 多态 9](#_Toc497328421)

[11. Abstract类和方法 10](#_Toc497328422)

[12. 接口 10](#_Toc497328423)

[13. 封装 10](#_Toc497328424)

[14. 递归 11](#_Toc497328425)

[15. LinkedList和ArrayListed的区别 11](#_Toc497328426)

[16. 异常-try-catch 12](#_Toc497328427)

[17. 创建写入和读取文件 12](#_Toc497328428)

[二、 Java高级篇 12](#_Toc497328429)

[1. 为什么要掌握Java高级篇知识 13](#_Toc497328430)

[2. Java基础知识快速过一遍 14](#_Toc497328431)

[3. 装箱和拆箱的基本概念 16](#_Toc497328432)

[4. String类 18](#_Toc497328433)

[5. StringBuffer和StringBuilder 20](#_Toc497328434)

[6. Arrays类 21](#_Toc497328435)

[7. 数组拷贝方法 21](#_Toc497328436)

[8. Java中方法的参数是如何传递的 23](#_Toc497328437)

[9. 构造方法和初始化 27](#_Toc497328438)

[10. 继承 28](#_Toc497328439)

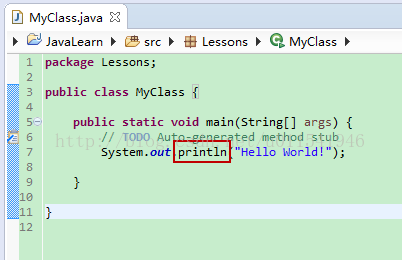
[11. 静态代码块和静态方法 29](#_Toc497328440)

1. JAVA基础教程

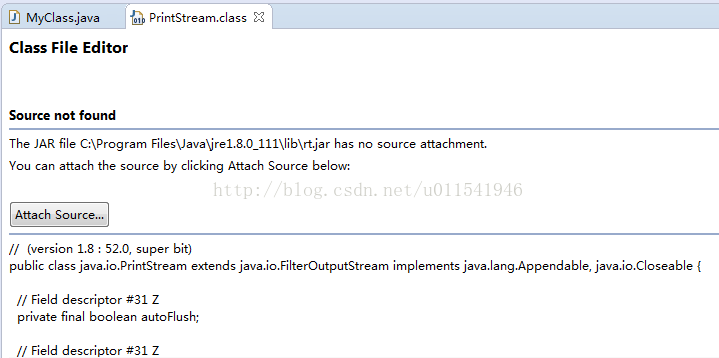
# Eclipse中查看JDK源码遇到source not found问题

这篇文章介绍在Eclipse上面如何查看JDK源码，如果你用过Python下的Pycharm这个IDE，你会发现，它没有Eclipse好用，就光查看源码这个功能来讲，确实没有eclipse好使。那么问题来了，你知道在eclipse上如何查看JDK源码吗，你是否做过这样操作。我们都知道，在我们学习Java过程中，遇到很多不明白的类或者方法，我们就想通过阅读查看对应JDK源码去学习和理解这个方法的实现原理。

## Eclipse上如何查看相关方法的源码



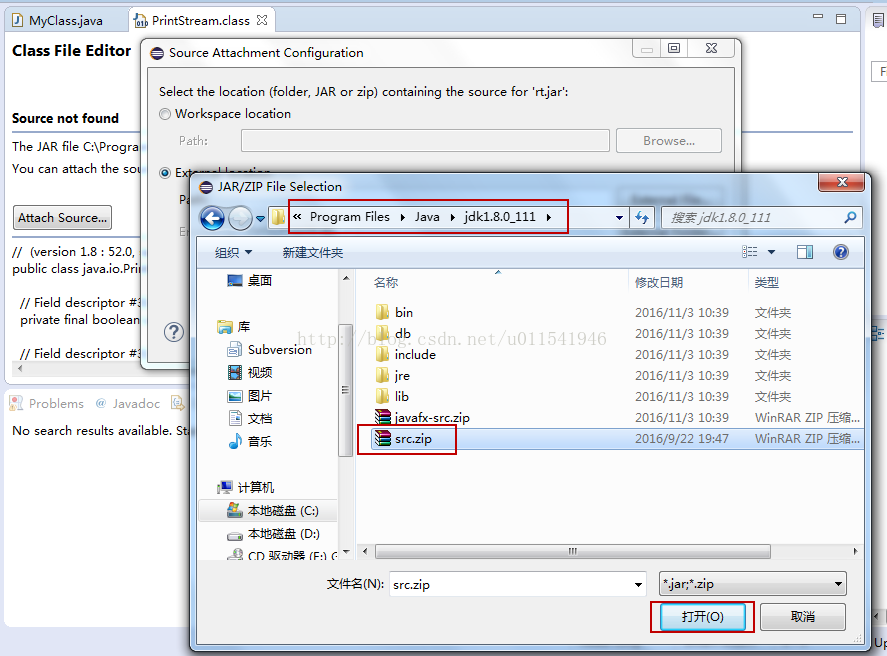
      例如上面这个打印语句，我想看看println()这个方法的源码，具体步骤是这样的：鼠标悬停在println上方，然后按下键盘ctrl键，你会发现println下会出现一个下划线，鼠标点击就发生跳转。如果没有配置源码文件来源，就会出现source not found的页面。



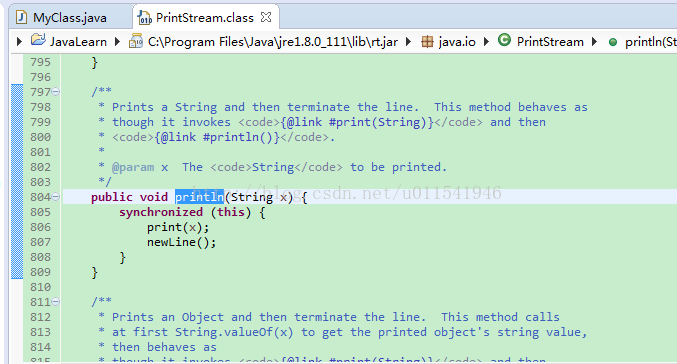
看到上面提示“you can attach the fource by cicking Attach Source belows”，那么添加什么文件呢？

## 1.2 如何解决source not found的问题

点击Attach Source..，选择你电脑安装jdk路径下的一个src.zip包就可以。



再次回到打印这句代码，试试鼠标悬停在println上面，按下ctrl和点击下划线，看看效果



       到这里，如何查看源码和如何设置源码文件就介绍完了，你可以通过这种方法去查看你想学习相关JAVA中方法或者类的实现原理。通过阅读JDK源码，有助于你的编程能力提高和养成想JDK开发人员那样标准的代码规范。

# 类和对象

  本文介绍Java中类和对象的概念。实话讲，刚开始接触这些概念，确实很头疼，不好理解。如果你理解了Java的类和对象，那么就基本上学好了Java一半知识。先来理解下什么是对象和什么是类的概念。

## 什么是类和对象

每一个Java程序至少有一个类和一个方法。类是Java程序的组织单位，一般一个类下，写多个方法。简单来说，物以类聚，类就是分类，归纳的意思。在Java中处理import语句和package语句，其他的代码行都需要写在类里面。说到类，不得不提对象，Java是纯面向对象编程。什么是对象，在客观世界中的任何东西，特别复杂的问题，我们习惯把这些事物抽象成对象。世间万物皆是对象，这句话听说过吧，哈哈。对象有名字，对象有内部属性和行为描述，也叫方法。在Java程序中类是创建对象的模板，对象是类的实例。不说了，估计你会很绕，烧脑子。实在不行，强记住这句话：在Java程序中类是创建对象的模板，对象是类的实例。

## 2.2如何定义一个类

我们前面例子，其实都在一直定义类，public class MyClass {} 就是在定义一个名称叫MyClass的类。public表示修饰符，修饰符以后介绍，class是关键字，表示定义的是一个类，class后面跟着的就是类的名称。类里面可以定义多个方法，还有类本身的属性。

## 2.3看一个具体类的实现

在包lesson下新建两个Class文件，一个叫Student.java，一个叫MyClass.java

Student.java代码如下

1. **package** lessons;
3. **public** **class** Student {
4. **int** id;
5. String name;
6. **int** age;
8. // 这里只写了类属性，没有写类方法。
10. }

MyClass.java内容如下

1. **package** lessons;
3. **public** **class** MyClass {
5. **public** **static** **void** main(String[] args) {
6. Student Anthony = **new** Student(); // Anthony 是一个对象或者叫实例
8. Anthony.id = 1;
9. Anthony.name = "Anthony";
10. Anthony.age = 18;
12. System.out.println(Anthony.name + " is " + Anthony.age + " years old.");
14. }
16. }

运行结果：Anthony is 18  years old.

## 2.4参数设置和获取调优

       仔细看MyClass.Java中的代码，你会发现，如果别人使用你的代码，很可能你Student里面的属性（id,name,age）被别人修改。为了防止别人修改你的值，我们采用set和get方法去修改上面代码。方法是这样：在Student这个类里面，鼠标选择类属性id，然后点击eclipse上传菜单栏的Source,下一级菜单选择Generate Getters and Setter...，你就会发现Student.java代码变化如下：

# 构造函数

  前面一篇介绍了类和对象的基本概念，希望你有一个基本的了解。Java这门语言是纯面向对象的，为了描述实际生活的万物，人们习惯把事物先分类，然后抽象出来。抽象就是抽出类似的部分，把类似的部分分类，用类去管理和组织代码。类是对象的创建模板，所以，要创建一个对象实例，必须先要定义一个类。本文介绍类的中的构造函数。先来了解，什么是构造函数，在类中，函数名称和类名称相同，大小写也要相同的函数就叫构造函数。构造函数的作用就是，每个该类的对象在创建过程就被初始化。也就是说，你新建一个该类的对象，就会自动调用构造函数里面的内容。构造函数还有一个特点，构造函数内部不能有返回语句，即构造函数没有返回值。

总结：

1）构造函数名称一定要和类名称保持一致。

2）构造函数会在对象创建过程是调用

3）构造函数可以有多个，用参数个数和参数类型区分

4）构造函数内部没有返回语句

# 方法的重载

这篇我们介绍方法的重载，方法的重载和构造函数的调用有一点相同。重载的基本特点就是，方法名称相同，但是方法参数类型不同，参数类型不同，包括，参数个数和参数变量类型不同。

# static关键字

  本文来介绍static关键字，在Java中用static关键字的方法，就叫静态方法，静态方法只能被类本身调用，不能被类的实例对象调用（会出现警告，类型不匹配），而没有static关键字的方法，在类中叫实例方法，任何一个实例对象都能调用实例方法。同样，类变量也和方法类似。

总结：

用到了static修饰的方法或者变量，只能被类本身调用，没有static修饰的方法是实例方法，实例对象能调用实例方法。

# [public/protected/private/this介绍](http://blog.csdn.net/u011541946/article/details/73215841)

本文介绍Java中权限访问控制的修饰符，你注意到没有，前面我们的例子，不管是main方法还是实例方法，都是public static 开头。学习了今天这篇文章，你就明白为什么是public开头。Java中权限访问控制的修饰符有四种情况：public private protected 和默认为空。

**先来看看四种情况的访问权限**



       这里提一下任何地方，有些资料，你可以看到的是全世界。如何去理解任何地方这个权限访问呢，例如你到网上下载一个Selenium源码文件，然后关联到你的项目上，在你的项目代码中调用了或者引用了Selenium里面的接口或者方法，这个就可以理解任何地方或者全世界都可以访问的权限。由于我们还没有介绍Java中的继承，所以这个子类权限，我们这里也不做演示，我们来来看看同一个包下不同类文件的访问情况。public任何地方都可以访问，这里也不举例演示。

这里提下this关键字，在Java中，this关键字主要是用来调用该类的属性，例如类成员变量，该类的实例方法，该类的构造方法。你可以这样去理解，this就相当于指的是当前这个类本身，等同于这个类的类名。this.name = name，很多人看到这个会有疑问，这个地方有两个name，如何区分呢。等号左边的name是一个类成员变量，等号右边的name是实例方法中的局部变量，所以this.name = name的意思就是将局部变量的值传递给类成员变量。

# 继承

  本文介绍Java中的继承，Java有三大特性：封装，继承和多态。我们先来简单了解下什么是继承？为什么要使用继承？在现实生活中，我们有很多继承的例子，例如儿子某些特征像他父亲，就可以说儿子这些特征是从他父亲上继承过来的。所以，Java中有了父类和子类的说法，父类就是被继承的类，子类就是将要继承父类的类。使用继承，是为了减少代码的重复，增加代码的复用性。下面我们举例来简单介绍继承。我们的父类是多边形类，有两个子类，分别是长方形，和三角形。长方形和三角形都是多边形，通过这个例子，我们来了解继承。

**继承总结就三句话：**

1) 子类拥有父类非private属性和方法

2) 子类可以拥有自己的属性和方法，即子类可以对父类进行扩展

3) 子类可以用自己的方式去实现父类的方法

# Final关键字

前面一篇介绍了Java中的继承的基本使用和继承的特点，这篇介绍Java中final关键字。什么叫final,字面意思就是最终的意思，也就是不需要修改。在设计类时候，如果这个类不需要有子类，类的实现细节不允许改变，并且确信这个类不会再被扩展，那么就设计为final类。final在Java可以声明成员变量、方法、类以及本地变量。一旦你将引用声明作final，你将不能改变这个引用了，编译器会检查代码，如果你试图将变量再次初始化的话，编译器会报编译错误。final需要记住以下三点就可以。

1)   final类不能被继承  
2）final方法不能被子类的方法覆盖  
3）final变量只能被赋值一次，赋值后值无法改变

# 方法重写

  看到标题，是否想起前面有一篇文章叫方法重载。什么是方法重写呢？在Java编程语言中，子类可继承父类中的方法，而不需要重新编写相同的方法。但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。方法重写又称方法覆盖。

方法的重写有几个特点：

1）发生在子类，有子类就有继承

2）父类方法的参数列表必须完全与被子类重写的方法的参数列表相同

3）父类的返回类型必须与被子类重写的方法返回类型相同

4）Java中规定，被子类重写的方法不能拥有比父类方法更加严格的访问权限

  说到方法重写，不得不提文章开头的方法重载，他们的根本区别就是：方法重写是在子类和父类之间，而方法重载是在同一个类里面。方法重写，子类和父类方法必须保持，返回类型，方法名称，参数个数和参数类型完全保持一致。而方法重载，只有方法名称相同，其他都可以不相同。

还记得前面final文章中的一点：final方法不能被子类的方法覆盖

# 多态

本文介绍Java中三大特性之多态。实际上封装和继承都是为了多态服务的。什么是多态呢？多态指同一个实体同时具有多种形式。或者这样描述，同一个事件发生在不同对象上面，就有不同效果，这样的例子叫多态。下面，我们用多个子类调用同一个方法（子类重写了该方法），会得到不同结果。调用了同样的方法，得到的结果不相同，这就是多态。

**多态存在的三个必要条件：**

1）需要继承

2）需要重写

3）父类引用指向了子类对象

这里解释下第三点，父类引用指向了子类对象。Student st = new CollegeStudent(); 这个st是父类Student的一个引用，但是它指向了子类CollegeStudent()。多态的概念了解一下就可以，在自动化测试和开发中，你也很少遇到。

# Abstract类和方法

  这篇介绍Java中抽象类和抽象方法，用关键字abstract表示抽象，是一个可以修饰类和方法的关键字。如果类名前面用abstract修饰，这个类就是抽象类。如果方法名称前面有abstract修饰，这个方法就是抽象方法。为什么要有抽象类呢，抽象类主要是用来写接口用的，接口在下一篇介绍。只需要抽象定义一个类，不需要具体实现。然后子类可以继承这个抽象类，子类分别去实现抽象类里面的抽象方法。抽象类里面的方法不一定是抽象方法，但是有抽象方法的类一定就是抽象类。抽象类中的抽象方法不需要具体实现，应该由子类去实现。

抽象类和抽象方法有以下特点

1）抽象类不能初始化类的实例对象，但是可以创建子类的引用。

2）抽象方法不需要实现，只需要声明，让子类去实现

# 接口

  本篇介绍Java中的接口的基本使用和简单的概念。接口和抽象类有点类似，接口不是类，接口只是声明了方法，没有具体实现。在JAVA编程语言中是一个抽象类型，是抽象方法的集合，接口通常以interface来声明。一个类通过继承接口的方式，从而来继承接口的抽象方法。接口无法被实例化，但是可以被实现。一个实现接口的类，必须实现接口内所描述的所有方法，否则就必须声明为抽象类。

在Java中接口的实现需要用implements关键字，关键字后面跟着接口。

**什么时候用implements和extends**

1）如果前后两个都是class,那么就用extends

2）如果前面一个是class,后面一个是interface，那么就用implements

3）如果前后两个都是interface，就用extends。

# 封装

封装是把过程和数据包围起来，对数据的访问只能通过已定义的接口。面向对象计算始于这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的接口访问其他对象。封装是一种信息隐藏技术，在java中通过关键字private，protected和public实现封装。什么是封装？封装把对象的所有组成部分组合在一起，封装定义程序如何引用对象的数据，封装实际上使用方法将类的数据隐藏起来，控制用户对类的修改和访问数据的程度。 适当的封装可以让程式码更容易理解和维护，也加强了程式码的安全性。

**封装的作用**

1）对象的数据封装特性彻底消除了传统结构方法中数据与操作分离所带来的种种问题，提高了程序的可复用性和可维护性，降低了程序员保持数据与操作内容的负担。

2）对象的数据封装特性还可以把对象的私有数据和公共数据分离开，保护了私有数据，减少了可能的模块间干扰，达到降低程序复杂性、提高可控性的目的。

# 递归

  本篇介绍递归，什么是递归呢？递归就是自己调用了自己。很多练习题或者面试题，会考察递归的思想，到时候你要想起用递归的思想去解决问题。

构成递归需具备的条件：

1. 子问题须与原始问题为同样的事，且更为简单；

2. 不能无限制地调用本身，须有个出口，化简为非递归状况处理。  
为了描述递归，我们这里演示计算阶乘的值。当阶乘中N=1或者N=0的时候，阶乘就等于1，当N>1的时候， N! = N\*(N-1)\*(N-2)\*...2\*1

# LinkedList和ArrayListed的区别

  前面分别介绍了ArrayList和LinkedList，那么这两个有什么区别呢。特别是两者在操作性能这块，经常在面试中被问到。我也从网上拷贝了几条不同点过来：

1.ArrayList是实现了基于动态数组的数据结构，LinkedList基于链表的数据结构。

2.对于随机访问get和set，ArrayList觉得优于LinkedList，因为LinkedList要移动指针。

3.对于随机新增和删除操作add和remove，LinedList比较占优势，因为ArrayList要移动数据。

注意这里红色字体随机标出来，看完下面例子，你就明白为什么这里要标随机新增操作。

会不会发现，在新增操作，LinkedList花费时间比ArrayList要多很多，不是说新增操作，LinkedList要更快吗。所以，前面标准红色字体随机新增操作是前提，上面例子中，我们在linkedList是做从尾部顺序的新增操作，而不是随机新增，由于linkedList要移动指针，所以这里打印出来的时间要比arraylist要多。

# 异常-try-catch

本篇介绍Java中的异常知识，第一个我们来了解，什么是异常。异常和错误有没有区别。在了解异常之前，我们先学习两个概念。编译错误和运行时错误。编译错误就是，写好了程序，点击编译，在编译这个过程，编译器报的错误提示。运行时错误，就是程序通过了编译，没有发现语法错误，但是在运行程序的时候，java抛出了错误。一般来说，**运行时的错误，就是异常。**所以，我们就很好理解异常。异常就是一种发生在运行程序，突然中断程序的正常流程而发生的错误事件。异常发生在程序运行过程中，下面我们来举例一个简单的异常，也就是一个整数除以零，看看报什么异常。异常的定义：是一种发生在执行一个程序过程，出现打断程序正常流程的一种事件。

在了解异常之后，我们需要学会如何捕获并处理异常。在你写代码的时候，你要根据你的经验，然后主动去捕获可能出现的异常，并写出处理异常的代码，这样你写的程序就会不会因为这个异常而崩溃。在实际测试项目中，经常看到空指针异常而是网页出现HTTP 500的报错，移动端app测试，经常因为数组索引越界而发生崩溃。下面我们学习通过try catch语句来捕获和处理异常。

# 创建写入和读取文件

这篇介绍Java中文件操作，在Java中文件操作，需要用到IO这个包，所以需要导入IO包。我们来介绍如何创建一个空白的文件，然后如果用printwriter来写入内容到文件。

  文件操作结束，记得要关闭文件，否则会一直在内存中加载，浪费内存资源。创建的filename.txt会在你当前eclipse项目的根目录下生成，如果看不到，请右键该项目名称，选择刷新，然后双击打开，看看内容是不是代码中输入的内容。文件的写入，在测试脚本开发这方面运用的主要是日志文件的生成，还有就是自动化截图，保存图片。

我的文件filename.txt是来源上一篇文章生成的文件，这里用BuffreredRead和FileReader类去实现读取文件内容。BuffreredRead类是从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取。

1. Java高级篇

# 为什么要掌握Java高级篇知识

好长时间了，就想要花时间系统去学习下Java的高级篇知识，这部分是我个人目前比较欠缺的，而且是急缺的知识。我认为的Java高级篇内容是这样划分的：对Java这个编程语言有基本了解，基本掌握了基础语法，能够利用基础部分知识去写一些小项目，例如，我们的webUI自动化测试包括简单的框架设计。而Java的高级部分，主要从基础语法到Java web编程中间的这个断层，例如，Java的几个核心常用的类的使用，集合类，JVM，多线程等等。

为什么要学习Java编程高级篇的知识？我个人认为，主要有以下原因。

## 1.1面试需要

       自动化测试，测试开发，高级测试工程师这些职位都需要编程语言技能。特别是国内一线互联网大公司，例如BAT, 京东，小米，滴滴出行，美团点评，新浪，搜狐，再规模小一点的互联网公司，爱奇艺，摩拜单车，OFO单车，今日头条，WIFI万能钥匙。上面的一些公司，一般以上有参加过他们的测试开发的面试，结果很打击，有几个能勉强面试通过，剩下不通过的基本上都是由于代码能力不够。好多次，你对面的面试官，都是从BAT跳槽出来的，他们共同点都是看重代码能力，然后才考虑你的其他能力。有时候，心里会恨不得骂他们，我是来做测试，为什么问题全部是开发的知识。没办法，测试的发展趋势，就是要用代码去测试开发的代码。

## 1.2个人技术成长需要

       每个人都应该有一个好的清晰的职业发展规划。应该知道自己现在所处阶段，最缺的是什么。代码编程能力，绝对是很多测试工程师欠缺最多的一方面。我也同意，有些人说，不管是黑盒测试还是白盒测试，测试的思维最重要。代码能力和测试工具只不过是手段。这个观点，在一定情况下是成立的，但是你根本就没有代码能力，不要再为自己不会代码而且解释什么。我在QQ群里说过，我从webui自动化入门然后到放弃。我也建议大家这么去做，从一个工具或者一门语言去入门自动化测试，然后积累到一定时候，需要自己去思考，你当前的是不是达到了自动化测试的效果，有什么缺点和优点，面临什么挑战，工作中哪些问题不能实现自动化，不能实现的原因是什么。最后，应该会得出一个结论，webui自动化只是前端，当然包括一部分接口测试，但是后端呢，还有端对端的自动化呢?说白了，我们代码能力太窄了，不足以支撑我们是实现很多场景自动化测试。如果你要设计一个自动化测试平台，Java语言的高级编程部分+web编程，绝对是你需要学习的地方。

## 1.3市场需要

       很多公司，基本上Java开发工程师的比例是最多的。你学习好Java，面试机会和选择机会更大。只是，目前Java自动化测试要比Python自动化测试要多很多。很多公司Python只是运用在自动化运维这方面，他们喜欢认为python是脚本语言。还一个现实问题就是，你如果用Java做自动化测试，遇到什么问题，直接问公司开发人员。他们都懂Java，能快速帮你解决问题。如果开发都是Java程序员，你测试采用Python，遇到问题，只能自己默默去解决。

## 1.4高级部分知识点大纲

       下面大概列举出一些Java高级编程的知识点，我们很多是在webui自动化中没有用到过的，学习好了这些，更好地帮我们实现工作中的一些自动化测试，而且你发现没有，面试经常出现的多线程，集合类，算法和高并发这些概念，基本都包括在本系列。一起来学习Java，一起讨论和进步。

**Java基础**

**断言**

**正则表达式**

**集合类**

**泛型**

**高级I/O类**

**日志API**

**网络**

**多线程**

**方法远程调用**

**JDBC数据库连接**

# Java基础知识快速过一遍

在学习Java高级部分知识之前，我们来简单梳理一下Java基础语法部分的知识，不会太全，但是基本的，常用到的一一复习一下。

## 2.1 Java编码规范

       在这里，还是要强调Java编码标准或者规范。以前，我写代码或者脚本不多，也不知道这个重要性。突然，一天，看别人写的代码。火大了。写了什么鬼东西。很多没有写注释，搞得就是看得懂代码，结果不知道这段代码实现什么业务功能。很多公司，这个代码规范是由开发的领导来约束，也有是QA部门来做这个事情。编码规范，主要是注意以下几点。  
**类名称** ，例如Car, BusinessObjectName。建议首选英文，中文拼音还是算了吧。没有意义的单词更加不要用。多个英文单词组合的名称，每个英文单词开头字母都要大写，其他字母小写。这个是国际规范，看起来舒服。  
**函数名称**，例如getAge, setHeight。主要规则是，第一个字母小写开头，后面单元首字母大写。和类名称有一点区别。  
**变量名称**，例如age, myAddress。主要约束是，不能用系统保留的关键字，不能数字开头，除了字母之外字符只支持下划线。  
**常量名称**，例如MAX\_HEIGHT,MAX\_AGE.主要规则，常量名称每个字母都大写。

## 2.2流程控制和循环语句

主要会以下几种语句，Java的基础部分知识。

For 循环

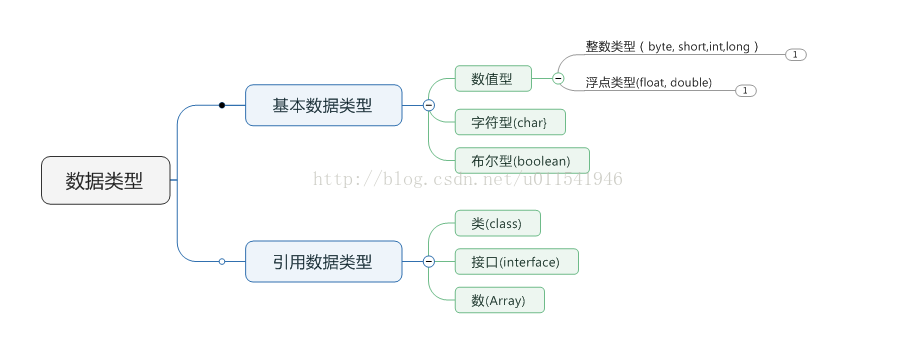
While 循环

If 判断语句

Switch-case 语句

## 2.3关键字，变量作用范围，基本数据类型，引用数据类型。

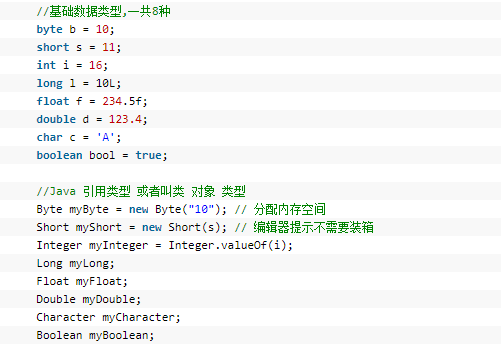
**关键字**也叫系统保留字，主要是学习知识点多了，这些就记得住，这里可以自己去百度查一下。  
**变量作用范围**：全局变量和成员变量  
**基本数据类型**：主要有八大基本数据类型,整数类型（byte,short,int,long），浮点类型（float,double），字符型（char）,布尔型（boolean）  
**引用数据类型**：类，接口，数组



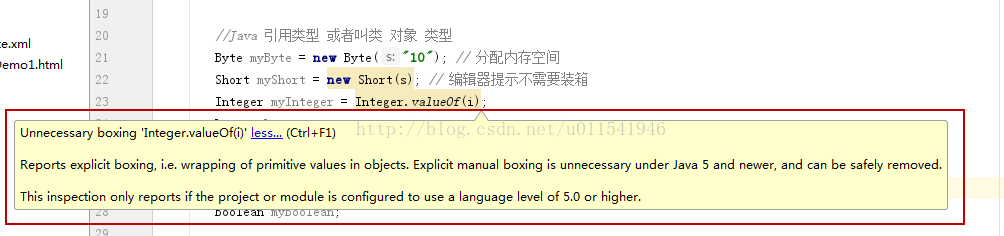
       其实，掌握好了这些Java基础部分知识，完全是可以实现webui自动化测试中的很多工作，写写脚本用例还是没有问题。当然，框架设计部分知识可能依靠上面的还不能实现。

# 装箱和拆箱的基本概念

前面一篇的结尾，我们画了一张图，对Java的数据类型进行了分类。其中，有8种基本数据类型，还有引用类型。我们先来看看下面这段程序。



       我们不需要运行这段代码，前面我们知道8种基本数据类型的声明和初始化操作。但是看第二段代码，发现有和八种基本数据类型一一对应的类。先不管它们是什么和前面有什么关系，这里先看看IDE中的一个警告提示。



       第一句说，没必要的装箱。下面一段详细解释了，在Java5和之后版本，基本数据类型装箱是不需要了。那么问题来了，这么出现了一个新的概念，装箱，既然有装箱，肯定有拆箱的概念。所以，我们先学会基本的装箱和拆箱的概念。

## 什么是装箱和拆箱

       在回答这个问题之前，我们可以网上查阅资料。在Java5版本之前，如果需要创建一个值为8的Integer对象，必须要这样写代码：

Integer i = new Integer(8);

而在Java5之后，提供了自动装箱的特性，如果要创建一个值为8的Integer对象，变成了这样写代码：

Integer i = 8;

       这个过程中会自动根据数值创建对应的 Integer对象，这就是装箱。那什么是拆箱呢？顾名思义，跟装箱对应，就是自动将包装类型转换为基本数据类型：

Integer i = 8 //装箱

int n = i; //拆箱

       具体到一种类型来讲，int是基本数据类型，Integer是int的包装器类。我们平时习惯的基本数据类型的声明和初始化就是利用了自动拆箱的原理。所以，结论：装箱就是自动将基本数据类型转换为包装器类型；拆箱就是自动将包装器类型转换为基本数据类型。

## 3.2看int包装器类的valueof方法源代码

对应的valueof()的源码是

public static Integer valueOf(int i) {

if (i >= IntegerCache.low && i <= IntegerCache.high)

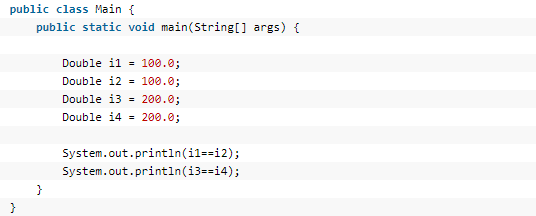
return IntegerCache.cache[i + (-IntegerCache.low)];

return new Integer(i);

}

       这里代码很短，我们应该都能读懂。就两个分支，如果在范围内，调用IntegerCache.cache()方法，返回已经存在的对象引用。如果不在范围就新建一个对象，看到new就想起新开辟内存空间去存储对象的值。那么这个范围是什么呢？按住ctrl键，点击IntegerCache，看源码。

      看代码和这个方法前面的注释文字，我们知道这个方法就是给范围在-128到127内的int类型的基础类型进行自动包装。大概对这个范围有印象就可以。下面这道笔试题就是考察这个知识点。



       如果你知道范围在-128到127会返回内存中存在的引用，超过127就会采用new一个对象，而new是需要新开辟内存空间的。所以上面的==其实就是比较内存地址，而不是比较数值大小。所以i1和i2两个内存地址一样，输出true。但是i3和i4大于127，这个自动装箱走了new这个代码，新开辟了内存，所以两个内存地址肯定不同。

# String类

  这篇来学习Java中的String类。这个类比较基础，也很重要，工作中经常要使用。例如自动化测试中，前后两个字符串变量的比较，有时候我们需要比较是否包含关系，比较是否相等关系。有时候我们需要判断是否以什么开头，以什么结尾等。这些比较都可以在String这个类找到相应方法。实际上，字符串对象在Java中是用有一个char数组实现的。字符串是常量，一旦创建好了就不能修改，但是可以拷贝字符串对应的char数组去做其他操作。

## 字符串创建

字符串创建有两种方式，直接看下面代码。

String s1 = "Hello";

String s2 = new String("Hello");

       一般我们都是使用s1的创建方式来创建一个字符串对象。s2这种方式是利用了String类的构造方法来实现，需要在内存地址开辟一个空间，把原字符串的值复制一份，存储到新的内存地址中。所以，s2创建字符串对象有一个性能问题，一般没必要这么做，耗内存。

## 4.2返回字符串长度

       字符串长度和比较是有关系的，如果比较相等，长度都不同，立马就判断不相等。获取字符串长度的方法是length(), 返回的是int类型的值。

## 4.3判断字符串是否为空

判断是否为空的条件是，字符串长度等于0，方法名称是isEmpty()。返回的是布尔型值，true或者false两个值。

## 4.4根据索引，获取字符串中一个字母

       这个场景，也是经常用，例如如果要判断两个字符串相等。肯定要两个字符串字母一一取出来，然后去比较。这个取出来的过程就需要用到charAT(int index)方法，如果找到就返回这个字母。

## 4.5判断两个字符串是否相等

这个经常在面试中遇到，利用equals（）方法。先看看源码，然后自己动手根据自己的比较逻辑去尝试实现下。

## 4.6 startsWith(String prefix, int toffset)

判断这个字符串是否以xxx开头，这个经常用来定位文件中某一行。然后才进行其他操作，例如替换字符。

## 4.7 endsWith(String suffix)

判断这个字符串是否以xxx结尾

## 4.8 indexOf(int ch)

返回某一个字母的下标索引。

## 4.9 subSequence(int beginIndex, int endIndex)

得到该字符串的一个子字符串。

## 4.10 concat(String str)

新字符串拼接到原字符串后面。类似我们采用str1+str2效果。

## 4.11 String replace(char oldChar, char newChar)

字符串中字符替换，替换全部符合匹配的。

## 4.12 String replaceFirst(String regex, String replacement)

       替换找到匹配的第一个字符。这个在将配置文件中的注释给去除，很有用。例如某行配置文件的内容默认是用#开头给注销，现在需要去掉注销，利用这个替换就到达目的。

## 4.13 contains(CharSequence s)

判断是否包含某些字符串

## 4.14 split(String regex, int limit)

字符串切割，工作中经常用到。

       上面介绍了String类中常用方法，还有一些方法用到很少，就没有去调用测试下。String 类包括的方法可用于检查序列的单个字符、比较字符串、搜索字符串、提取子字符串、创建字符串副本并将所有字符全部转换为大写或小写。使用频率高的方法还是需要去读一读源码，了解下该功能的实现原理和思路。

# StringBuffer和StringBuilder

  前面一篇，我们介绍了String这个类，其中String类提供的一些方法中有些方法是基于StringBuffer实现的。在Java中如果涉及到频繁的字符串修改操作，建议采用StringBuilder和StringBuffer，肯定不用String类。肯定有人会问，StringBuilder和StringBuffer有什么区别呢？这个在面试中也经常被问道。就一句话，StringBuffer和StringBuilder的方法基本上是一样的，唯一的区别是StringBuffer是线程安全的，而StringBuilder不是。

        StringBuilder和StringBuffer和String不一样，Stirng是不可变化的。但是StringBuilder和StringBuffer，它不是final的，可以修改。字符串最常见的操作就是添加和删除。这里写一个程序分别用String类和StringBuilder和StringBuffer，三种方法来对字符串添加的操作。看看分别耗时多少，而且前面一段提到StringBuilder不是线程安全，StringBuffer是线程安全。线程我们暂时不管，后面学习。线程安全，意味着有代价，那就是要慢一些。下面的例子，可以证明三种类对字符串添加操作的速度快慢对比。

  结论就是，从操作效率来看：StringBuilder > StringBuffer > String。 所以，如果不考虑线程安全角度，尽量多用StringBuilder来操作字符串的修改。

# Arrays类

前面介绍了字符串的操作的String类，这篇，我们学习数组类，Java中提供了一个数组常见操作方法的类Arrays.java。其实前面也说过，String其实源码中就是用一个char[]数组来存储的，也是属于数组的范围。来看看数组类常见操作方法举例。

1. 数组的创建和元素遍历

2.数组中元素查找和替换和排序

主要有根据索引查找元素，根据索引替换元素的值，排序Java提供了从小到大排序，这里我写了一个从大到小的逆序，前提是基于从小到大排好序的数组。

# 数组拷贝方法

前面我们介绍了Array和Arrays类几个常用的方法，没有提到对象clone()方法和Arrays.copyOf()，关于拷贝，放到这篇来讨论。拷贝或者复制，一般是对象的引用的操作，有三种方式来实现：直接赋值，浅拷贝，深拷贝。下面我们用代码来理解这三种情况。面试中经常拿数组的几种拷贝方法，来考察你对浅拷贝和深拷贝的理解。特别是直接赋值和对象clone这两个方法。

## 7.1直接赋值

直接赋值很简单，例如Int a = 10, 第二行b=a,这里直接把a赋值给了b。这个时候，a和b是相等，内存地址相同，值也相等。我们用一个int基本类型的数组来举例。

int[] a = {40, 20, 50, 10, 60, 30};

int[] b = a;

从运行结果看出：数组a和数组b确实是完全相等，值相等，内存地址也相同。也就是说数组a和数组b实际上都是指向了同一个对象。上面修改了数组a的值，直接赋值得到的数组b的值也跟随变化。

## 7.2浅拷贝

上面直接赋值的结果就是数组a和数组b完全相同，如果修改a，b也会跟着修改。实际上，我们可能不希望这样，我们希望复制过来的b和a进行区分，两者保持独立，不能互相影响。实现这个功能有一个方法就是Object的clone()方法。Object是Java的对象，所有类的父类。这个克隆实现的功能是：创建一个新对象，然后将当前对象的非静态字段复制到该新对象，如果字段是值类型的，那么对该字段执行复制；如果该字段是引用类型的话，则复制引用但不复制引用的对象。因此，原始对象及其副本引用同一个对象。

7.2.1 我们先看看如果字段是值类型，也就是八大基本数据类型的情况

int[] a = {40, 20, 50, 10, 60, 30};

int[] b = a.clone();

上面可以看出，数组a是Int类型，也就是一个基本数据类型的数组，所以直接进行了值的复制。复制之后，数组b和数组a是内存地址不相等。如果我们操作数组a，数组b是不会受到影响。

7.2.2 看看如果是引用型的数组

String[] a = {"Beijing", "ShangHai", "HeBei", "GuangZhou", "XiAn"};

String[] b = a.clone();

上面我们数组变成了String类型，里面的元素值也是String，我们知道String是一个类，不属于基本数据类型，所以，这里是引用类型。因为数组索引为2的元素是一个String类型，数组a和数组b的索引为2的元素都指向同一个索引，当修改a中引用指向的值是WuHan,数组b由于引用没有发生改变，所以值也没有跟着变化，还是指向了HeBei。clone()比较特殊，对于对象而言，它是深拷贝，但是对于数组而言，它是浅拷贝，因为数组每个元素如果是引用类型的话，数组本来存储的就是对象的引用。

7.2.3 System.arrayCopy()实现浅拷贝

String[] a = {"Beijing", "ShangHai", "HeBei", "GuangZhou", "XiAn"};

String[] b = new String[a.length];

System.arraycopy(a, 0, b, 0, a.length);

7.2.4 Arrays.copyOf()实现浅拷贝

String[] a = {"Beijing", "ShangHai", "HeBei", "GuangZhou", "XiAn"};

String[] b = Arrays.copyOf(a, a.length);

关于浅拷贝，推荐7.3或者7.4的方法，7.3执行速度更快。我分别测试了下，System.arraycopy显示0毫秒，而Arrays.copyOf()输出1毫秒。其实Arrays.copyOf()内部实现是调用了System.arraycopy方法。

## 7.3深拷贝

深拷贝，就是说创建一个新对象，然后将当前对象的非静态字段复制到该新对象，无论该字段是值类型的还是引用类型，都进行复制。由于数组元素本来就存储对象的引用，所以，这里无法举例来实现数组的深拷贝。以后在类和对象的知识来介绍对象的深拷贝。

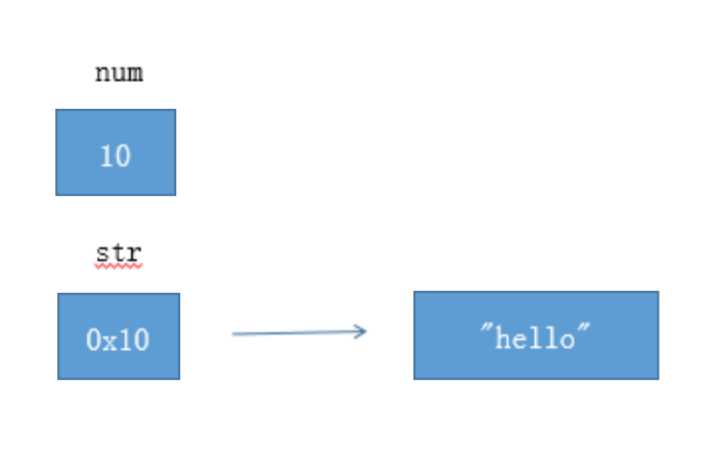
# Java中方法的参数是如何传递的

本篇来学习下Java中的方法的参数是如何传递的。首先清楚两点：

一：搞清楚 基本类型和引用类型的不同之处

int num = 10;

String str = "hello";

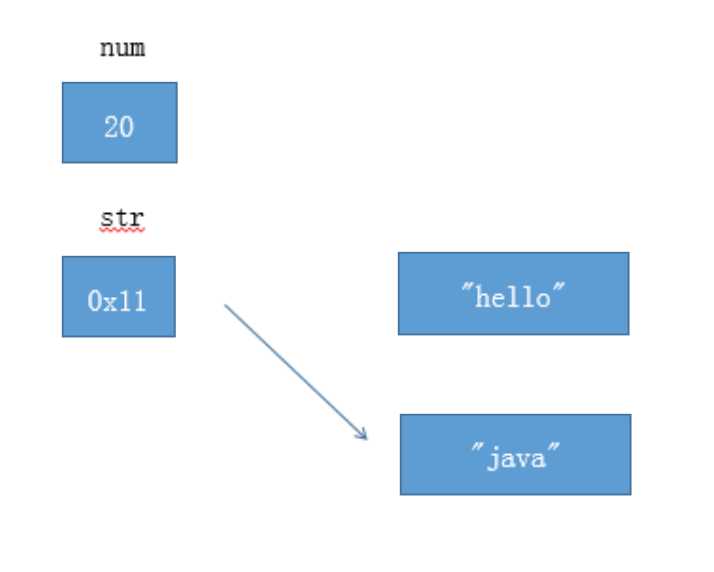


如图所示，num是基本类型，值就直接保存在变量中。而str是引用类型，变量中保存的只是实际对象的地址。一般称这种变量为"引用"，引用指向实际对象，实际对象中保存着内容。

二：搞清楚赋值运算符（=）的作用

num = 20;

str = "java";

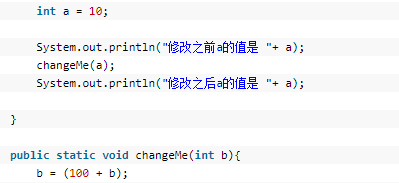


对于基本类型 num ，赋值运算符会直接改变变量的值，原来的值被覆盖掉。  
对于引用类型 str，赋值运算符会改变引用中所保存的地址，原来的地址被覆盖掉。**但是原来的对象不会被改变（重要）。**  
如上图所示，"hello" 字符串对象没有被改变。（没有被任何引用所指向的对象是垃圾，会被垃圾回收器回收）

应该都听说过值传递和引用传递，值传递很好理解，就是把这个实际的值给传进去，引用传递就有点费脑子。在Java的方法中，引用传递传递的不是引用本身，而是之前应用的一个副本。这样说，不能理解没关系，我们分别通过代码来理解值传递和引用传递。

## 8.1值传递

       Java中按值传递（call by value），先记住这个结论：值传递，方法得到的是所有参数值的一个拷贝，方法并不能修改传递给它的任何参数变量的内容。下面那一个例子来举例。



**分析：**

为甚执行了changeMe(a)这个方法里的代码，a的值没有变成110呢？

1）先声明一个int类型变量a，a的值是10.

2）这里把a当做参数传给changeMe方法，关键点在这里，这个时候b（为了和a区分开来，用b）被初始化为a的对应值，也就是10的一个拷贝，此时b也等于10

3）b执行了b=100+b，也就是等于110，但是这个不是a的值，此时仍等于10.

4）这个方法结束后，参数b不再使用，会被JVM回收。

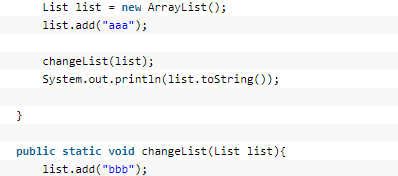
5）所以，打印a的值还是等于10

      我们可以得出这个结论：当方法参数的类型是基本数据类型（8种）的时候，一个方法是不可能修改一个基本类型的参数的值。

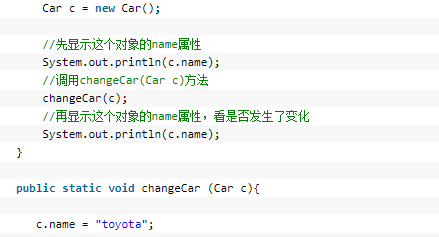
## 8.2引用传递

       前面我们知道，java中数据类型除了8种基本数据类型，剩下的就是引用类型，包括类，数组。所以，这里我们举例一个List。

      分析：main()方法先创建了一个ArrarList对象list，并且添加了一个元素aaa，然后manin()方法调用changeList()方法的时候，changeList中参数list是main()方法中变量list的一个副本。这两个list变量都指向堆中的同一个ArrayList对象，所以操作的是同一个，结果会添加bbb到ArrayList中。

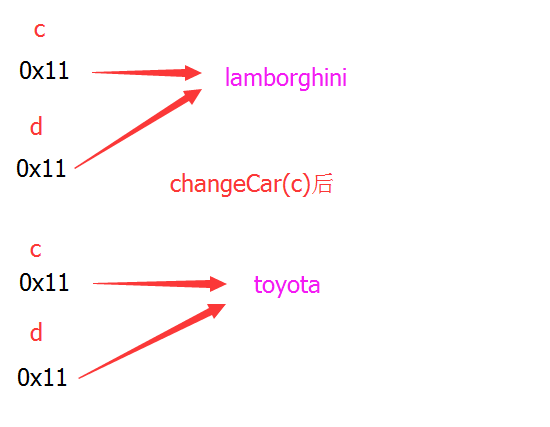


来看一个类作为参数传入的引用传递



       正确理解：当一个对象实例作为一个参数被传递到方法中时，参数的值就是该对象的引用一个副本。指向同一个对象,对象的内容可以在被调用的方法中改变，但对象的引用(不是引用的副本)是永远不会改变的。

       这个知识点很费脑子，我理解也不是很透彻，还是有疑惑，先这样吧，以后对内存堆和栈有了深入理解再来填坑。

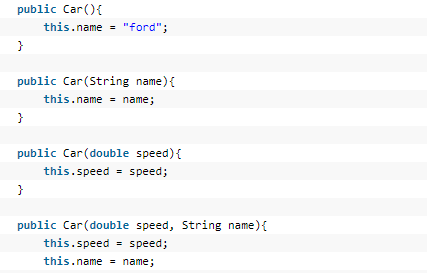


# 构造方法和初始化

这篇来学习下构造方法的话题，首先什么是构造方法。所谓构造方法，就是这个类在被实例化时（创建对象时）就要执行的方法，方法名和类名相同。下面，主要从三个方面来学习构造方法：构造方法的定义，构造方法重载，构造方法内参数的初始化。

## 9.1构造方法定义

       我们已经知道了构造方法是和类同名，其实还有其他的条件。构造方法是一定没有返回值的，所以不能有类型返回，连void都不可以有。构造方法前面可以用private,public,protected修饰，也可以不写，但是不能有static、final、synchronized、abstract关键字修饰。下面来看一个例子。



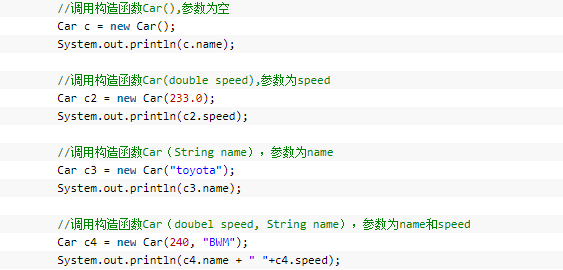
       上面代码，我们只看上面四个构造函数，不看构造函数内部代码。上面有四个构造函数，分别是Car(), Car(String name), Car(double speed), Car(double speed, String name), 构造函数的名称和类名称相同，前面不能有static或者返回类型的修饰。关于构造函数的定义，就可以像上面这样去写，下面我们讨论下构造函数内参数。

## 9.2构造函数的重载

       构造方法（函数）和普通的实例方法一样，也支持方法的重载（overload）。先复习一下什么是重载。重载就是方法名称相同，只是参数的个数，类型，顺序不同。Car(), Car(String name), Car(double speed), Car(double speed, String name)这四个就是构造方法的重载。第一个是没有参数的，第二个和第三个都是一个参数，只是参数类型不同，第四个是两个参数。构造函数写了好几个，那么，类初始化的时候到底调用哪个呢。

## 9.3类初始化和调用构造函数方式

       先说一下，如果一个类没有写构造函数。Java编译器会自动给你添加一个默认的什么也不干的构造函数。类的初始化是根据提供参数个数，类型来自动去匹配调用哪个构造函数，如果匹配不上，就好报错。下面我们新建一个main方法，来测试下构造函数的调用和类实例的初始化。



通过打印出来的结果，可以看到调用了那个构造函数。这里先说说类的初始化， Car c = new Car(), 我们这里看到了new这个关键字，new表示新建一个对象，放在内存中。所以构造函数在调用的时间就是执行了new Car()就完成了调用。然后我们来思考下，构造方法中参数如果没有传参数是如何初始化的。很简单，你可以这样测试输出看看效果。

Car c = new Car();

System.out.println(c.name);

System.out.println(c.color);

System.out.println(c.speed);

       你会发现，由于这个实例在初始化过程没有去管color和speed的值是什么，但是运行结果我们看到color=null, speed=0.0 所以，在构造函数中基本数据类型的参数一定要进行初始化操作，如果没有，int型就是0，上面我们speed是doubel型，所以显示0.0，字符串默认返回null，char默认返回'\u0000'，布尔型默认是false。

# 继承

本篇来学习下Java中的继承知识，说到继承，应该能想起Java面向对象的三大特性，继承，多态，封装。我们本篇来了解继承，其实多态和封装和继承是有关系的，理解了继承，剩下两个概念就很好理解。要了解继承，我们先要了解父类的概念。Java中任何类都有一个父类，即使你没有声明父类，也有一个隐含的父类，叫Object。Object没有定义，但是定义了一些方法，例如我们常用的equals和toString方法。

**什么是继承和为什么要继承**

继承，就像生活中的财产继承。在Java中，有一个父类，通过继承，子类就自动拥有了父类的属性和行为。子类继承父类是通过关键字extends实现的。因为通过继承，能减少我们很多重复的代码，提高开发效率。

这里，提一下，重写方法，会提示final方法不能被overwrite。所以，在实际项目中，如果父类哪个方法不希望被子类修改，就在这个方法添加final修饰。

**测试中用到继承的项目举例**

很多人写过webui自动化测试脚本，如果脚本写多了，就会发现很多重复的代码。这个时候就要考虑框架，其中框架最重要的是实现帮你少些很多重复代码，这个时候继承就是很好的选择。我们一般做法是，定义一个页面基类，这个类封装一些常见的元素操作方法，子类通过继承这个父类。这样所有的测试页面都同时有元素操作方法。

**封装**

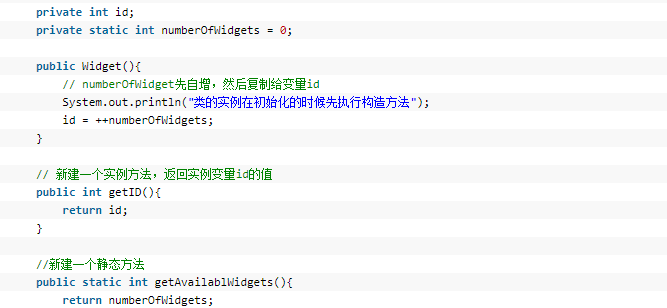
封装就是隐藏了实现细节。例如父类定义了一个画图方法，继承的子类都有了画图功能，但是不知道父类画图是如何实现的。还有很多类似不希望外部了解的实现过程，都可以通过封装来实现。函数和类其实就是封装的过程，函数或者类提供给很多其他调用，但是调用的对象不需要知道函数和类内部实现细节。

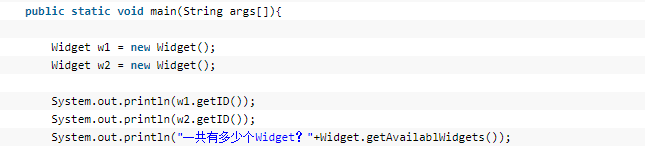
**多态**

多态，就是有多种状态，一个对象有多种特征。多态的实现，主要是通过子类修改父类的方法实现。

# 静态代码块和静态方法

这篇来讨论下静态代码块和静态方法，静态方法我们已经知道了，就是这个方法的名称前有一个static修饰符。先来看看什么是静态代码块。一般情况下,如果有些代码必须在项目启动的时候就执行的时候,需要使用静态代码块,这种代码是主动执行的，例如Java中程序执行入口的main（）函数。在不创建对象的情况下,其他程序来调用的时候,需要使用静态方法。静态方法是直接通过类名.静态方法名来调用的。静态代码块和静态方法的唯一区别就是，静态代码块是自动执行的，而静态方法是被动执行，也就是说只有被调用，静态方法才会被执行。





为什么是这个输出结果，我们要弄明白这个main函数里面代码的执行过程。

1. 实例在初始化过程就会调用构造方法

Widget w1 = new Widget();

第一行执行这个代码，实例w1 通过new关键字开始初始化，在这个过程中，类的构造函数里面就执行，而我们构造函数写了numberOfWidgets是运行一次就自增1，这个时候调用w1.getID(),打印的结果就是1.

2. 静态方法是不需要初始化，直接通过类名调用

上面我们初始化了两个Widget实例，所以，执行了两个构造函数，numberOfWidgets和ID就变成了2. 直接通过类名来调用类的静态方法。

3. 实例对象是不能调用静态方法

实例对象是不能调用静态方法的，你可以在main方法中试试w1.getAvariableWidgets(),会提示错误。因为静态方法在使用前是不需要创建实例对象。

4. 静态方法不能以任何方式引用this和super关键字

因为静态方法在使用前不用创建任何实例对象，当静态方法调用时，this所引用的对象根本没有产生。this一般只实例对象本身，super只这个实例的父类。

java静态变量和静态方法会在程序初始化运行时候就加载到内存。

优点：不需要在进行实例化。静态变量的值，直接赋新值即可，不需要参数传递，之后可以直接进行参数引用即可；静态方法可以直接通过"类名.方法"的形式进行方法调用。通常方法被多次调用，并且方法中没有动态方法引用的时候使用比较方便。

缺点：初始化加载到内存，如果后续没有被引用，加大了内存负担和程序运行负担，影响程序运行效率（一般很小），并且静态变量如果多处被引用赋值，很可能导致参数值混乱（如果是不变的值，最后加上final修饰，强制不能修改）。

不要随便定义成静态方法, 因为静态方法内部只能使用静态变量,而静态变量是属于类的, 如果都是静态方法那么这个类就没有实例化的必要.所以一般的工具类才使用较多的静态方法

一开始加载class文件时，会把static加载到静态区，长存，JVM不会回收。

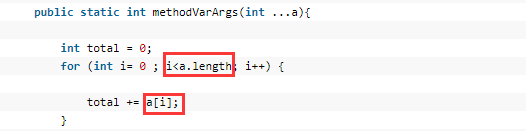
对性能有一定的影响，对于static内容不要太多，尽量避免大内容static。

特别提醒：**如果不是常量固定值的话，建议少用static定义变量。**

备注：上面的都是理论知识，实际开发中，根据实际项目需求进行灵活调整即可。

# 可变长参数的使用

  本篇学习java中的可变长参数（varargs），使用varargs的时候，说明这个方法中，你暂时不确定有多少个参数传入。但是在调用该方法的时候，你就需要具体写出多少个参数。在同一个方法中，使用varargs必须要求参数都是同一个数据类型。一般格式是这样的public void fun(int ...x)，小括号的三个点表示省略。



  从上面可以看出，可变参数可以当做数组使用，在for循环里，使用了a.length和a[i]。这里要注意一点，一个方法只能有一个可变参数，而且这个可变参数必须是该方法的最后一个参数，注意书写的时候，把可变参数放小括号里最后一个参数。

# 静态导入

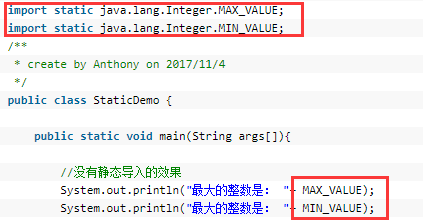
前面文章，我们知道了Java中的静态方法和静态代码块。这篇来说说静态方法的使用相关的知识，静态导入。什么是静态导入呢，就是在导入包语句中直接导入该类的静态方法。

1.打印int的最大和最小整数是多少，我们先来看看没有使用静态导入的效果。

上面我们直接输入包含静态方法的类，通过类名.静态属性来输出了最大整数和最小整数。这种方式，就是我们前面介绍的常用的静态方法的基本使用方式。

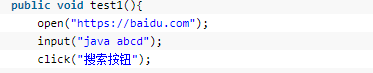


2.下面来看看使用了静态导入的效果。



 注意上面的两行代码，，import static ...，在Java中使用import static叫静态导入包，简称静态导入。通过静态导入方式，该类文件中可以直接写静态方法名称或者属性，这样写，代码看起来简单。例如，你不想写System.out.println，你可以先使用静态导入，然后直接写println就可以。

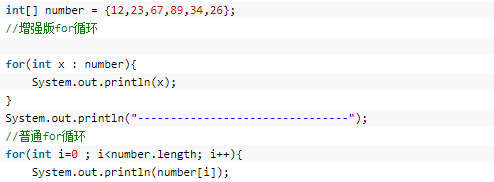
      下面来举例一个测试中可以使用静态导入的例子。假如，下面是我们平时例如testng来写webUI 自动化测试的场景，BasePage这个类下封装了open(),input(),click(),close()四个静态方法。



# 增强型For循环

本篇来学习Java的增强型For循环，也叫for-each循环。一般，我们遍历元素的时候，喜欢采用for循环。那么到底采用哪种for循环呢？主要是取决你的需求，如果你要对元素下标，也就是索引进行操作，那么选择传统的for循环，如果你实现操作和索引没有关系，建议采用for-each循环。下面通过例子，来介绍下这样的情况。

1. 分别用for-each 和传统for循环实现数组遍历



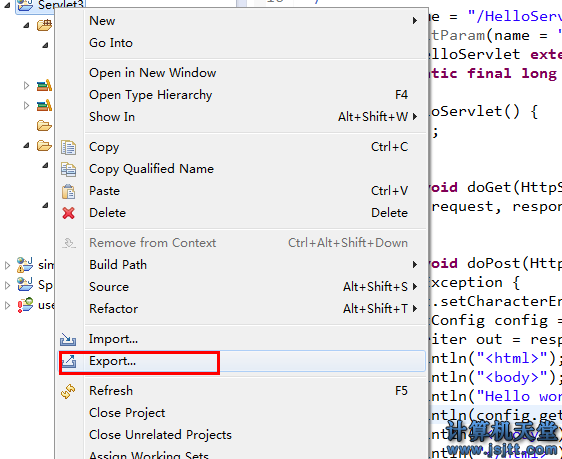
总结：特别是集合类操作中，如果要实现的需求不需要操作索引，我们最好使用for-each循环。这个语法，我们在python中也可以找到，Python是这样写的for(x in list)。

# 如何把项目导出jar包

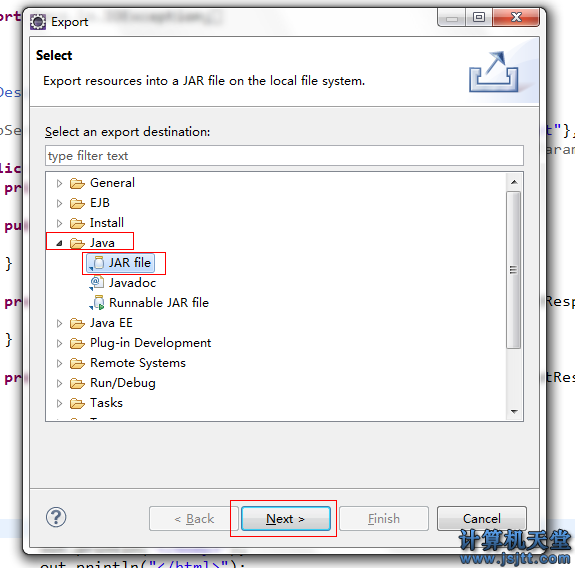
很多时候，我们需要把项目导出成jar包，通过运行jar来运行程序，不希望别人直接看到我们写的代码。例如，在java开发的很多开源插件中，都是通过导出项目成...jar包，别人通过引用这个jar文件，就可以使用里面的API。

前提条件：新建一个java项目，包下有一个main方法。

第一步：右击项目名找到 Export选项 如图：

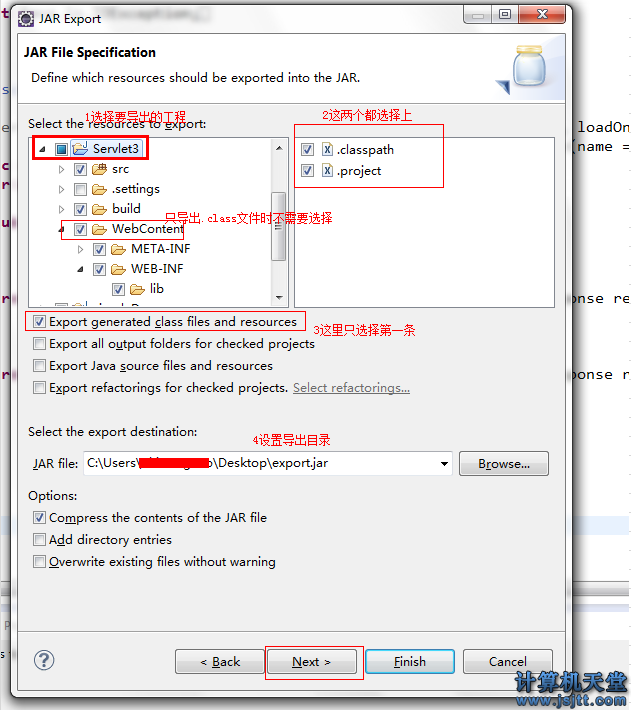


第二步：找到Java->JAR file->next 如图：



第三步：选择要导出的项目，选择要导出的文件目录，勾选calsspath，project

选择Export generated class files and resources，在Browse中选择要导出的目录 下一步,如图：

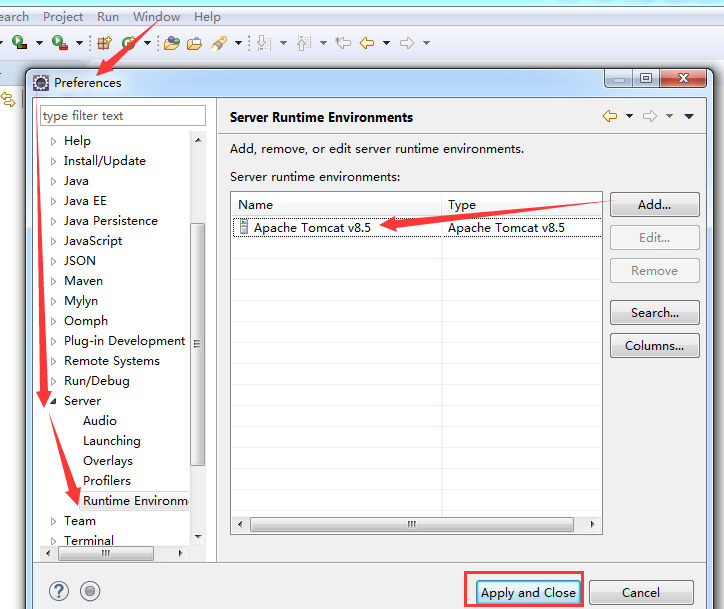


第四步：下面下一步，下一步就可以了 最后点击Finish

# 创建java web工程

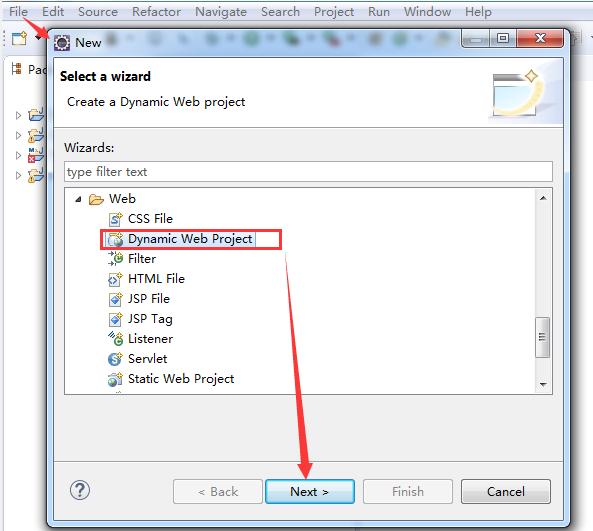
## 16.1设置运行的Server Runtime Environments

Window-Preferences-Server-Server Runtime Environments

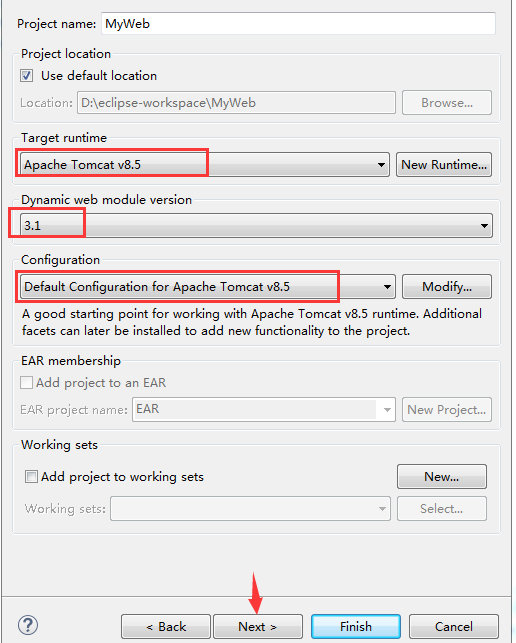


## 16.2创建web项目

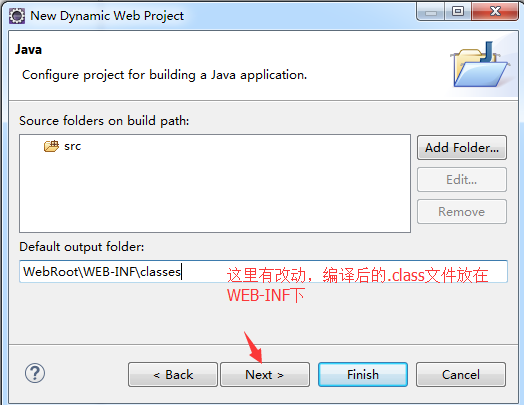
1.File-New-other-Dynamic web Project,点击next



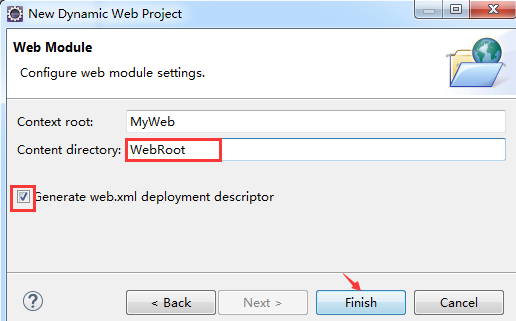
出现工程设定窗口中设定工程的名字（ MyWeb）和Dynamic web module version（版本），我使用的是最新的版本，tomacat8.5对应的是3.1）单击next



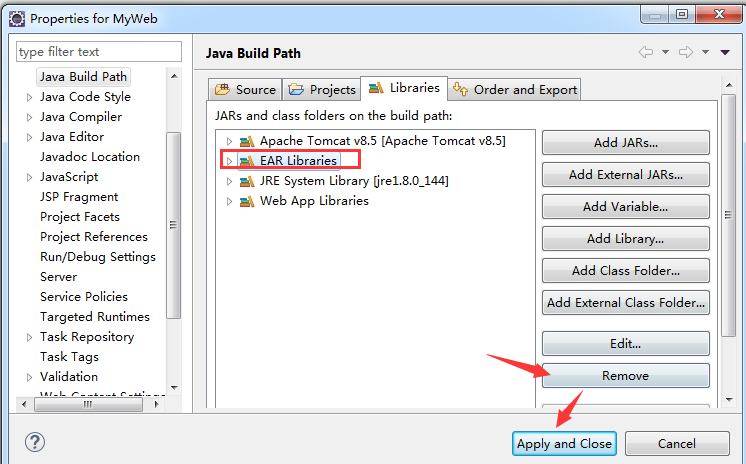
指定Java文件的编译路径（放在WebRoot\WEB-INF\classes）,一般我们编译后的.class文件是放入WEB-INF下，如图：



设置web工程的根目录（也就是部署路径，一般设置成WebRoot），勾选创建web.xml

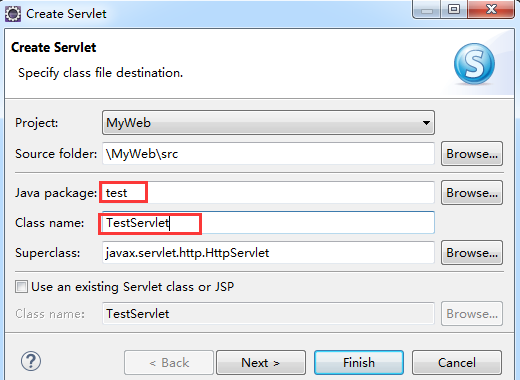


右击工程目录，选择Build Path-Configure Build Path..选项，把EAR删除就好其他的不动

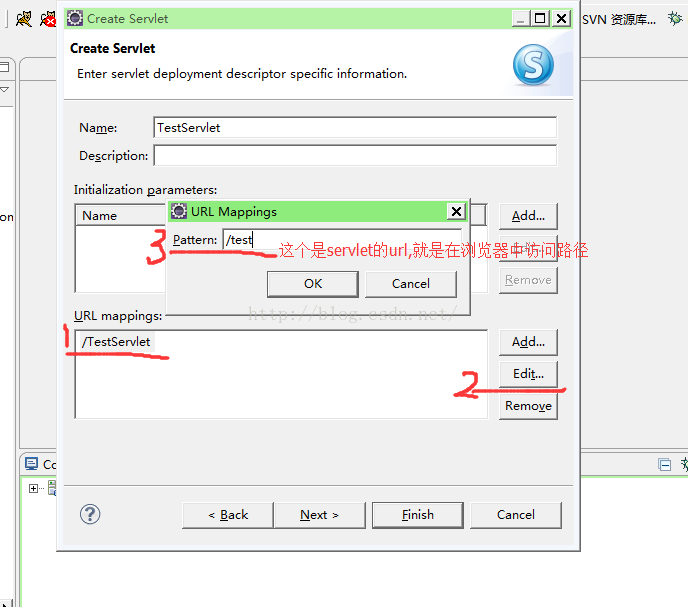


## 16.3测试并部署web项目是否成功

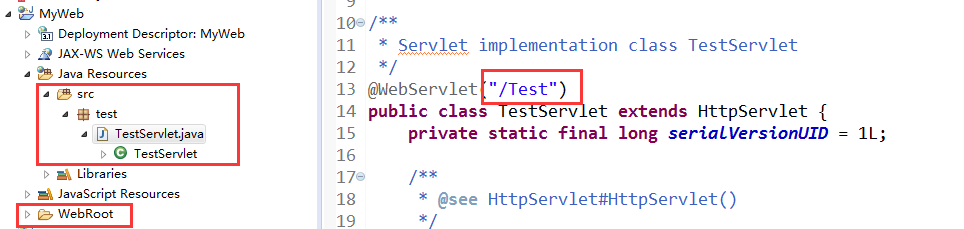
选择MyWeb项目，在src右击new-servlet如下图操作：（我们创建serverlet为例）



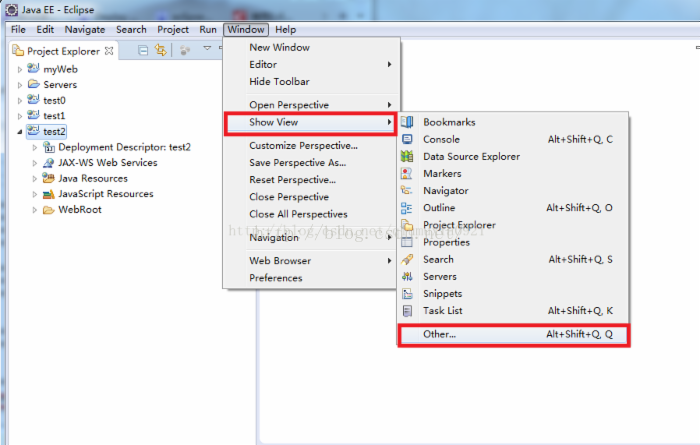
单击下一步进行如下操作：

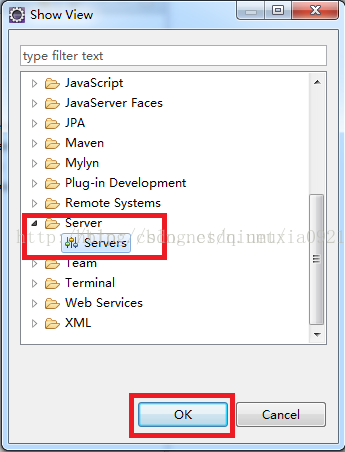


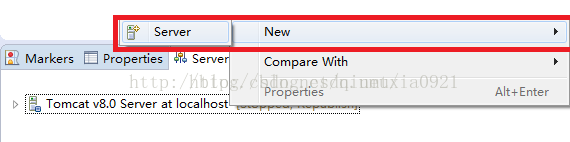
单击next最后finish完成servlet创建，打开创建好的servlet类,在webRoot下创建index.jsp，如下截图

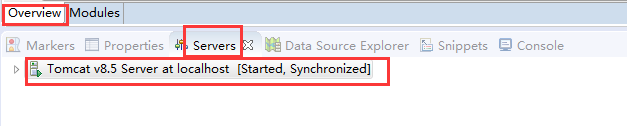


把创建的web项目添加到server中如下操作

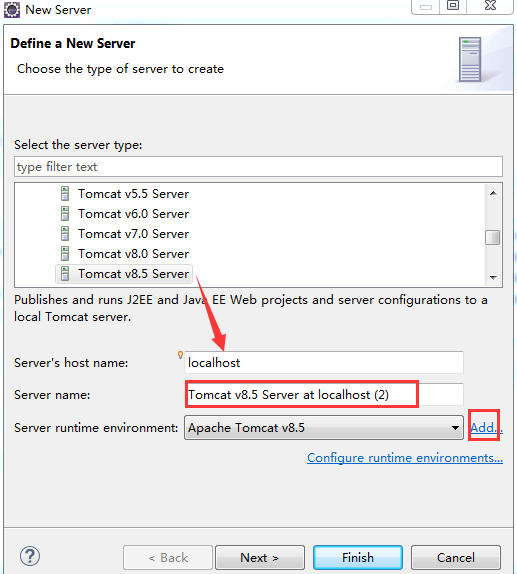


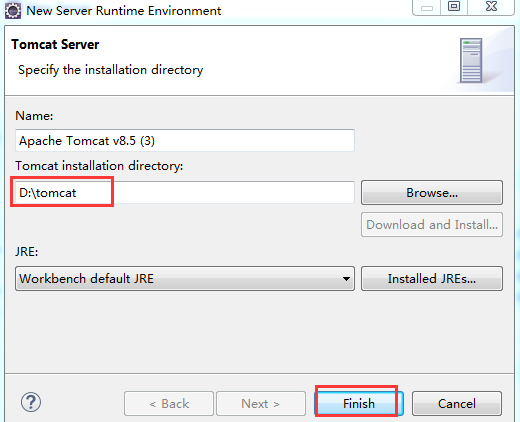






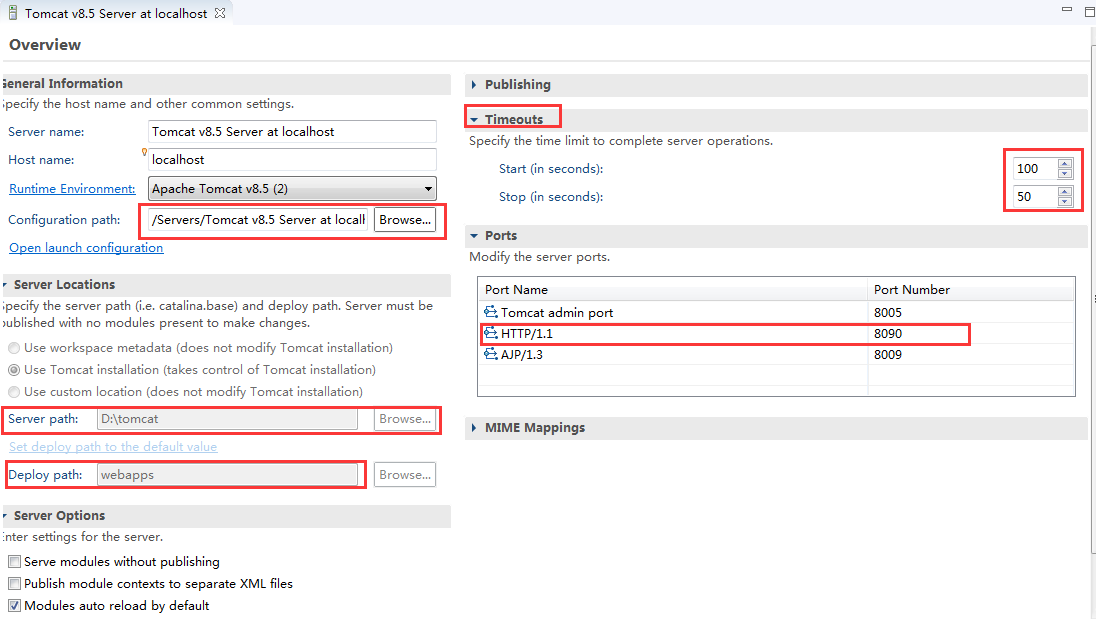
新建一个tomcat服务





到此我们就成功了添加web项目到tomcat服务器上，但是我们还不能直接进行部署，因为eclipse与myeclipse不同的是，项目部署默认不在webapp下，而是在wtpwebapp下，故在部署之前我们先修改部署路径及其他配置，如下截图：

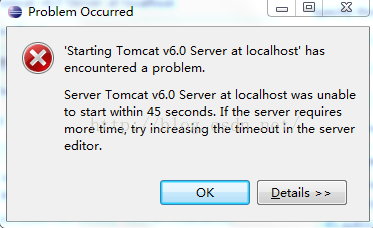
首先进行双击tomcat server打开对话框，或者选中右击Open



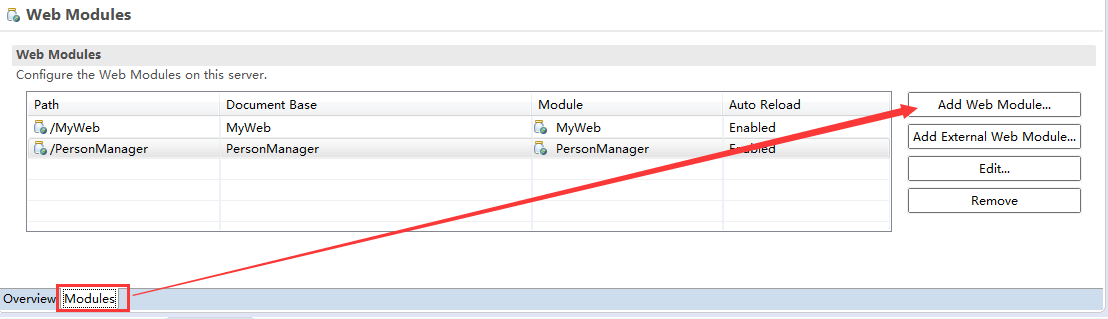
备注：1.若是你发现你根据编辑不了，那么就先移除你添加的web项目，或者删除tomcat server，重新创建，然后设置好如上配置后最后添加web项目

2.Server Locations配置为本地tomcat的目录，这样一来方便，二来可以规避很多问题，修改默认Deploy path为webapp，保证web项目部署到tomcat的webapp下。

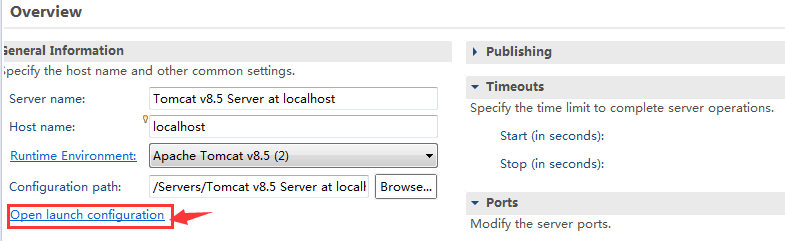
3.响应时间 （timeout默认start45 stop15）start最好调的稍微大点，以防启动时因为时间太长而不成功(根据个人部署，我确实遇到因为启动时间太短tomcat启动不成功),注意有时会跳出一个错误：”

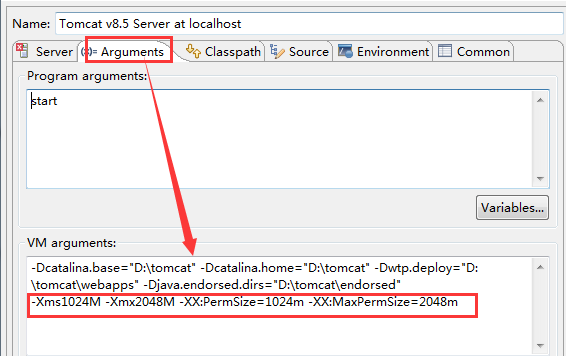


只要把start时间设置大点就不会这样的情况。当这些都配置好以后，记得一定要ctrl+s，否则你的功夫就白费了。当配置完服务器的时候，你会发现你的项目列表中多了个Servers的项目，这个里面存了你的服务的配置文件，你也可以添加多个你创建的web项目，如下图：



最后配置下eclipse的虚拟内存，在日常开发中我们经常遇到这样的的错误，启动tomcat，后台报**"java.lang.OutOfMemoryError: PermGen space"是因为eclipse的虚拟内存太小导致的，如下操作即可：**

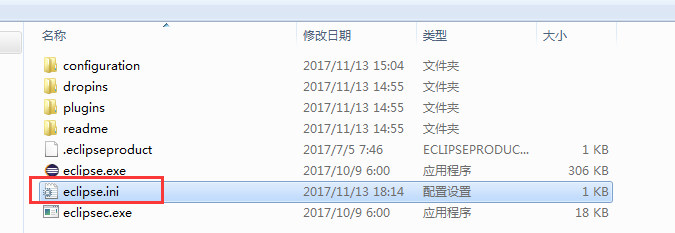




参考：-Xms1024M -Xmx2048M -XX:PermSize=1024m -XX:MaxPermSize=2048m

或者在eclipse的隐藏文件中设置：

-Xms1024M   
-Xmx2048M   
-XX:PermSize=256M   
-XX:MaxNewSize=256M   
-XX:MaxPermSize=512M



启动tomcat，浏览器查看是否正常访问。浏览器：http://localhost:8090/MyWeb