java语言基础
谈到Java语言基础学习的书籍，大家肯定会推荐Bruce Eckel的《Thinking in Java》。它是一本写的相当深刻的技术书籍，Java语言基础部分基本没有其它任何一本书可以超越它。该书的作者Bruce Eckel 在网络上被称为天才的投机者，作者的《Thinking in C++ 》在1995年曾获SoftwareDevelopment Jolt Award最佳书籍大奖，《Thinking in Java 》被评为1999年Java World“最爱读者欢迎图书”，并且赢得了编辑首选图书奖。作者从1986年至今，已经发表了超过150篇计算机技术文章，出版了6本书（其中4本是关于C++的），并且在全世界做了数百次演讲。他是《Thinking in Java 》、《Thinking in C++ 》、《C++ Inside & Out 》《Using C++ 》和《Thinking in Patterns 》的作者，同时还是《Black Belt C++ 》文集的编辑。他的书被读者称为“最好的Java参考书……绝对让人震惊”；“购买Java参考书最明智的选择”；“我见过的最棒的编程指南”。作者的非凡才华，极其跨越语言的能力，使作者被选为Java发展10年间与Java关系最密切的10个人物之一。

《Thinking in Java 》讲述了Java语言的方方面面，很多Java语言的老手都评价“这是一本将Java语言讲得相当丑陋的书”。该书谈及了java语言的很多细节，每一个方面都是相当深刻的。通过本书你可以看到“丑陋的”java语言。

网络上关于java语言讲解的视频很多很多，其中不凡有垃圾。《翁恺—JAVA语言》可能是你学习java语言基础的唯一选择，该讲座基本按照《Thinking in Jav a》这本书讲解，其中不凡有翁老师的很多有意思的笑话。我很幸运学习就是从此视频开始的。内容包括30讲，我总共看了3遍。

不过，对于初学者我不太推荐使用《Thinking in Java 》，我比较推荐Prentice Hall PTR 的《Core Java 2》国内称为《Java 2 核心技术》，目前是第七版。网络上大家都可以下载到电子版。Oreilly的《Java in a nutshell 》也是一个不错的选择。读完以上两本后，你可以看看翁恺老师的视频，接着可以研究《Thinking in Java》了。

2. Java数据结构
市面上关于Java数据结构的书本身就很少很少。大致有APress 的《Java Collections》，Jones 和Bartlett 的《Data Structures in Java 》、《Object-oriented Data Structures Using Java 》以及Prentice Hall 出版的《Data Structures and Algorithms in Java 》(Dec 19, 2005) 还有一本就是《Data Structures And Algorithms With Object-oriented Design Patterns In Java 》。很幸运我的第一本英文书就是APress 的《Java Collections》（本书在国内可能根本就没有中文版－－只能下载英文版了），很不错，讲得很有条例、很简单，是一本完完全全Java Collections API 介绍的书籍，其中不凡有扩展API的例子。这是我推荐你学习java数据结构的唯一一本好书。其它的Jones 和Bartlett的那两本国内好像有一本中文版，想看你也可以看看。

在学习完API后，你可以看看java.util包中对应的类了。不过只有在学习过设计模式后你才有可能完全理解整个Java Collections Framework 。Java Collections Framework 使用了很多著名的设计模式如：迭代器（Iterator）模式，工厂方法模式、装饰器模式、适配器模式等等。通过研究 java.util包中数据结构的源代码，你可以知道臭名昭著的Properties类的设计了，同时可能基本具备设计简单的数据结构的能力了。

所谓学习无止境，学习完Sun提供了Java Collections Framework 后，你可以研究Apche的另一个Java Collections Framework，很有意思哦。互为补充的两个Framework。

在大家学习、研究Java Collections 之前，我提示一下Java Collections主要包括以下三部分：接口（Interface）、实现（Implemention）和算法（Algorithm）。

1. 接口主要有List、Set、Queue和 Map。List 、Se t和Queue是 Collection接口的子接口。

2. 实现主要是实现这些接口的具体类。如实现List接口的ArrayList、LinkedList、Stack和Vector；实现Set接口的 HashSet 、TreeSet 和LinkedHashSet；实现Queue接口的PriorityQueue、SynchronousQueue等等；实现Map接口的HashMap、TreeMap、Hashtable、Properties、WeakHashMap等等。

3. 算法主要是由Arrays类和Collections类提供的，它是整个Java Collection Framework算法的核心。支持各种类型的排序，查找等常用操作。

Java Collections中包含两个版本的数据结构，主要是原先的支持同步的数据结构和后来不支持同步的数据结构。

Java Collection Framework 在使用Comparator和Comparable接口支持排序。同时提供新旧两个版本的迭代器Iterator和Enumeraton，以及它们如何转换等等。

在java.util包中的Obserable接口和Observer类是考察者模式的核心。

……

3. Java IO
市面上关于IO的书籍也仅仅只有Oreilly出版社的两本，都是Elliotte Rusty Harold的著作。两本书的风格基本一致，推荐阅读是第一版的《Jvava I/O》，讲得比较浅显，内容相对比较集中，实例也很多。第二版今年5月国外才出版，很有幸我在网络上下载了第二版，讲得极其详细－－726页的大块头（我化了两个星期），这次将NIO和IO和在一起，还包括J2ME部分的，不过串口、并口通信部分好像类库支持不够，自己不能实际操作。

与第一版的《Jvava I/O》一起的Oreilly还有一本《Jvava NIO》，也是很不错的哦。

大家在依次阅读完《Jvava I/O 》以及《Jvava NIO》后，可以研究java.io包中的源代码了。在大家研究源代码前我给点提示：

Java的io包主要包括：

1. 两种流：字节流（byte Stream）和字符流（character stream），这两种流不存在所谓的谁代替谁、谁比谁高级之说，它们互为补充，只是侧重点不同而已。

2. 两种对称：1.字节流、字符流的对称；2.输入、输出的对称。

3. 一个桥梁：将字节流转变为字符流的InputStreamReader和OutputStreamWriter。

其中必须注意：

1. PipedInputStream和PipedOutputStrem是两个比较有趣的类。

2. 支持Buffered的流是我们经常使用的类。

3. 装饰器（Decorator）模式在java最著名的应用就是用于io的设计。仔细研究各个Filter流与具体流的关系，多看设计模式的书籍。相信你会有所所获。

4. 学习好io包，是研究net包，rmi包……的基础哦！

4 . Java数据库

数据库的书籍太多太多了，也是太烂太烂了！这方面的书我基本都研究过，推荐的你就看看Apress的《JDBC Recipes A Problem Solution Approach 》很不错，国外2005年底才出版，（国内好像没有中文版，不过出了中文版也不一定值得看－－国内经常将国外的书翻译得一塌糊涂、不堪入目）不过我们真的很幸运，网络上有电子版的。值得一看。推荐我看的第一本比较满意的－－Wiley出版的《Java Database Bible 》，讲得很不错！Sun公司自己的关于JDBC API介绍的那一本《JDBC API Tutorial andRefernece》也不错。我第二本JDBC的就是研究的这套API。

不过目前这些书都是一些相对比较浮浅的API应用的书籍。有机会我会给大家带来介绍JDBC API以及JDBC实现内部细节的书！我尽快努力，同时希望得到大家的支持！

顺便给学习JDBC的朋友一点提示：

JDBC的学习和使用主要是这套API，其使用过程也是极其简单，下面是使用JDBC的一般流程：

1. 加载某个数据库的驱动（Driver类），通常使用Class.forName(“驱动的类名“);

2. 连接数据库－－

Connection con = DriverManager.getConnection(url,username,password);

3. 得到会话－－Statement stmt = con.createStatement();

4. 执行操作－－Result rs = stmt.executeQuery(“SQL查询语句”);

5. 处理结果－－

while(rs.next()){

String col1 = rs.getString(1);

……

}

简单吧！整个JDBC中可以变化的一般是：

1. 可以由Connection对象创建Statement、PreparedStatement和CallableStatement创建三种类型的Statement。

2. 可以创建多种类型的ResultSet：支持单向移动和个自由移动；可更新的和不可更新的；支持不同等级的交易的…..

3. 数据输入的批处理。

4. 结果集中特殊类型（Blob、Clob、Arrary和Ref、Struct）列的操作。

5. 这些特殊类型的录入数据库。

6. javax.sql包中特殊结果集（CachedRowSet、JdbcRowSet、WebRowSet）的操作。

7. 其它的就是一个DataSource了，也很简单！一个J2EE中的被管理对象

简单吧！相信大家很快就会征服JDBC。

5. Java 网络编程
网络编程－－一个神秘的、充满挑战的方向。不过在谈Java网络编程之前首先感谢Sun公司的开发人员，因为它们天才的设想，充满智慧的架构，使广大java程序员学习java网络编程变得异常简单。

Java网络编程方面的书，我推荐O'Reilly的《Java Network Programming 》，目前已经第三版了，以前的版本市面上肯定有！网络上早有第三版的电子版，国外2004年出版，706页哦！讲得很全，比较深入，太深入的可能由于Sun有些东西没有完全公开，所以也就不好讲了，有兴趣的可以下载看看！第二本还是O'Reilly 1998 年出版的《Java distributed computing 》，基础部分写得比较详细，后面的实例还是值得研究的。

在大家阅读这些书之前，给大家一点提示：

java网络编程其实相对比较简单，入门也很快很快。java网络编程主要包括两个部分：1.Socket；2.URL部分。不过第二部分也完全建立在第一部分的基础上。

1. Socket包括客户端的Socket和服务器端的ServerSocket。还有就是DatagramSocket和DatagramPacket，它对应于UDP通信协议。 总之，Socket部分是建立其它高级协议的基础。

2. URL类是一个网络资源定位器，通常和具体的网络协议如HTTP，FTP，Telnet……相关。通过该类可以连接网络上的资源，通过其 openStream可以以io包中的流（InputStream）的形式读取网络资源；通过其OpenConnection方法，可以打开一个连接，在此连接上可以不仅可以完成读的操作，还可以完成写的操作。

Java的网络编程大体包括以上两部分。网络编程和IO以及多线程部分非常密切，在学习此部分前大家一定对这两部分了解比较透彻。

学习了以上部分你可以研究java.net包中的与此相关的源代码了！研究所有的源代码还为时尚早。在整个net包中包含：ContentHandlerFactory、URLStreamHandlerFactory、URLStreamHandler、 URLClassLoader等辅助类，它们构成了java.net网络编程的框架，通过研究其源代码，你不仅可以快速理解java.net包，还可以为以后扩展该包打下基础，甚至可以将此思维方式运用到自己的项目中。

到此为止你对java.net包应该才了解60％，还有一部分你可以使用JDecompiler之类的反编译软件打开你JDK安装目录下\jdkxxx\ jre\lib目录中的rt.jar，用WinRAR之类的软件打开它的sun.net包，反编译所有的文件，它是URL类工作的细节。当研究完该 sun.net包，你就会对整个网络编程很熟悉很熟悉了。

一切看起来我们已经对网络编程很精通了。其实不然，刚刚开始而已，要想深入，请继续吧！网络上很多优秀的网络编程库甚至软件可以为我们“添加功力”。如 Apache的HttpCore和HTTPConnection 是两个和HTTP协议相关库；JGroups是研究分布式通信、群组通信的必读库；接着我们可以研究P2P的软件包，如Sun公司的JXTA，它可能是 java平台点对点通信未来的标准哦！接着你可以研究成熟得不得了，使用极其广泛得P2P软件Azureus！www.sourceforge.net可以下载到！

千里之行始于足下！Just do it ！（目前我也只研究了net包，其它的会在不久的将来继续深入。Sun公司因为某些原因没有公开net的其它实现细节，在其允许将其源代码以文字的形式加以研究，以及允许将其没有公开的实现写入书中时，我很希望能出一本java网络编程的书籍，以飨广大读者！！）

6. Servlet和JSP

Servlet、JSP的书也是满地都是！值得推荐的也仅仅两三本。实推Addison Wiley的《Servlets and JavaServer pages ： The J2EE Technology Web Tier》，又是一本很厚的哦！国外2003年出版、784页，讲得比较全，例子也很多，特别是第八章Filter，举了几个不错的例子。其它所有我看到的关于Servlet和JSP的书都没有如此深入的！（可能有我没有看到而已）。O'reilly的《Java Servlet Programming 》和《Java Server Pages》相对比较好懂一些，可以读读！

在大家学习Servlet和Jsp之前我还是要提醒一下：

本质上说Servlet就是一个实现Servlet接口的、部署于服务器端的服务器端的程序罢了！它可以象写其它任何java应用程序一样编写，它可以操作数据库、可以操作本地文件、可以连接本地EJB……编写Servlet程序的一般流程为：

1. 继承一个HttpServlet类；

2. 覆盖其doGet、doPost方法；

3. 在覆盖方法的内部操作方法参数HttpServletRequest和HttpServletResponse。

4. 读取请求利用HttpServletRequest。利用HttpServletRequest你可以操作Http协议的协议头、可以得到请求的操作方法、可以得到请求的路径、可以得到请求的字符串、以及和请求客户相关的信息，更主要的你可以得到Cookie和HttpSession这两个对象。

5. 利用Cookie你可以操作“甜心”对象或者将其写入HttpServletResponse中。

6. 向客户输出信息可以使用HttpServletResponse。使用HttpServletResponse可以写入各种类型的协议头、可以增加Cookie、可以重定向其它URL、可以向客户发送Http协议的状态码。

7. 利用HttpSession在会话内完成你想实现的任何功能。

同时Servlet还提供了一些事件和事件监听器（简单的观察者模式而已）。还有就是过滤器（Filter）和包装器（ServletRequestWrapper、ServletResponseWrapper）——简单的流的使用和装饰器模式的使用。

学习Sevlet、JSP必然要部署到服务器中，记住通常文件部署的步骤和参数的设置以及在程序中如何使用就可以了。

完全理解Servlet后，学习jsp相对比较容易了！Jsp完全建立在Servlet的基础上，它是为了迎合那些喜欢在Html文档中嵌入脚本（如：PHP之类的网页编程语言）的程序员的需要罢了！学起来也相当的容易！

一切看起来似乎那么的风平浪静，简单好学！简单的表象背后有其复杂的机理。要想对Servlet和Jsp彻底研究，你得研究Tomcat等开源软件的具体实现。它无非就是一个服务器，在客户利用网页通过HTTP协议向服务器发送请求后，服务器将此HTTP请求转化为相应的 HttpServletRequest 对象，调用你编写的Servlet罢了，在你的Servlet中你肯定操作了此HttpServletRequest了吧，同时操作了HttpServletResponse了吧，服务器就将此HttpServletResponse按照HTTP协议的要求利用HTTP协议发送给你的浏览器了！在服务器端的Jsp网页在被客户请求后， Tomcat会利用编译软件，使用javax.servlet.jsp包中的模板，编译此jsp文件，编译后就是一个Servlet！以后的操作和 Servlet完全一样哦！

在Servlet和Jsp的基础上出现了，所谓的高级技术：JSTL，Struts……无非就是一些标签和MVC模式的使用。

继续前进吧！胜利就在前方！！

7. 多线程
一个看起来很神秘，却很容易上手、很难精通的方向！

我推荐两本我感觉很好的书籍。首先是我第一本能上手看的这方面的书，Sams 1998年出版的《Java Thread Programming 》，写得暴好，很容易读懂，我有空还时常看当时的笔记！要知道怎么好你自己看吧！第二本OReilly三次出版的《Java Threads》，最新是2004版，国内好像有中文版，推荐你还是看英文版的吧！书中谈到了与多线程相关的N个方向，如IO、Swing、 Collection等等。

给大家一点提示吧！java类库中与多线程相关的类不是很多，主要有：Thread、ThreadGroup以及ThreadLocal和 InheritableThreadLocal四个类和一个Runnable接口；关键字synchronize、volatile ；以及Object对象的wait、notify、notifyAll方法！

1 Thread是多线程的核心类，提供了一系列创建和操作多线程的方法。

2 ThreadGroup是一个管理Thread的工具类。

3 ThreadLocal和InheritableThreadLocal为Thread提供了一个类似保险箱功能的存储线程对象的类！

4 Runnable不用说了吧！

5 synchronize是同步方法和同步块的核心哦！多个线程调用此方法时，只有一个线程可以使用此方法，其它方法阻塞，从而保证被操作对象内部状态完整性。某个线程调用带有synchronize的方法或块时会得到该对象的对象锁，完成块中的操作后释放此对象锁，从而其它对象可以继续操作。

6 wait、notify、notifyAll提供了有效的等待/通知机制。Java语言中每一个对象都有一个休息室，任何线程在其操作的对象的状态不满足的情况下，在该对象的休息室中休息，释放对象锁；当其它线程操作该对象后，唤醒休息室中的线程，它们再检查条件，当条件满足后，执行相应的操作。

多线程大致就这么多基础的！简单吗！这对于一个真正的程序员应该是不够的，真正对多线程要有所掌握，请您研究java.util.concurrent包吧！大师Doug Lea的作品，原先是一个开源的一致性编程的库，后来被Sun公司并入java类库。作者的网站上也有另外一个版本的该类库！值得研究的好东西! Hibernation、OpenJMS等开源软件都使用了此包！

8. 设计模式
谈到设计模式很多人多会推荐GOF的那本，该书在Amzon上是五星级的推荐书籍。不过对于学习java没多久的、特别是java初学者，我很不推荐这本书。主要是该书的例子基本都是C++的，很多细节没有讲述得足够清楚。

我给大家推荐的第一本是阎宏博士的《Java 与模式》，它是第一本中国人自己写的关于设计模式的书籍，写的比较有趣，融合了很多中华民族的文化和观念，例子、类图都比较多，且相对简单！非常不错的入门书籍－－又是大块头哦！

其次我推荐Wiley出版社出版的《Pattern In Java 》一套三本，我才看了第一本，好像第二本不怎么样，第三本还不错！

第三本是中文翻译版的关于多线程模式的（很难得的中文翻译版）中国铁道出版社2003年出版的《Java多线程设计模式》，将多线程模式讲得非常浅显，配有大量的图例，每章都有习题，最后有答案！我研究多线程模式就是由它开始的！

第四本，今年出版的Head First 系列的《Head First Design Pattern 》，秉承Head First系列图书的优点，大量的类图、丰富的实例、有趣的注解，值得购买！

其次在J2EE方向你可以研究阅读Addison Wesley 2002 年出版的《Patterns of Enterprise Application Architecture 》，众多大腕的作品，讲企业消息集成的！Sun提供的《J2EE PATTERNS SL500 》也很好！晚了推荐那一本Amzon 4 星半的《Holub on patterns 》，大师的作品，提供了，很值得研究的例子，不过对上面四本不是很熟悉的读者，最好不要读它！可能会让你比较累！

我学习设计模式经过一段很曲折的路线，前前后后大约看了20本，阎宏博士的《Java 与模式》我看了4遍，还排除我第一次基本没看懂的看！记得研一时老师给我们讲了GOF的那本，作为选修课，我和它们计算机系的硕士、博士们一起，到最后一个班40－50个人，不超过3个人明白，我也没有明白任何一点（基础差吧－－主要我对C++语言一点都不了解），凭我不伏输的性格，我认为我对java语言理解还可以，我就借了《Java 与模式》，结果还是基本没看懂。很有幸的是读研三时，听过了上交大饶若楠老师关于Java OOP语言的讲座，我懂了组合书籍模式等三种设计模式后，对其它模式有了强烈的兴趣和要征服它的愿望！工作后我买的第一本就是《Java 与模式》，第一遍花了2个月研究了这个1000多页的大块头，后来第三遍15天左右就可以搞定，笔记记了一大本！从此一发不可收拾。

选对书、埋头研究。相信很快就会入门的！

学习Java语言8个简单的部分，这只是我们研究Java语言的开始！这些都懂了充其量一个java程序员而已，后面的路很长很长！我们可以继续研究数据库实现的源代码、Servlet服务器的源代码、RMI、EJB、JNDI、面向方面编程、重构、ANT工具、Eclipse工具、Spring工具、 JBoss 、JOnAS、Apache Geronimo等J2EE服务器！研究了这些你可能会成为一个出色的J2EE Architecture！你可以继续研究剖析器、编译器、JNODE（java写的操作系统）……

Java Learning Path （一）、工具篇

一、 JDK (Java Development Kit)

JDK是整个Java的核心，包括了Java运行环境（Java Runtime Envirnment ），一堆Java工具和Java基础的类库(rt.jar)。不论什么Java应用服务器实质都是内置了某个版本的JDK。因此掌握JDK是学好Java的第一步。最主流的JDK是Sun公司发布的JDK，除了Sun之外，还有很多公司和组织都开发了自己的JDK，例如IBM公司开发的JDK，BEA公司的Jrocket，还有GNU组织开发的JDK等等。其中IBM的JDK包含的JVM（Java Virtual Machine）运行效率要比Sun JDK包含的JVM高出许多。而专门运行在x86平台的Jrocket在服务端运行效率也要比Sun JDK好很多。但不管怎么说，我们还是需要先把Sun JDK掌握好。

1、 JDK的下载和安装
JDK又叫做J2SE（Java2 SDK Standard Edition ），可以从Sun的Java网站上下载到，http://java.sun.com/j2se/downloads.html ，JDK当前最新的版本是J2SDK1.4.2，建议下载该版本的JDK，下载页面在这里：http://java.sun.com/j2se/1.4.2/download.html。

下载好的JDK是一个可执行安装程序，默认安装完毕后会在C:\Program Files\Java\目录下安装一套JRE（供浏览器来使用），在C:\j2sdk1.4.2下安装一套JDK（也包括一套JRE）。然后我们需要在环境变量PATH的最前面增加java的路径C:\j2sdk1.4.2\bin。这样JDK就安装好了。

2、 JDK的命令工具
JDK的最重要命令行工具：

java： 启动JVM执行class
javac： Java编译器
jar： Java打包工具
javadoc： Java文档生成器
这些命令行必须要非常非常熟悉，对于每个参数都要很精通才行。对于这些命令的学习，JDK Documentation上有详细的文档。

二、 JDK Documentation

Documentation在JDK的下载页面也有下载连接，建议同时下载Documentation。Documentation是最最重要的编程手册，涵盖了整个Java所有方面的内容的描述。可以这样说，学习Java编程，大部分时间都是花在看这个Documentation上面的。我是随身携带的，写Java代码的时候，随时查看，须臾不离手。

三、 应用服务器(App Server)

App Server 是运行Java企业组件的平台，构成了应用软件的主要运行环境。当前主流的App Server是BEA公司的Weblogic Server和IBM公司的Websphere以及免费的Jboss，选择其中一个进行学习就可以了，个人推荐Weblogic，因为它的体系结构更加干净，开发和部署更加方便，是Java企业软件开发人员首选的开发平台。下面简要介绍几种常用的App Server：

1、 Tomcat

Tomcat严格意义上并不是一个真正的App Server，它只是一个可以支持运行Serlvet/JSP的Web容器，不过Tomcat也扩展了一些App Server的功能，如JNDI，数据库连接池，用户事务处理等等。Tomcat被非常广泛的应用在中小规模的Java Web应用中，因此本文做一点下载、安装和配置Tomcat的介绍：

Tomcat是Apache组织下Jakarta项目下的一个子项目，它的主网站是：http://jakarta.apache.org/tomcat/ ，Tomcat最新版本是Tomcat4.1.27，软件下载的连接是：http://www.apache.org/dist/jakarta/tomcat-4/binaries/ 。

下载Tomcat既可以直接下载zip包，也可以下载exe安装包（个人建议zip更干净些），不管哪种情况，下载完毕安装好以后（zip直接解压缩就可以了）。需要设置两个环境变量：

JAVA_HOME=C:\j2sdk1.4.2
CATALINA_HOME=D:\tomcat4 (你的Tomcat安装目录)

这样就安装好了，启动Tomcat运行CATALINA_HOME\bin\startup.bat，关闭Tomcat运行shutdown.bat脚本。Tomcat启动以后，默认使用8080端口，因此可以用浏览器访问http://localhost:8080来测试Tomcat是否正常启动。

Tomcat提供了两个Web界面的管理工具，URL分别是：
http://localhost:8080/admin/index.jsp
http://localhost:8080/manager/html
在启用这两个管理工具之前，先需要手工配置一下管理员用户和口令。用一个文本工具打开CATALINA_HOME\conf\tomcat-users.xml这个文件，加入如下几行：

```
&lt;role rolename="manager"/&gt;
&lt;role rolename="admin"/&gt;
&lt;user username="robbin" password="12345678" roles="admin,manager,tomcat"/&gt;
```

这样用户“robbin”就具备了超级管理员权限。重新启动Tomcat以后，你就可以使用该用户来登陆如上的两个管理工具，通过Web方式进行Tomcat的配置和管理了。

2、 BEA Weblogic
Weblogic可以到BEA的网站上免费注册之后下载到最新的Weblogic8.1企业版，License可以免费使用1年时间，其实这已经完全足够了。Weblogic的下载连接：http://commerce.bea.com/index.jsp，Weblogic的在线文档：http://edocs.bea.com/ 。

3、 IBM Webshpere
Websphere同样可以下载到免费的试用版本，到IBM的developerWorks网站可以看到Websphere试用产品的下载和相关的Websphere的资料，developerWorks中文网站的连接是：http://www-900.ibm.com/developerWorks/cn/wsdd/ ，Websphere的下载连接：http://www7b.software.ibm.com/wsdd/downloads/WASsupport.html 。

4、 Jboss
Jboss是免费开源的App Server，可以免费的从Jboss网站下载：http://www.jboss.org/index.html，然而Jboss的文档是不免费，需要花钱购买，所以为我们学习Jboss设置了一定的障碍。在Jdon上有几篇不错的Jboss配置文档，可以用来参考：http://www.jdon.com/idea.html

四、 Java应用的运行环境

Java的应用可以简单分为以下几个方面：

1、 Java的桌面应用
桌面应用一般仅仅需要JRE的支持就足够了。

2、 Java Web应用
Java的Web应用至少需要安装JDK和一个web容器（例如Tomcat），以及一个多用户数据库，Web应用至少分为三层：
Browser层：浏览器显示用户页面
Web层：运行Servlet/JSP
DB层：后端数据库，向Java程序提供数据访问服务

3、 Java企业级应用
企业级应用比较复杂，可以扩展到n层，最简单情况会分为4层：
Browser层：浏览器显示用户页面
Client层：Java客户端图形程序（或者嵌入式设备的程序）直接和Web层或者EJB层交互
Web层：运行Servlet/JSP
EJB层：运行EJB，完成业务逻辑运算
DB层：后端数据库，向Java程序提供数据访问服务

4、 Java嵌入式应用
Java嵌入式应用是一个方兴未艾的领域，从事嵌入式开发，需要从Sun下载J2ME开发包，J2ME包含了嵌入式设备专用虚拟机KVM，和普通的JDK中包含的JVM有所不同。另外还需要到特定的嵌入式厂商那里下载模拟器。

Java Learning Path（二）、书籍篇

学习一门新的知识，不可能指望只看一本，或者两本书就能够完全掌握。需要有一个循序渐进的阅读过程。我推荐Oreilly出版的Java系列书籍。

在这里我只想补充一点看法，很多人学习Java是从《Thinking in Java》这本书入手的，但是我认为这本书是不适合初学者的。我认为正确的使用这本书的方法应该是作为辅助的读物。《Thinking in Java 》并不是在完整的介绍Java的整个体系，而是一种跳跃式的写作方法，是一种类似tips的方法来对Java很多知识点进行了深入的分析和解释。

对于初学者来说，最好是找一本Java入门的书籍，但是比较完整的循序的介绍Java的语法，面向对象的特性，核心类库等等，在看这本书的同时，可以同步来看《Thinking in Java》，来加深对Java的理解和原理的运用，同时又可以完整的了解Java的整个体系。

对于Java的入门书籍，蔡学镛推荐的是Oreilly的《Exploring Java, 2nd Edition 》 或者《Java in a Nutshell,2nd Edition （针对C++背景）》，我并没有看过这两本书。其实我觉得电子工业出版社的《Java 2编程详解》或者《Java 2从入门到精通》就很不错。

在所有的Java书籍当中，其实最最有用的，并不是O'reilly的 Java Serials ，真正最最有用处是JDK的Documentation！几乎你想获得的所有的知识在Documentation里面全部都有，其中最主要的部分当然是Java基础类库的API文档，是按照package来组织的，对于每一个class都有详细的解释，它的继承关系，是否实现了某个接口，通常用在哪些场合，还可以查到它所有的public的属性和方法，每个属性的解释，意义，每个方法的用途，调用的参数，参数的意义，返回值的类型，以及方法可能抛出的异常等等。可以这样来说，所有关于Java编程方面的书籍其实都不过是在用比较通俗易懂的语言，和良好的组织方式来介绍Documentation里面的某个package里面包含的一些类的用法而已。所以万变不离其宗，如果你有足够的能力来直接通过Documentation来学习Java的类库，那么基本上就不需要看其他的书籍了。除此之外，Documentation也是编程必备的手册，我的桌面上有三个Documentation的快捷方式，分别是J2SDK1.4.1的Documentation，Servlet2.3的Documentation和J2SDKEE1.3.1的Documentation。有了这个三个Documentation，什么其他的书籍都不需要了。

对于Java Web 编程来说，最核心的是要熟悉和掌握HTTP协议，这个就和Java无关了，在熟悉HTTP协议之后，就需要熟悉Java的实现HTTP协议的类库，也就是Servlet API，所以最重要的东西就是Servlet API 。当然对于初学者而言，直接通过Servlet API来学习Web编程有很大的难度，我推荐O'reilly的《Java Server Pages 》这本书来学习Web 编程。

EJB的书籍当中，《Enterprise JavaBeans, 2nd Edition 》是一本很不错的书， EJB的学习门槛是比较高，入门很难，但是这本书完全降低了学习的难度，特别重要的一点是，EJB的学习需要结合一种App Server 的具体实现，所以在学习EJB的同时，必须同步的学习某种App Server，而这本书相关的出了三本书，分别是Weblogic6.1，Websphere4.0和JBoss3.0上面部署书中例子的实做。真是既有理论，又有实践。在学习EJB的同时，可以边看边做，EJB的学习会变得很轻松。

但是这本书也有一个问题，就是版本比较旧，主要讲EJB1.1规范和部分EJB2.0的规范。而Ed Roman 写的《Mastering EJB 2.0 》这本书完全是根据EJB2.0规范写的，深入浅出，覆盖了EJB编程的各个方面，并且还有很多编程经验tips，也是学习EJB非常推荐的书籍之一。

如果是结合Weblogic来学习J2EE的话，《J2EE应用与BEA Weblogic Server 》绝对是首选读物，虽然是讲述的Weblogic6.0，仍然值得购买，这本书是BEA官方推荐的教材，作者也是BEA公司的工程师。现在中文版已经随处可见了。这本书结合Weblogic介绍了J2EE各个方面的技术在Weblogic平台上的开发和部署，实践指导意义非常强。

在掌握了Java平台基础知识和J2EE方面的知识以后，更进一步的是学习如何运用OO的方法进行软件的设计，那么就一定要学习“设计模式”。Sun公司出版了一本《J2EE核心模式》，是每个开发Java企业平台软件的架构师必备的书籍。这本书全面的介绍了J2EE体系架构的各种设计模式，是设计师的必读书籍。

Java Learning Path（三）过程篇

每个人的学习方法是不同的，一个人的方法不见得适合另一个人，我只能是谈自己的学习方法。因为我学习Java是完全自学的，从来没有问过别人，所以学习的过程基本上完全是自己摸索出来的。我也不知道这种方法是否是比较好的方法，只能给大家提供一点参考了。

学习Java的第一步是安装好JDK，写一个Hello World ， 其实JDK的学习没有那么简单，关于JDK有两个问题是很容易一直困扰Java程序员的地方：一个是CLASSPATH的问题，其实从原理上来说，是要搞清楚JRE的ClassLoader是如何加载Class的；另一个问题是package和import问题，如何来寻找类的路径问题。把这两个问题摸索清楚了，就扫除了学习Java和使用JDK的最大障碍。推荐看一下王森的《Java深度历险》，对这两个问题进行了深入的探讨。

第二步是学习Java的语法。Java的语法是类C++的，基本上主流的编程语言不是类C，就是类C++的，没有什么新东西，所以语法的学习，大概就是半天的时间足够了。唯一需要注意的是有几个不容易搞清楚的关键字的用法，public，protected，private，static，什么时候用，为什么要用，怎么用，这可能需要有人来指点一下，我当初是完全自己琢磨出来的，花了很久的时间。不过后来我看到《Thinking in Java》这本书上面是讲了这些概念的。

第三步是学习Java的面向对象的编程语言的特性的地方。比如继承，构造器，抽象类，接口，方法的多态，重载，覆盖，Java的异常处理机制。对于一个没有面向对象语言背景的人来说，我觉得这个过程需要花很长很长时间，因为学习Java之前没有C++的经验，只有C的经验，我是大概花了一个月左右吧，才彻底把这些概念都搞清楚，把书上面的例子反复的揣摩，修改，尝试，把那几章内容反复的看过来，看过去，看了不下5遍，才彻底领悟了。不过我想如果有C++经验的话，应该一两天时间足够了。那么在这个过程中，可以多看看《Thinking in Java 》这本书，对面向对象的讲解非常透彻。可惜的是我学习的时候，并没有看到这本书，所以自己花了大量的时间，通过自己的尝试和揣摩来学会的。

第四步就是开始熟悉Java的类库。Java的基础类库其实就是JDK安装目录下面jre\lib\rt.jar这个包。学习基础类库就是学习rt.jar。基础类库里面的类非常非常多。据说有3000多个，我没有统计过。但是真正对于我们来说最核心的只有4个，分别是
java.lang.*;
java.io.*;
java.util.*;
java.sql.*;

这四个包的学习，每个包的学习都可以写成一本厚厚的教材，而O'reilly也确实是这样做的。我觉得如果时间比较紧，是不可能通过读四本书来学习。我觉得比较好的学习方法是这样的：
首先要通读整个package的框架，了解整个package的class，interface，exception的构成，最好是能够找到介绍整个包框架的文章。这些专门介绍包的书籍的前几章应该就是这些总体的框架内容介绍。

对包整体框架的把握并不是要熟悉每个类的用法，记住它有哪些属性，方法。想记也记不住的。而是要知道包有哪些方面的类构成的，这些类的用途是什么，最核心的几个类分别是完成什么功能的。我在给人培训的时候一般是一次课讲一个包，所以不可能详细的介绍每个类的用法，但是我反复强调，我给你们讲这些包的不是要告诉你们类的方法是怎么调用的，也不要求你们记住类的方法调用，而是要你们了解，Java给我们提供了哪些类，每个类是用在什么场合，当我遇到问题的时候，我知道哪个类，或者哪几个类的组合可以解决我的问题，That'all！，当我们具体写程序的时候，只要你知道该用哪个类来完成你的工作就足够了。编码的时候，具体的方法调用，是边写代码，边查Documentation，所有的东西都在Documentation里面，不要求你一定记住，实际你也记不住3000多个类的总共将近10万个方法调用。所以对每个包的总体框架的把握就变得极为重要。

第五步，通过上面的学习，如果学的比较扎实的话，就打好了Java的基础了，剩下要做的工作是扫清Documentation里面除了上面4个包之外的其他一些比较有用处的类。相信进展到这一步，Java的自学能力已经被培养出来了，可以到了直接学习Documentation的水平了。除了要做GUI编程之外，JDK里面其他会有用处的包是这些：
java.text.*;
java.net.*;

javax.naming.*;
这些包里面真正用的比较多的类其实很少，只有几个，所以不需要花很多时间。

第六步，Java Web 编程
Web编程的核心是HTTP协议，HTTP协议和Java无关，如果不熟悉HTTP协议的话，虽然也可以学好Servlet/JSP编程，但是达不到举一反三，一通百通的境界。所以HTTP协议的学习是必备的。如果熟悉了HTTP协议的话，又有了Java编程的良好的基础，学习Servlet/JSP简直易如反掌，我学习Servlet/JSP就用了不到一周的时间，然后就开始用JSP来做项目了。

在Servlet/JSP的学习中，重头仍然是Servlet Documentation。Servlet API最常用的类很少，花比较少的时间就可以掌握了。把这些类都看一遍，多写几个例子试试。Servlet/JSP编程本质就是在反复调用这些类来通过HTTP协议在Web Server和Brower之间交谈。另外对JSP，还需要熟悉几个常用JSP的标记，具体的写法记不住的话，临时查就是了。

此外Java Web 编程学习的重点要放在Web Application的设计模式上，如何进行业务逻辑的分析，并且进行合理的设计，按照MVC设计模式的要求，运用Servlet和JSP分别完成不同的逻辑层，掌握如何在Servlet和JSP之间进行流程的控制和数据的共享，以及Web Application应该如何配置和部署。

第七步，J2EE编程
以上的学习过程如果是比较顺利的话，进行到这一步，难度又陡然提高。因为上面的知识内容都是只涉及一个方面，而像EJB，JMS，JTA等核心的J2EE规范往往是几种Java技术的综合运用的结晶，所以掌握起来难度比较大。

首先一定要学习好JNDI，JNDI是App Server定位服务器资源（EJB组件，Datasouce，JMS）查找方法，如果对JNDI不熟悉的话，EJB，JMS这些东西几乎学不下去。JNDI其实就是javax.naming.*这个包，运用起来很简单。难点在于服务器资源文件的配置。对于服务器资源文件的配置，就需要看看专门的文档规范了，比如web.xml的写法，ejb-jar.xml的写法等等。针对每种不同的App Server，还有自己的服务资源配置文件，也是需要熟悉的。

然后可以学习JTA，主要是要理解JTA对于事务的控制的方法，以及该在什么场合使用JTA。这里可以简单的举个例子，我们知道一般情况可以对于一个数据库连接进行事务控制(conn.setAutoCommit(false),....,conn.commit())，做为一个原子操作，但是假设我的业务需求是要把对两个不同数据库的操作做为一个原子操作，你能做的到吗？这时候只能用JTA了。假设操作过程是先往A数据库插一条记录，然后删除B数据库另一个记录，我们自己写代码是控制不了把整个操作做为一个原子操作的。用JTA的话，由App Server来完成控制。

在学习EJB之前要学习对象序列化和RMI，RMI是EJB的基础。接着学习JMS和EJB，对于EJB来说，最关键是要理解EJB是如何通过RMI来实现对远端对象的调用的，以及在什么情况下要用到EJB。

在学习完EJB，JMS这些东西之后，你可能会意识到要急不可待学习两个领域的知识，一个是UML，另一个是Design Pattern。Java企业软件的设计非常重视框架(Framework)的设计，一个好的软件框架是软件开发成功的必要条件。在这个时候，应该开始把学习的重点放在设计模式和框架的学习上，通过学习和实际的编程经验来掌握EJB的设计模式和J2EE的核心模式。

J2EE规范里面，除了EJB，JMS，JTA，Servlet/JSP，JDBC之外还有很多很多的企业技术，这里不一一进行介绍了。

另外还有一个最新领域Web Services 。Web Services也完全没有任何新东西，它像是一种黏合剂，可以把不同的服务统一起来提供一个统一的调用接口，作为使用者来说，我只要获得服务提供者给我的WSDL（对服务的描述），就够了，我完全不知道服务器提供者提供的服务究竟是EJB组件，还是.Net组件，还是什么CORBA组件，还是其他的什么实现，我

也不需要知道。Web Services最伟大的地方就在于通过统一的服务提供方式和调用方式，实现了整个Internet服务的共享，是一个非常令人激动的技术领域。Web Services好像目前还没有什么很好的书籍，但是可以通过在网络上面查资料的方式来学习。

Java Learning Path（四） 方法篇

Java作为一门编程语言，最好的学习方法就是写代码。当你学习一个类以后，你就可以自己写个简单的例子程序来运行一下，看看有什么结果，然后再多调用几个类的方法，看看运行结果，这样非常直观的把类给学会了，而且记忆非常深刻。然后不应该满足把代码调通，你应该想想看如果我不这样写，换个方式，再试试行不行。记得哪个高人说过学习编程就是个破坏的过程，把书上的例子，自己学习Documentation编写的例子在运行通过以后，不断的尝试着用不同的方法实现，不断的尝试破坏代码的结构，看看它会有什么结果。通过这样的方式，你会很彻底的很精通的掌握Java。

举个例子，我们都编过Hello World

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

很多初学者不是很理解为什么main方法一定要这样来定义public static void main(String[] args)，能不能不这样写？包括我刚学习Java的时候也有这样的疑问。想知道答案吗？很简单，你把main改个名字运行一下，看看报什么错误，然后根据出错信息进行分析；把main的public取掉，在试试看，报什么错误；static去掉还能不能运行；不知道main方法是否一定要传一个String[]数组的，把String[]改掉，改成int[]，或者String试试看；不知道是否必须写args参数名称的，也可以把args改成别的名字，看看运行结果如何。

我当初学习Java的时候就是这样做的，把Hello World程序反复改了七八次，不断运行，分析运行结果，最后就彻底明白为什么了main方法是这样定义的了。

此外，我对于staic，public，private，Exception，try{ }catch {}finally{} 等等等等一开始都不是很懂，都是把参考书上面的例子运行成功，然后就开始破坏它，不断的根据自己心里面的疑问来重新改写程序，看看能不能运行，运行出来是个什么样子，是否可以得到预期的结果。这样虽然比较费时间，不过一个例子程序这样反复破坏几次之后。我就对这个相关的知识彻底学通了。有时候甚至故意写一些错误的代码来运行，看看能否得到预期的运行错误。这样对于编程的掌握是及其深刻的。

其中特别值得一提的是JDK有一个非常棒的调试功能，-verbose
java –verbose
javac –verbose 以及其它很多JDK工具都有这个选项
-verbose 可以显示在命令执行的过程中，JVM都依次加载哪里Class，通过这些宝贵的调试信息，可以帮助我们分析出JVM在执行的过程中都干了些什么。

另外，自己在学习过程中，写的很多的这种破坏例程，应该有意识的分门别类的保存下来，在工作中积累的典型例程也应该定期整理，日积月累，自己就有了一个代码库了。遇到类似的问题，到代码库里面 Copy & Paste ， Search & Replace，就好了，极大提高了开发速度。最理想的情况是把一些通用的例程自己再抽象一层，形成一个通用的类库，封装好。那么可复用性就更强了。

所以我觉得其实不是特别需要例程的，自己写的破坏例程就是最好的例子，如果你实在对自己写的代码不放心的话，我强烈推荐你看看JDK基础类库的Java源代码。在JDK安装目录下面会有一个src.zip，解开来就可以完整的看到整个JDK基础类库，也就是rt.jar的Java源代码，你可以参考一下Sun是怎么写Java程序的，规范是什么样子的。我自己在学习Java的类

库的时候，当有些地方理解的不是很清楚的时候，或者想更加清晰的理解运作的细节的时候，往往会打开相应的类的源代码，通过看源代码，所有的问题都会一扫而空。

Java Learning Path（五）资源篇

1、 http://java.sun.com/ (英文)
Sun的Java网站，是一个应该经常去看的地方。不用多说。

2、http://www-900.ibm.com/developerWorks/cn/
IBM的developerWorks网站，英语好的直接去英文主站点看。这里不但是一个极好的面向对象的分析设计网站，也是Web Services，Java，Linux极好的网站。强烈推荐！！！

3、http://www.javaworld.com/ (英文)
关于Java很多新技术的讨论和新闻。想多了解Java的方方面面的应用，这里比较好。

4、http://dev2dev.bea.com.cn/index.jsp
BEA的开发者园地，BEA作为最重要的App Server厂商，有很多独到的技术，在Weblogic上做开发的朋友不容错过。

5、http://www.huihoo.com/
灰狐动力网站，一个专业的中间件网站，虽然不是专业的Java网站，但是在J2EE企业应用技术方面有深厚的造诣。

6、http://www.theserverside.com/home/ (英文)
TheServerSide是一个著名的专门面向Java Server端应用的网站。

7、http://www.javaresearch.org/
Java研究组织，有很多优秀的Java方面的文章和教程，特别是在JDO方面的文章比较丰富。

8、http://www.cnjsp.org/
JSP技术网站，有相当多的Java方面的文章和资源。

9、http://www.jdon.com/
Jdon论坛，是一个个人性质的中文J2EE专业技术论坛，在众多的Java的中文论坛中，Jdon一个是技术含量非常高，帖子质量非常好的论坛。

10、http://sourceforge.net/
SourgeForge是一个开放源代码软件的大本营，其中也有非常非常丰富的Java的开放源代码的著名的软件。