钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：jin.qian.canada@gmail.com钱老师报名、答疑微信号：qianjincanada，或扫描以下二维码添加：

钱老师

加拿大

扫一扫上面的二维码图案，加我微信

Java高级速成班

本课程针对有一定编成基础，希望在短时间内对Java企业级开发有所了解的同学。

通过本课程的学习，学员能够顺利掌握J2EE的体系结构，了解常见的Java框架并熟练使用，具体内容包括:

1. B/s体系结构，HTTP协议栈相关内容（HTML，CSS，JS）
2. 数据库访问的不同方法（Hibernate使用）
3. Spring IOC在企业开发中的应用
4. Restful API/SOAP开发及应用
5. SVN/GIT/MAVEN的应用

第一讲：Java SE回顾

我们可以通过”语言“来控制计算机，让计算机为我们做事情，这样的语言就叫做编程语言（Programming Language）。

一般来说，编程语言有固定的格式和词汇，我们必须经过学习才会使用，才能控制计算机。

编程语言有很多种，常用的有C语言、C++、Java、C#、PHP、JavaScript等，每种语言都有自己擅长的方面，例如：

C语言和C++主要用于PC软件开发、底层开发、单片机和嵌入式系统；

Java和C#不但可以用来开发软件，还可以用来开发网站后台程序；

PHP主要用来开发网站后台程序；

JavaScript 主要负责网站的前端工作（现在也有公司使用 Node.js 开发网站后台）

程序语言发展历程

❖机器语言

```
10001001 11100101
```

二进制机器代码，不便理解，不好记忆，与硬件平台相关，不具有可移植性。

❖汇编语言

```
mov dptr,#2000
```

用助记符号来描述，与机器代码一一对应，能够理解，但同样不可移植。

❖高级语言

按自然语言的语法风格书写程序，方便理解，在原代码的层次上可以实现跨平台移植。

```
if(a>b)
    max=a;
else
    max=b;
```

C语言是在B语言的基础上发展起来的，它的根源可以追溯到ALGOL 60。 1960年出现的ALGOL 60是一种面向问题的高级语言，它离硬件比较远，不宜用来编写系统程序。

1963年英国的剑桥大学推出了CPL（CombinedProgram- ming Language）语言。CPL语言在ALGOL 60的基础上接近了硬件一些，但规 模比较大，难以实现。1967年英国剑桥大学的Matin Richards对 CPL语言作了 简化，推出了BCPL（Basic Combined Programming Language）语言。

1970年 美国贝尔实验室的 Ken Thompson以 BCPL语言为基础，又作了进一步简化， 设计出了很简单的而且很接近硬件的 B语言（ 取 BCPL的第一个字母），并用 B语言写第一个UNIX操作系统，在PDP-7上实现。 1971年在PDP-11/20上实 现了B语言，并写了UNIX操作系统。但B语言过于简单，功能有限。

1972年至 1973年间，贝尔实验室的 D.M.Ritchie在B语言的基础上设计出了C语言（取 BCPL的第二个字母）。C语言既保持了BCPL和B语言的优点（精练、接近硬 件），又克服了它们的缺点（过于简单、数据无类型等）。 最初的C语言只是 为描述和实现UNIX操作系统提供一种工作语言而设计的。

1973年，K.Thom- pson和D.M.ritchie两人合作把UNIX的90%以上用 C改写（UNIX第5版。原来的 UNIX操作系统是1969年由美国的贝尔实验室的 K.Thompson和D.M.Ritchie开发成功的，是用汇编语言写的）。

以1978年发表的UNIX第7版中的C编译程序为基础，Brian W.Kernighan和 Dennis M.Ritchie(合称K&R)合著了影响深远了名著《The C Programming Lan- guage》，这本书中介绍的C语言成为后来广泛使用的C语言版本的基础，它被称为标准C。

1983年，美国国家标准化协会（ANSI）根据C语言问世以来各种 版本对C的发展和扩充，制定了新的标准，称为ANSI C。ANSI C比原来的标 准C有了很大的发展。K&R在1988年修改了他们的经典著作《The C Progra- mming Language》，按照ANSI C的标准重新写了该书。

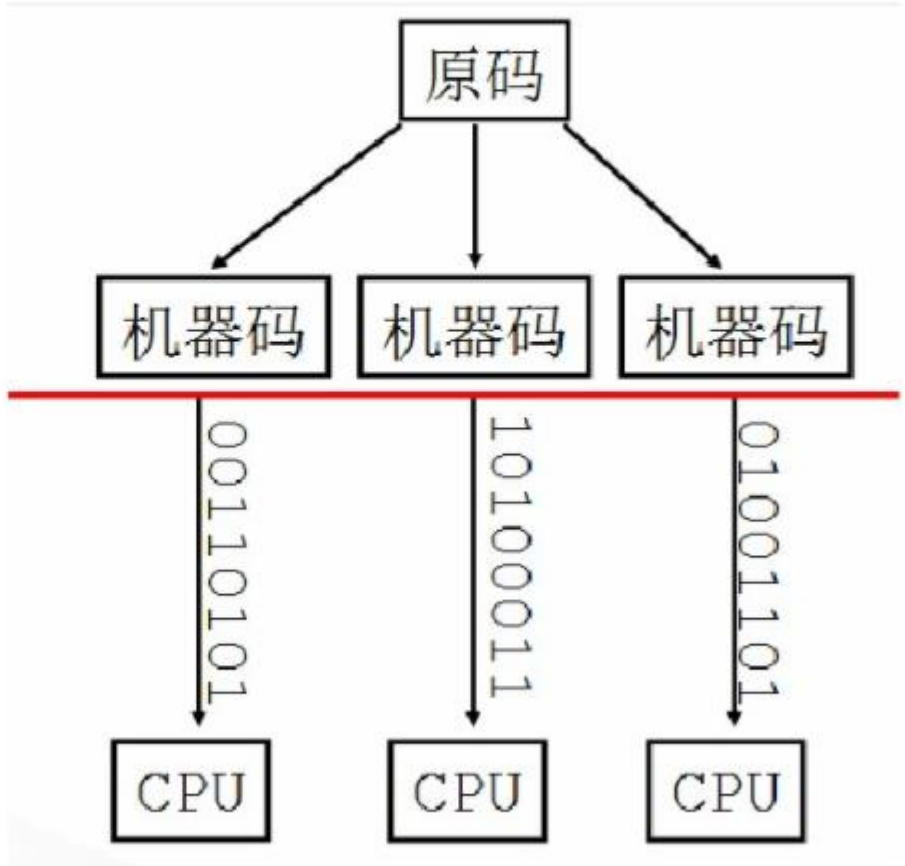
1987年，ANSI C又 公布了新标准--87 ANSI C 。目前流行的C编译系统都是以它为基础的。

高级语言开发过程

编辑原代码

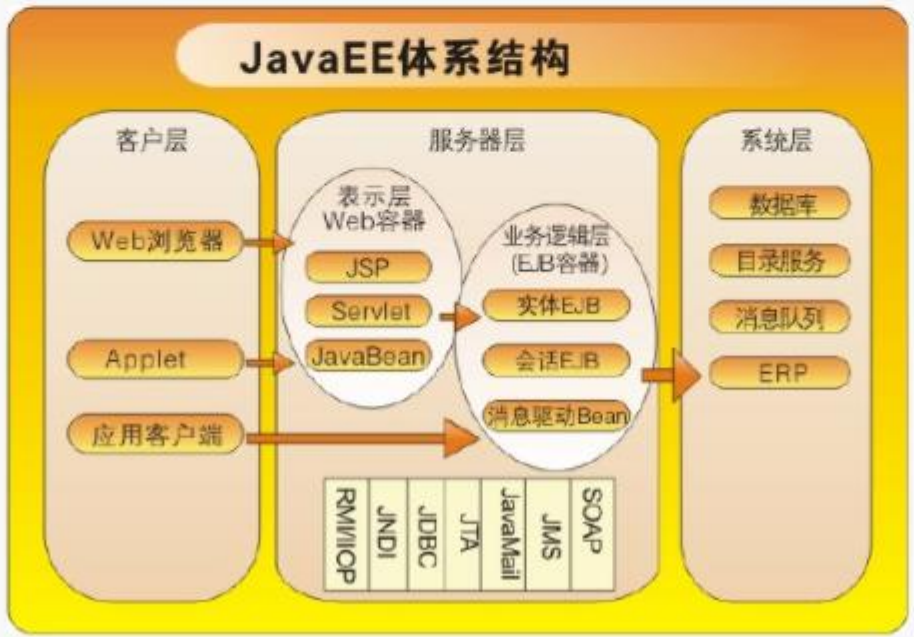
编译

执行



Java EE

Java2 EE是Java2的企业版，主要用于开发企业级分布式的网络程序，如电子商务网站和ERP（企业资源规划）系统，其核心为EJB（企业Java组件模型）。



1、.JVM -- Java virtual machine

JVM就是我们常说的Java虚拟机，它是整个java实现跨平台的最核心的部分，所有的java程序会首先被编译为.class的类文件，这种类文件可以在虚拟机上执行，也就是说class并不直接与机器的操作系统相对应，而是经过虚拟机间接与操作系统交互，由虚拟机将程序解释给本地系统执行。

2.、JRE -- java runtime environment

JRE是指java运行环境。光有JVM还不能成class的执行，因为在解释class的时候JVM需要调用解释所需要的类库lib。在JDK的安装目录里你可以找到jre目录，里面有两个文件夹bin和lib,在这里可以认为bin里的就是jvm，lib中则是jvm工作所需要的类库，而jvm和 lib和起来就称为jre。所以，在你写完java程序编译成.class之后，你可以把这个.class文件和jre一起打包发给朋友，这样你的朋友就可以运行你写程序了。（jre里有运行.class的java.exe）

3、.JDK -- java development kit

JDK是java开发工具包，基本上每个学java的人都会先在机器上装一个JDK，那他都包含哪几部分呢？让我们看一下JDK的安装目录。在目录下面有六个文件夹、一个src类库源码压缩包、和其他几个声明文件。其中，真正在运行java时起作用的是以下四个文件夹：bin、include、lib、 jre。现在我们可以看出这样一个关系，JDK包含JRE，而JRE包含JVM。

在所有字符集中，最知名的可能要数被称为ASCII的7位字符集了。它是美国标准信息交换代码（American Standard Code for Information Interchange）的缩写, 为美国英语通信所设计。它由128个字符组成，包括大小写字母、数字0-9、标点符号、非打印字符（换行符、制表符等4个）以及控制字符（退格、响铃等）组成。

但是，由于他是针对英语设计的，当处理带有音调标号（形如汉语的拼音）的亚洲文字时就会出现问题。因此，创建出了一些包括255个字符的由ASCII扩展的字符集。其中有一种通常被称为IBM字符集，它把值为128-255之间的字符用于画图和画线，以及一些特殊的欧洲字符。另一种8位字符集是ISO 8859-1Latin 1，也简称为ISOLatin-1。它把位于128-255之间的字符用于拉丁字母表中特殊语言字符的编码，也因此而得名。欧洲语言不是地球上的唯一语言，因此亚洲和非洲语言并不能被8位字符集所支持。仅汉语字母表（或pictograms）就有80000以上个字符。但是把汉语、日语和越南语的一些相似的字符结合起来，在不同的语言里，使不同的字符代表不同的字，这样只用2个字节就可以编码地球上几乎所有地区的文字。因此，创建了UNICODE编码。它通过增加一个高字节对ISO Latin-1字符集进行扩展，当这些高字节位为0时，低字节就是ISO Latin-1字符。UNICODE支持欧洲、非洲、中东、亚洲（包括统一标准的东亚象形汉字和韩国表音文字）。但是，UNICODE并没有提供对诸如Braille,Cherokee, Ethiopic, Khmer, Mongolian, Hmong, Tai Lu, Tai Mau文字的支持。同时它也不支持如Ahom, Akkadian, Aramaic, BabylonianCuneiform, Balti, Brahmi, Etruscan, Hittite, Javanese, Numidian, Old Persian Cuneiform, Syrian之类的古老文字。

1. ASCII码

我们知道，在计算机内部，所有的信息最终都表示为一个二进制的字符串。每一个二进制位（bit）有0和1两种状态，因此八个二进制位就可以组合出256种状态，这被称为一个字节（byte）。也就是说，一个字节一共可以用来表示256种不同的状态，每一个状态对应一个符号，就是256个符号，从0000000到11111111。

上个世纪60年代，美国制定了一套字符编码，对英语字符与二进制位之间的关系，做了统一规定。这被称为ASCII码，一直沿用至今。

ASCII码一共规定了128个字符的编码，比如空格"SPACE"是32（二进制00100000），大写的字母A是65（二进制01000001）。这128个符号（包括32个不能打印出来的控制符号），只占用了一个字节的后面7位，最前面的1位统一规定为0。

2、非ASCII编码

英语用128个符号编码就够了，但是用来表示其他语言，128个符号是不够的。比如，在法语中，字母上方有注音符号，它就无法用ASCII码表示。于是，一些欧洲国家就决定，利用字节中闲置的最高位编入新的符号。比如，法语中的é的编码为130（二进制10000010）。这样一来，这些欧洲国家使用的编码体系，可以表示最多256个符号。

但是，这里又出现了新的问题。不同的国家有不同的字母，因此，哪怕它们都使用256个符号的编码方式，代表的字母却不一样。比如，130在法语编码中代表了é，在希伯来语编码中却代表了字母Gimel (ג)，在俄语编码中又会代表另一个符号。但是不管怎样，所有这些编码方式中，0--127表示的符号是一样的，不一样的只是128--255的这一段。

至于亚洲国家的文字，使用的符号就更多了，汉字就多达10万左右。一个字节只能表示256种符号，肯定是不够的，就必须使用多个字节表达一个符号。比如，简体中文常见的编码方式是GB2312，使用两个字节表示一个汉字，所以理论上最多可以表示256x256=65536个符号。

3.Unicode

正如上一节所说，世界上存在着多种编码方式，同一个二进制数字可以被解释成不同的符号。因此，要想打开一个文本文件，就必须知道它的编码方式，否则用错误的编码方式解读，就会出现乱码。为什么电子邮件常常出现乱码？就是因为发信人和收信人使用的编码方式不一样。

可以想象，如果有一种编码，将世界上所有的符号都纳入其中。每一个符号都给予一个独一无二的编码，那么乱码问题就会消失。这就是Unicode，就像它的名字都表示的，这是一种所有符号的编码。

Unicode当然是一个很大的集合，现在的规模可以容纳100多万个符号。每个符号的编码都不一样，比如，U+0639表示阿拉伯字母Ain，U+0041表示英语的大写字母A，U+4E25表示汉字"严"。具体的符号对应表，可以查询unicode.org

UTF-8

互联网的普及，强烈要求出现一种统一的编码方式。UTF-8就是在互联网上使用最广的一种Unicode的实现方式。其他实现方式还包括UTF-16（字符用两个字节或四个字节表示）和UTF-32（字符用四个字节表示），不过在互联网上基本不用。重复一遍，这里的关系是，UTF-8是Unicode的实现方式之一。

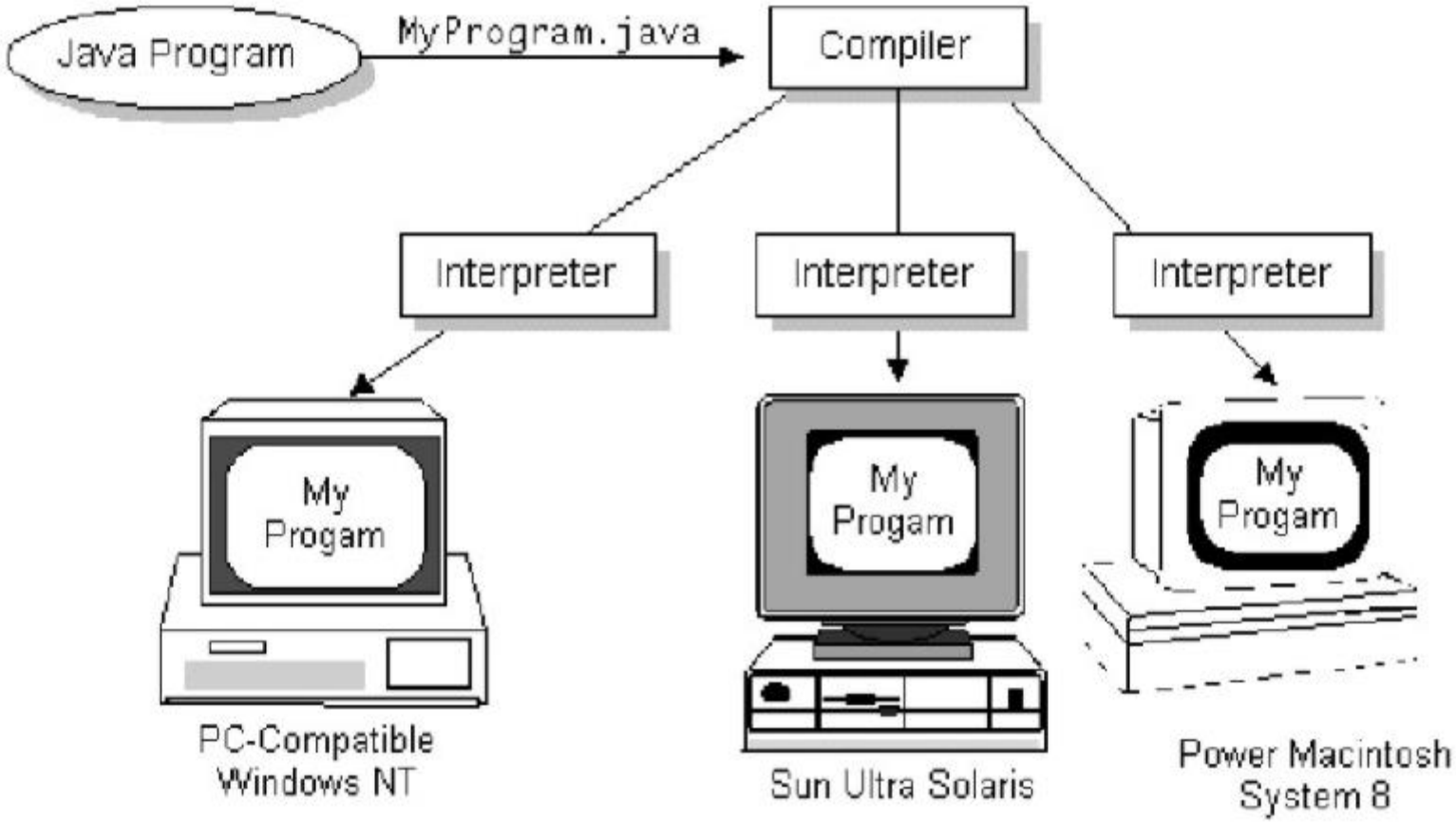
UTF-8最大的一个特点，就是它是一种变长的编码方式。它可以使用1~4个字节表示一个符号，根据不同的符号而变化字节长度。

UTF-8的编码规则很简单，只有二条：

1）对于单字节的符号，字节的第一位设为0，后面7位为这个符号的unicode码。因此对于英语字母，UTF-8编码和ASCII码是相同的。

2）对于n字节的符号（n>1），第一个字节的前n位都设为1，第n+1位设为0，后面字节的前两位一律设为10。剩下的没有提及的二进制位，全部为这个符号的unicode码。

JVM的作用


Java Program
MyProgram.java
Compiler
Interpreter
Interpreter
Interpreter
My Progam
My Progam
My Progam
PC-Compatible
Windows NT
Sun Ultra Solaris
Power Macintosh
System 8

垃圾收集器做什么？

- 释放非存活对象占据的内存空间
- 管理内存，决定了内存分配机制

垃圾算法的基本要求

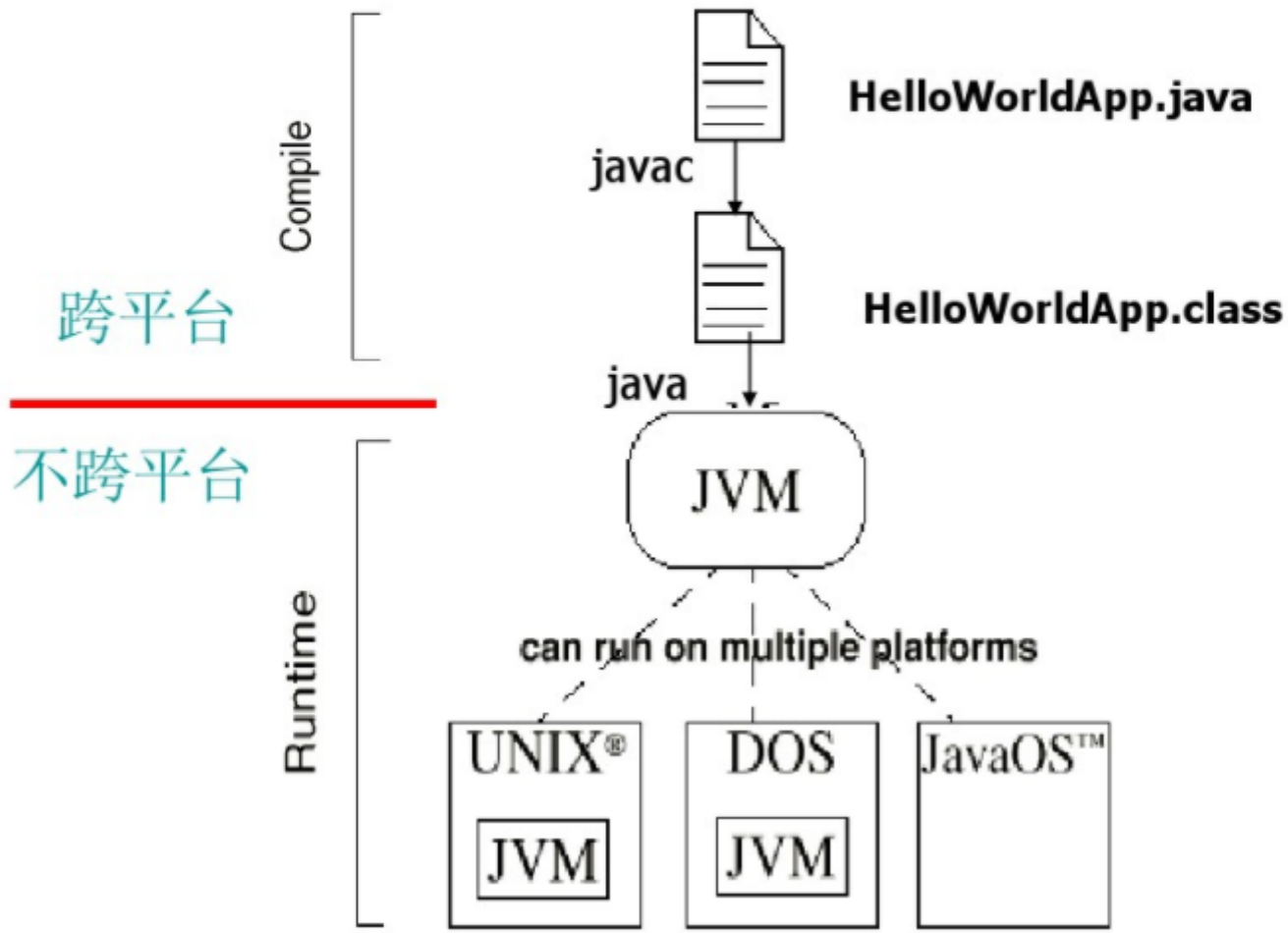
- 必须是安全的，存活数据不能被错误回收
- 应该是全面的，垃圾对象会在固定的收集周期被回收
- 应该有合理的开销，时间/空间/运行频率
- 尽可能少的内存碎片
- 应该是可扩展的，不会成为可扩展瓶颈

Java垃圾收集的概念

- Java语言使用new运算符来分配内存空间，没有动态内存分配的概念。
- Java系统线程自动处理无用内存空间的垃圾收集
- Java程序员只能建议，但不能强制JVM去执行垃圾收集程序，使用下面的代码：

```
java.lang.System.gc()
java.lang.Runtime.gc()
```

Java程序的运行过程



javac，即javac.exe，是JDK中自带的一个Java源代码编译工具。Eclipse等开发工具实际上也是调用javac来编译Java源代码的。javac.exe一般位于JDK安装目录/bin文件夹中，想要在命令行窗口中直接使用javac命令，我们需要将该路径追加到系统环境变量path中

-classpath <路径>（-cp缩写）：指定要使用的类路径或要使用的jar包的路径(jar文件、zip文件（里面都是错啦搜索文件）)，使用后覆盖CLASSPATH的设定

C:\work-train\workspaces\Assign1\src>javac C:\work-train\workspaces\Assign1\src\lab5\Lottery.java

C:\work-train\workspaces\Assign1\src>javac -cp . C:\work-train\workspaces\Assign1\src\lab5\LotteryDemo.java

C:\>java -classpath C:\work-train\workspaces\Assign1\src\ lab5.LotteryDemo
lab5.Lottery@459bdb65

Java 包(package)

为了更好地组织类，Java 提供了包机制，用于区别类名的命名空间。

包的作用

1、把功能相似或相关的类或接口组织在同一个包中，方便类的查找和使用。

2、如同文件夹一样，包也采用了树形目录的存储方式。同一个包中的类名字是不同的，不同的包中的类的名字是可以相同的，当同时调用两个不同包中相同类名的类时，应该加上包名加以区别。因此，包可以避免名字冲突。

3、包也限定了访问权限，拥有包访问权限的类才能访问某个包中的类。

package pkg1[. pkg2[. pkg3…]];

例如,一个Something.java 文件它的内容

```
package net.java.util

public class Something{

  ...
```

那么它的路径应该是 net/java/util/Something.java 这样保存的。 package(包) 的作用是把不同的 java 程序分类保存，更方便的被其他 java 程序调用。

创建包

创建包的时候，你需要为这个包取一个合适的名字。之后，如果其他的一个源文件包含了这个包提供的类、接口、枚举或者注释类型的时候，都必须将这个包的声明放在这个源文件的开头。

包声明应该在源文件的第一行，每个源文件只能有一个包声明，这个文件中的每个类型都应用于它。

如果一个源文件中没有使用包声明，那么其中的类，函数，枚举，注释等将被放在一个无名的包（unnamed package）中。

例子

让我们来看一个例子，这个例子创建了一个叫做animals的包。通常使用小写的字母来命名避免与类、接口名字的冲突。

在 animals 包中加入一个接口（interface）：

Animal.java 文件代码：

```
/* 文件名: Animal.java */
package animals;


interface Animal {
  public void eat();
  public void travel();
}
```

import 关键字

为了能够使用某一个包的成员，我们需要在 Java 程序中明确导入该包。使用 "import" 语句可完成此功能。

在 java 源文件中 import 语句应位于 package 语句之后，所有类的定义之前，可以没有，也可以有多条，其语法格式为：

import package1[.package2…].(classname|*);

如果在一个包中，一个类想要使用本包中的另一个类，那么该包名可以省略。

通常，一个公司使用它互联网域名的颠倒形式来作为它的包名.例如：互联网域名是 runoob.com，所有的包名都以 com.runoob 开头。包名中的每一个部分对应一个子目录。

例如：有一个 com.runoob.test 的包，这个包包含一个叫做 Sites.java 的源文件，那么相应的，应该有如下面的一连串子目录：

....\com\runoob\test\Sites.java

Windows下JAVA用到的环境变量主要有3个，JAVA_HOME、CLASSPATH、PATH。下面逐个分析。

JAVA_HOME 指向的是JDK的安装路径，如C:\jdk1.5.0_06，在这路径下你应该能够找到bin、lib等目录。值得一提的是，JDK的安装路径可以选择任意磁盘目录，

PATH 环境变量原来Windows里面就有，你只需修改一下，使他指向JDK的bin目录，这样你在控制台下面编译、执行程序时就不需要再键入一大串路径了。设置方法是保留原来的PATH的内容，并在其中加上%JAVA_HOME%\bin

CLASSPATH是什么？它的作用是什么？

它是javac编译器的一个环境变量。

它的作用与import、package关键字有关。

当你写下improt java.util.*时，编译器面对import关键字时，就知道你要引入java.util这个package中的类；但是编译器如何知道你把这个package放在哪里了呢？所以你首先得告诉编译器这个package的所在位置；如何告诉它呢？就是设置CLASSPATH， 如果java.util这个package在c:\jdk\ 目录下，你得把c:\jdk\这个路径设置到CLASSPATH中去！当编译器面对import java.util.*这个语句时，它先会查找CLASSPATH所指定的目录，并检视子目录java\util是否存在，然后找出名称吻合的已编译文件（.class文件）。如果没有找到就会报错！

Java归档文件格式(Java Archive, JAR)能够将多个源码、资源等文件打包到一个归档文件中。这样，有如下好处：

安全性

可以对整个jar包的内容进行签名。

减少了下载时间

如果applet被打包成一个jar文件，那么所有相关的资源就可以在一个HTTP transaction中下载完成，而无需为每一个文件新建一个连接。

压缩

减少了磁盘空间的占用。

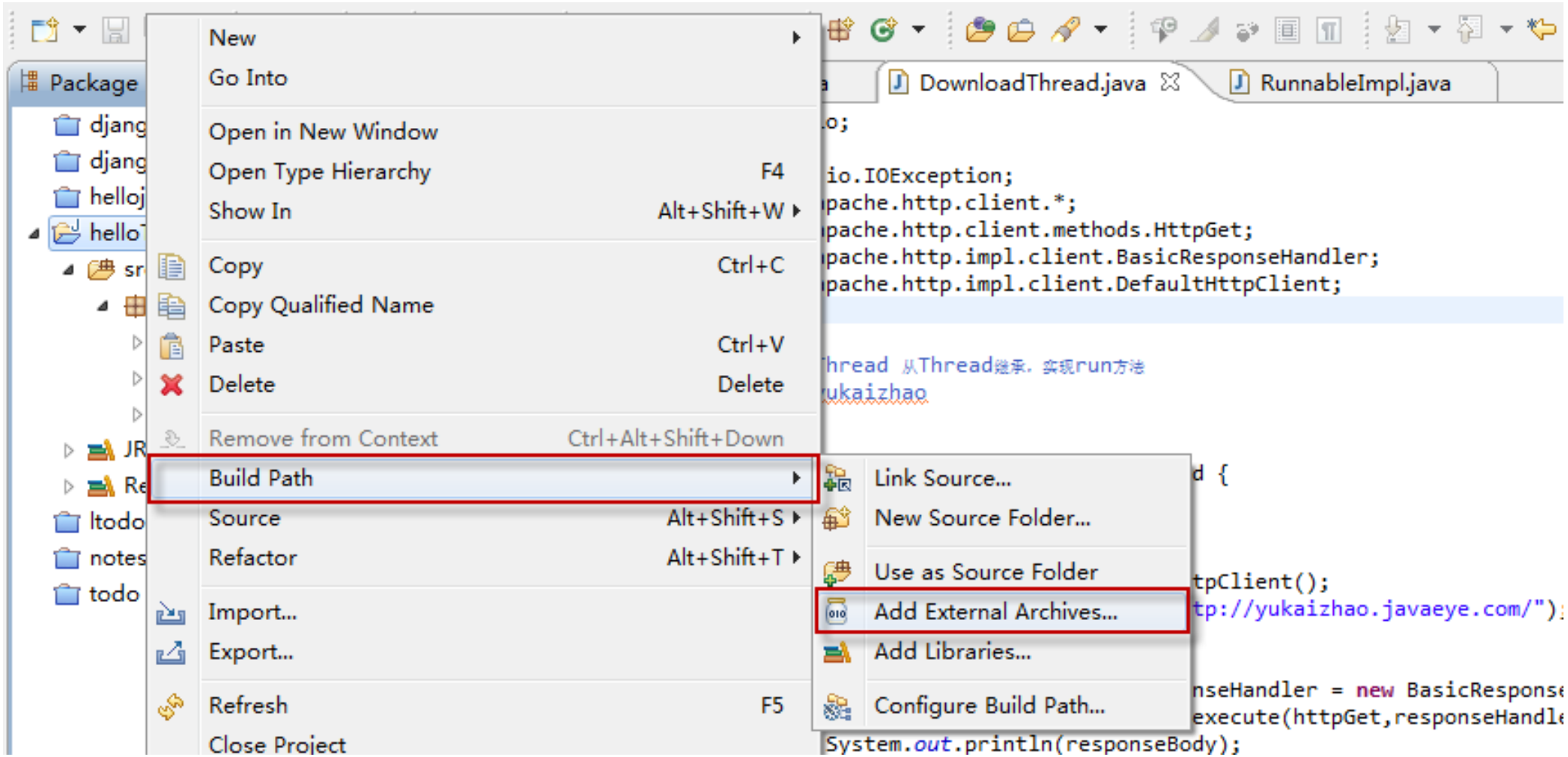
容易扩展

通过jar这种格式，可以和容易地将自己的程序打包提供给别人使用。

2、jar的使用

JDK自带的打包工具通过jar命令来调用，jar是通过zip格式进行打包的。所以，这个jar工具也可以作为日常的压缩、解压缩工具来进行使用。

如果安装了JDK并配置了环境变量，在命令行中输入jar命令，不加任何参数，就可以看到jar命令的使用说明：


Package
DownloadThread.java
RunnableImpl.java
New
Go Into
Open in New Window
Open Type Hierarchy F4
Show In Alt+Shift+W
Copy Ctrl+C
Copy Qualified Name
Paste Ctrl+V
Delete Delete
Remove from Context Ctrl+Alt+Shift+Down
Build Path
Source Alt+Shift+S
Refactor Alt+Shift+T
Import...
Export...
Refresh F5
Close Project
Link Source...
New Source Folder...
Use as Source Folder
Add External Archives...
Add Libraries...
Configure Build Path...
io.IOException;
pache.http.client.*;
pache.http.client.methods.HttpGet;
pache.http.impl.client.BasicResponseHandler;
pache.http.impl.client.DefaultHttpClient;
hread 从Thread继承，实现run方法
ukaizhao
tpClient();
tp://yukaizhao.javaeye.com/");
nseHandler = new BasicResponse
execute(httpGet,responseHandle
System.out.println(responseBody);

Apache软件基金会（Apache Software Foundation，简称为ASF），是专门为支持开源软件项目而办的一个非营利性组织。在它所支持的Apache项目与子项目中，所发行的软件产品都遵循Apache许可证（Apache License）

Apache软件基金会正式创建于1999年，它的创建者是一个自称为“Apache组织”的群体。这个“Apache组织”在1999年以前就已经存在很长时间了，这个组织的开发爱好者们聚集在一起，在美国伊利诺伊大学超级计算机应用程序国家中心（National Center for Supercomputing Applications，简称为NCSA）开发的NCSA HTTPd服务器的基础上开发与维护了一个叫Apache的HTTP服务器。

最初NCSA HTTPd服务器是由Rob McCool开发出来的，但是它的最初开发者们逐渐对这个软件失去了兴趣，并转移到了其他地方，造成了没有人来对这个服务器软件提供更多的技术支持。因为这个服务器的功能又如此强大，而代码可以自由下载修改与发布，当时这个服务器软件的一些爱好者与用户开始自发起来，互相交流并分发自己修正后的软件版本，并不断改善其功能。为了更好进行沟通，Brian Behlendorf自己创建了一个邮件列表，把它作为这个群体（或者社区）交流技术、维护软件的一个媒介，把代码重写与维护的工作有效组织起来。这些开发者们逐渐地把他们这个群体称为“Apache组织”，把这个经过不断修正并改善的服务器软件命名为Apache服务器（Apache Server）

https://commons.apache.org/proper/commons-lang/index.html

StringUtils 方法的操作对象是 java.lang.String 类型的对象，是对 JDK 提供的 String 类型操作方法的补充，并且是 null 安全的(即如果输入参数 String 为 null 则不会抛出 NullPointerException ，而是做了相应处理，例如，如果输入为 null 则返回也是 null 等，具体可以查看源代码)。

除了构造器，StringUtils 中一共有130多个方法，并且都是 static 的，所以我们可以这样调用 StringUtils.xxx()

1. public static boolean isEmpty(String str)

判断某字符串是否为空，为空的标准是 str==null 或 str.length()==0

下面是 StringUtils 判断是否为空的示例：

StringUtils.isEmpty(null) = true

StringUtils.isEmpty("") = true

StringUtils.isEmpty(" ") = false //注意在 StringUtils 中空格作非空处理

StringUtils.isEmpty(" ") = false

StringUtils.isEmpty("bob") = false

StringUtils.isEmpty(" bob ") = false

public static boolean isBlank(String str)

判断某字符串是否为空或长度为0或由空白符(whitespace) 构成

下面是示例：

StringUtils.isBlank(null) = true

StringUtils.isBlank("") = true

StringUtils.isBlank(" ") = true

StringUtils.isBlank("　　") = true

StringUtils.isBlank("\t \n \f \r") = true　//对于制表符、换行符、换页符和回车符

StringUtils.isBlank()　//均识为空白符

StringUtils.isBlank("\b") = false　//"\b"为单词边界符

StringUtils.isBlank("bob") = false

StringUtils.isBlank(" bob ") = false

IO文件操作，可以说是除了JDBC操作之外，日常最常用的功能之一了。IO的读写方式如何，直接影响到系统的性能。很多时候系统的性能瓶颈往往不是出现在对象层面，而是出现在底层的IO层面上。

Apache commosn IO包在input, output包的基础上，提供了一个高效，方便的文件类处理工具：FileUtils，其功能涵盖了所有日常常用的IO操作，由于这个类的部分方法底层是基于Apache commons IO自己的读写流去实现的，所以在性能上会相对高于JDK自带的类)

根据Apache commons IO官方的说法，这个类可以提供如下功能：

Facilities are provided in the following areas:

- writing to a file
- reading from a file
- make a directory including parent directories
- copying files and directories
- deleting files and directories
- converting to and from a URL
- listing files and directories by filter and extension
- comparing file content
- file last changed date
- calculating a checksum

JCP（Java Community Process 标准制定组织）

JCP（Java Community Process) 是一个开放的国际组织，主要由Java开发者以及被授权者组成，职能是发展和更新Java技术规范、参考实现（RI）、技术兼容包（TCK）。Java技术和JCP两者的原创者都是SUN计算机公司。然而，JCP已经由SUN于1995年创造Java的非正式过程，演进到如今有数百名来自世界各地Java代表成员一同监督Java发展的正式程序。

JCP维护的规范包括J2ME、J2SE、J2EE，XML，OSS，JAIN等。组织成员可以提交JSR（Java Specification Requests），通过特定程序以后，进入到下一版本的规范里面。

Java Specification Requests，Java规范请求，由JCP成员向委员会提交的Java发展议案，经过一系列流程后，如果通过最终会体现在未来的Java中

所有声称符合J2EE规范的J2EE类产品（应用服务器、应用软件、开发工具等），必须通过该组织提供的TCK兼容性测试（需要购买测试包），通过该测试后，需要缴纳J2EE商标使用费。两项完成，即是通过J2EE认证（Authorized Java Licensees of J2EE）。

https://www.jcp.org/en/home/index

练习使用 FileUtils和StringIUtils

64.242.88.10 - - [07/Mar/2004:17:31:39 -0800] "GET /twiki/bin/edit/Main/UvscanAndPostFix?topicparent=Main.WebHome HTTP/1.1" 401 12846

64.242.88.10 - - [07/Mar/2004:17:35:35 -0800] "GET /twiki/bin/view/TWiki/KlausWriessnegger HTTP/1.1" 200 3848

64.242.88.10 - - [07/Mar/2004:17:39:39 -0800] "GET /twiki/bin/view/Main/SpamAssassin HTTP/1.1" 200 4081

http://thechronicleherald.ca/heraldflyers

access_log

数组是大小固定的，并且同一个数组只能存放类型一样的数据（基本类型/引用类型），JAVA集合可以存储和操作数目不固定的一组数据。 所有的JAVA集合都位于 java.util包中.

在实际的项目开发中会有很多的对象，如何高效、方便地管理对象，成为影响程序性能与可维护性的重要环节。Java 提供了集合框架来解决此类问题，线性表、链表、哈希表等是常用的数据结构，在进行 Java 开发时，JDK 已经为我们提供了一系列相应的类来实现基本的数据结构，所有类都在 java.util 这个包里

JAVA集合主要分为三种类型：

Set(集)

Set是最简单的一种集合。集合中的对象不按特定的方式排序，并且没有重复对象. HashSet类按照哈希算法来存取集合中的对象，存取速度比较快.

中的值不允许重复，无序的数据结构

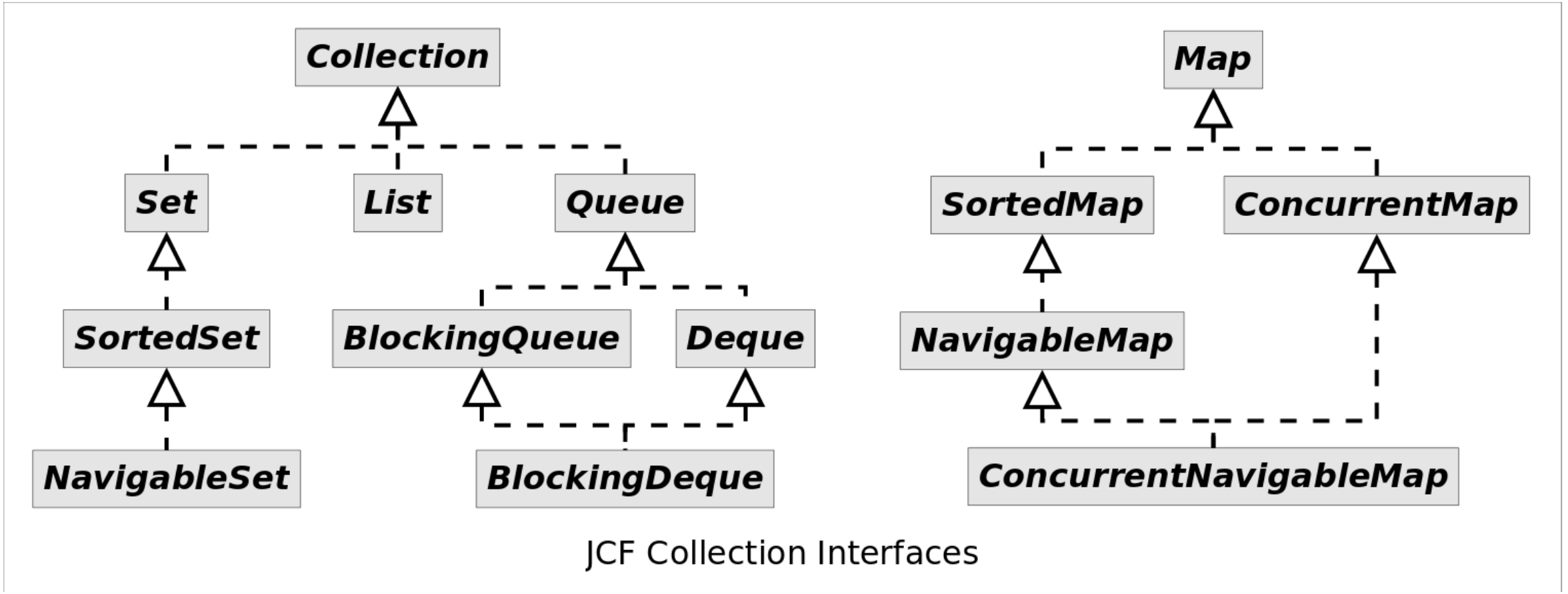
List(列表)

List的特征是其元素以线性方式存储，集合中可以存放重复对象。 ArrayList, 其中的值允许重复，因为其为有序的数据结构

Map(映射)

Map 是一种把键对象和值对象映射的集合，它的每一个元素都包含一对键对象和值对象。

为了规范容器的行为，统一设计，JCF定义了14种容器接口（collection interfaces），它们的关系如下图所示：



JCF Collection Interfaces

如何遍历Collection中的每一个元素？不论Collection的实际类型如何，它都支持一个iterator()的方法，该方法返回一个迭代器，使用该迭代器即可遍历访问Collection中每一个元素，通过Iterator的遍历是无序的。

迭代器（Iterator）为我们提供了遍历容器中元素的方法。只有容器本身清楚容器里元素的组织方式，因此迭代器只能通过容器本身得到。每个容器都会通过内部类的形式实现自己的迭代器。

```
// visit a list with iterator
ArrayList list = new ArrayList();
list.add(new String("Monday"));
list.add(new String("Tuesday"));
list.add(new String("Wensday"));
Iterator<String> it = list.iterator();
// 得到迭代器
while (it.hasNext()) {
        String weekday = (String)it.next();
        // 访问元素
        System.out.println(weekday.toUpperCase());
}
```

ArrayList 是一个数组队列，相当于 动态数组。与Java中的数组相比，它的容量能动态增长。

每个 ArrayList 实例都有一个容量，该容量是指用来存储列表元素的数组的大小。默认初始容量为 10。随着 ArrayList 中元素的增加，它的容量也会不断的自动增长。在每次添加新的元素时，ArrayList 都会检查是否需要进行扩容操作

ArrayList()：默认构造函数，提供初始容量为 10 的空列表。

add()是添加一个新的元素,remove()删除一个元素,size()获得ArrayList的长度。ArrayList的下标是从0开始。

```
rrayList al=new ArrayList();
//向Java动态数组中添加数据
al.add("a");
al.add("b");
al.add("c");
//输出Java动态数组
for(int i=0;i<al.size();i++)
{
String alEach=(String)al.get(i);
System.out.println(alEach);
}
//删除数组中的某个元素,删除第二个元素
al.remove(1);
//修改Java动态数组，把新的元素放到第二个位置
al.add(1,"2");
////输出Java动态数组
for(int i=0;i<al.size();i++)
{
String alEach=(String)al.get(i);
System.out.println(alEach);
```

HashSet 简介

HashSet 是一个没有重复元素的集合,不保证元素的顺序，而且HashSet允许使用 null 元素。

HashSet的主要API

```
boolean        add(E object)
void           clear()
Object         clone()
boolean        contains(Object object)
boolean        isEmpty()
Iterator<E>    iterator()
boolean        remove(Object object)
int            size()
```

HashMap 是一个散列表，它存储的内容是键值对(key-value)映射。

containsKey() 的作用是**判断HashMap是否包含key。**

put() 的作用是**对外提供接口，让HashMap对象可以通过put()将“key-value”添加到HashMap中**。

get() 的作用是**获取key对应的value**，它的实现代码如下：

```
// 假设map是HashMap对象
// map中的key是String类型，value是Integer类型
Integer integ = null;
Iterator iter = map.entrySet().iterator();
while(iter.hasNext()) {
    Map.Entry entry = (Map.Entry)iter.next();
    // 获取key
    key = (String)entry.getKey();
        // 获取value
    integ = (Integer)entry.getValue();
}
```

List转Set

```
Set set = new HashSet(new ArrayList());
```

Set转List

```
List list = new ArrayList(new HashSet());
```

数组转为List

```
List arr = Arrays.asList("1", "2", "3");
//或者
String[] arr = {"1", "2"};
List list = Arrays.asList(arr);
```

数组转为Set

```
int[] arr = { 1, 2, 3 };
Set set = new HashSet(Arrays.asList(arr));
```

Map的值转化为List

```
List list = new ArrayList(map.values());
```

Map的值转化为Set

```
Set set = new HashSet(map.values());
```

List转数组

```
List list = Arrays.asList("a","b");
String[] arr = (String[])list.toArray(new String[list.size()]);
```

Hashmap 基于value的排序

Java对象的比较

一、简单类型比较

Java中，比较简单类型变量用“==”，只要两个简单类型值相等即返回ture，否则返回false；

二、引用类型比较

引用类型比较比较变态，可以用“==”，也可以用“equals()”来比较，equals()方法来自于Object类，每个自定义的类都可以重写这个方法。Object类中的equals()方法仅仅通过“==”来比较两个对象是否相等。

在用“==”比较引用类型时，仅当两个应用变量的对象指向同一个对象时，才返回ture。言外之意就是要求两个变量所指内存地址相等的时候，才能返回true，每个对象都有自己的一块内存，因此必须指向同一个对象才返回ture。

在Java API中，有些类重写了equals()方法，它们的比较规则是：当且仅当该equals方法参数不是 null，两个变量的类型、内容都相同，则比较结果为true。这些类包括：String、Double、Float、Long、Integer、Short、Byte、、Boolean、BigDecimal、BigInteger等等，具体可以查看API中类的equals()方法，就知道了。

三、重写equals()方法

在定义一个类的时候，如果涉及到对象的比较，应该重写equals()方法。重写的一般规则是：

1、先用“==”判断是否相等。

2、判断equals()方法的参数是否为null，如果为null，则返回false；因为当前对象不可能为null，如果为null，则不能调用其equals()方法，否则抛java.lang.NullPointerException异常。

3、当参数不为null，则如果两个对象的运行时类（通过getClass()获取）不相等，返回false，否则继续判断。

4、判断类的成员是否对应相等。

课后作业：

1。继续练习、领会如何在工程中引入第三方的jar文件，到如下网站下载jar文件，新建一个java工程，添加jar文件。

https://commons.apache.org/proper/commons-lang/download_lang.cgi

http://commons.apache.org/proper/commons-io/download_io.cgi

2。自己在google中搜索StringUtils和FileUtils两个常用工具类的用法

3。分析access_log （我发给你的文件），找出访问最多的地址

思路： 1。用fileUtils把文件读取到一个string的变量中，尝试用两种方法做

第一种方法： 用 string类的自有函数 indexOf substring等，可以借助 array

第二种方法： 用StringUtils的静态函数，感受一下 StringUtils为什么方便

4。看一下 http://thechronicleherald.ca/heraldflyers

这个地址，想一下如果想把每个flyer的图标都下载到本地，如何实现？

List是一个接口，而ArrayList是一个类,ArrayList继承并实现了List。
所以List不能被构造，但可以为List创建一个引用，相对而言，ArrayList就可以被构造。
List list; //正确 list=null;
List list=new List(); // 是错误的用法

List list = new ArrayList();这句创建了一个ArrayList的对象后把上溯到了List。此时它是一个List对象了，有些ArrayList有但是List没有的属性和方法，它就不能再用了。
而ArrayList list=new ArrayList();创建一对象则保留了ArrayList的所有属性。
这是一个例子：

```
import java.util.*;

public class TestList{
public static void main(String[] args){
List list = new ArrayList();
ArrayList arrayList = new ArrayList();
list.trimToSize(); //错误，没有该方法。
arrayList.trimToSize();  //ArrayList里有该方法。
}
}
```

（1） 一个类（如：ClassA），如果没有声明任何构造函数，那么系统会自动生成一个无参构造函数，此时使用ClassA myClass= new ClassA（），不会报错。但是，如果显式声明了一个有参构造函数，却没有声明一个无参构造函数的话，系统不会自动生成一个无参构造函数，此时，再使用ClassA myClass = new ClassA（）就会报错。如果要消除报错，则必须使用有参构造函数，

或者添加一个无参构造函数。所以，一个类的构造函数，一般只有三种状况：无参 或者 无参 + 有参 或者 有参

（2） 在继承关系中，子类的所有构造函数（包括无参构造函数（默认构造函数），有参构造函数等），如果不显式声明调用哪种super，那么都会默认调用super（），即都会默认调用父类的无参构造函数（默认构造函数）。而，如果父类此时没有无参构造函数存在的话，就会报错。为了修改报错，只能显式调用父类显式声明的构造函数之一或者在父类中增加无参构造函数。

（3） 在继承关系中，当使用new一个类时，程序的执行顺序为：首先执行父类的构造函数方法，再执行自身的构造函数方法，这个执行顺序不可更改，否则报错或不能运行。

（4） 在深度继承关系中，即当存在C继承B，B继承A，或者更多层继承时，首先执行最上层的构造函数，再依次顺着继承链传递下去，一直到创建对象的那个类。例如：我们声明C对象时，调用的是C的构造函数c,构造函数c调用的是父类B的构造函数b,构造函数b调用

的是父类A中的构造函数a，那么执行结果就是：a => b => c 。

在语法层次，java 语言对于抽象类和接口分别给出了不同的定义。下面用Demo 类来说明他们之间的不同之处。

```
public abstract class Demo {
    abstract void method1();
    void method2(){
      //实现
    }
  }

 interface Demo {
    void method1();
    void method2();
  }
```

在使用抽象类时需要注意几点：

1、抽象类不能被实例化，实例化的工作应该交由它的子类来完成，它只需要有一个引用即可。

2、抽象方法必须由子类来进行重写。

3、只要包含一个抽象方法的抽象类，该方法必须要定义成抽象类，不管是否还包含有其他方法。

4、抽象类中可以包含具体的方法，当然也可以不包含抽象方法。

5、子类中的抽象方法不能与父类的抽象方法同名。

定义一个抽象动物类 Animal，提供抽象方法叫 cry()，猫、狗都是动物类的子类，由于 cry() 为抽象方法，所以 Cat、Dog 必须要实现 cry() 方法。如下：

```
public abstract class Animal {
    public abstract void cry();
}
public class Cat extends Animal{
    @Override
    public void cry() {
        System.out.println("猫叫： 喵喵...");
    }
}
public class Dog extends Animal{
    @Override
    public void cry() {
        System.out.println("狗叫:汪汪...");
    }
}
```

```
public class Test {

    public static void main(String[] args) {
        Animal a1 = new Cat();
        Animal a2 = new Dog();

        a1.cry();
        a2.cry();
    }
}
```

Output:

```
猫叫：喵喵...
狗叫:汪汪...
```

二、接口

接口是一种比抽象类更加抽象的“类”。

接口是用来建立类与类之间的协议，它所提供的只是一种形式，而没有具体的实现。同时实现该接口的实现类必须要实现该接口的所有方法，通过使用 implements 关键字，他表示该类在遵循某个或某组特定的接口，同时也表示着"interface"只是它的外貌，但是现在需要声明它是如何工作的”。

接口是抽象类的延伸，java 了保证数据安全是不能多重继承的，也就是说继承只能存在一个父类，但是接口不同，一个类可以同时实现多个接口，不管这些接口之间有没有关系，所以接口弥补了抽象类不能多重继承的缺陷，但是推荐继承和接口共同使用，因为这样既可以保证数据安全性又可以实现多重继承。

在使用接口过程中需要注意如下几个问题：

1、1个 Interface 的方所有法访问权限自动被声明为 public。确切的说只能为 public，当然你可以显示的声明为 protected、private，但是编译会出错！

2、接口中可以定义“成员变量”，或者说是不可变的常量，因为接口中的“成员变量”会自动变为为 public static final。可以通过类命名直接访问：ImplementClass.name。

3、接口中不存在实现的方法。

4、实现接口的非抽象类必须要实现该接口的所有方法。

5、不能使用 new 操作符实例化一个接口，但可以声明一个接口变量，该变量必须引用(refer to) 一个实现该接口的类的对象。可以使用 instanceof 检查一个对象是否实现了某个特定的接口。例如：if(anObject instanceof Comparable){}。

抽象类方式中，抽象类可以拥有任意范围的成员数据，同时也可以拥有自己的非抽象方法，但是接口方式中，它仅能够有静态、不能修改的成员数据（但是我们一般是不会在接口中使用成员数据），同时它所有的方法都必须是抽象的。在某种程度上来说，接口是抽象类的特殊化。

对子类而言，它只能继承一个抽象类，但是却可以实现多个接口。

1、 抽象层次不同。抽象类是对类抽象，而接口是对行为的抽象。抽象类是对整个类整体进行抽象，包括属性、行为，但是接口却是对类局部（行为）进行抽象。

2、 跨域不同。抽象类所跨域的是具有相似特点的类，而接口却可以跨域不同的类。我们知道抽象类是从子类中发现公共部分，然后泛化成抽象类，子类继承该父类即可，但是接口不同。实现它的子类可以不存在任何关系，共同之处。例如猫、狗可以抽象成一个动物类抽象类，具备叫的方法。鸟、飞机可以实现飞 Fly 接口，具备飞的行为，这里我们总不能将鸟、飞机共用一个父类吧！所以说抽象类所体现的是一种继承关系，要想使得继承关系合理，父类和派生类之间必须存在"is-a"关系，即父类和派生类在概念本质上应该是相同的。对于接口则不然，并不要求接口的实现者和接口定义在概念本质上是一致的， 仅仅是实现了接口定义的契约而已。

3、 设计层次不同。对于抽象类而言，它是自下而上来设计的，我们要先知道子类才能抽象出父类，而接口则不同，它根本就不需要知道子类的存在，只需要定义一个规则即可，至于什么子类、什么时候怎么实现它一概不知。比如我们只有一个猫类在这里，如果你这是就抽象成一个动物类，是不是设计有点儿过度？我们起码要有两个动物类，猫、狗在这里，我们在抽象他们的共同点形成动物抽象类吧！所以说抽象类往往都是通过重构而来的！但是接口就不同，比如说飞，我们根本就不知道会有什么东西来实现这个飞接口，怎么实现也不得而知，我们要做的就是事前定义好飞的行为接口。所以说抽象类是自底向上抽象而来的，接口是自顶向下设计出来的。

总结：接口和抽象类

“接口是个规范”，既然不是一个类去实现，那就是有很多地方有用到，大家需要统一标准。甚至有的编程语言（Object-C）已经不把接口叫 interface，直接叫 protocol。统一标准的目的，是大家都知道这个是做什么的，但是具体不用知道具体怎么做。

我知道 Comparable 这个接口是用来比较两个对象的，那么如何去比较呢？数字有数字的比较方法，字符串有字符串的比较方法，学生（自己定义的类）也有自己的比较方法。然后，在另外一个负责对象排序（不一定是数字喔）的代码里面，肯定需要将两个对象比较。这两个对象是什么类型呢？Object a,b？肯定不行，a > b 这样的语法无法通过编译。int a,b？也不行？一开始就说了，不一定是数字。....所以，Comparable 就来了。他告诉编译器，a b 两个对象都满足 Comparable 接口，也就是他们是可以进行比较的。具体怎么比较，这段程序不需要知道。所以，他需要一些具体的实现，Comparable 接口有一个方法，叫 compareTo。那么这个方法就是用来取代 <、> 这样的运算符。

java比较器的两种实现

比较器有两种实现方式：
1.让相应的类实现Comparable接口，重写接口中的compareTo(T o)方法。
2.由于第一种方法需要修改类的代码，那么第二种方法就另辟蹊径：再定义一个需要作比较类的比较器，让其实现比较器接口Comparator，重写接口中的比较器接口Compare(T o1, T o2)方法在需要使用时，将该需要作比较类与该比较器放在一起即可。

Comparable 是排序接口。

若一个类实现了Comparable接口，就意味着“该类支持排序”。 即然实现Comparable接口的类支持排序，假设现在存在“实现Comparable接口的类的对象的List列表(或数组)”，则该List列表(或数组)可以通过 Collections.sort（或 Arrays.sort）进行排序。

Comparable 定义

Comparable 接口仅仅只包括一个函数，它的定义如下：

```
package java.lang;
import java.util.*;

public interface Comparable<T> {
    public int compareTo(T o);
}
```

说明：

假设我们通过 x.compareTo(y) 来“比较x和y的大小”。若返回“负数”，意味着“x比y小”；返回“零”，意味着“x等于y”；返回“正数”，意味着“x大于y”。

Comparator 是比较器接口。

我们若需要控制某个类的次序，而该类本身不支持排序(即没有实现Comparable接口)；那么，我们可以建立一个“该类的比较器”来进行排序。这个“比较器”只需要实现Comparator接口即可。

也就是说，我们可以通过“实现Comparator类来新建一个比较器”，然后通过该比较器对类进行排序。

Comparator 定义

Comparator 接口仅仅只包括两个函数，它的定义如下：

```
package java.util;
public interface Comparator<T> {
    int compare(T o1, T o2);
    boolean equals(Object obj);
}
```

说明：

(01) 若一个类要实现Comparator接口：它一定要实现compareTo(T o1, T o2) 函数，但可以不实现 equals(Object obj) 函数。

为什么可以不实现 equals(Object obj) 函数呢？ 因为任何类，默认都是已经实现了equals(Object obj)的。 Java中的一切类都是继承于java.lang.Object，在Object.java中实现了equals(Object obj)函数；所以，其它所有的类也相当于都实现了该函数。

(02) int compare(T o1, T o2) 是“比较o1和o2的大小”。返回“负数”，意味着“o1比o2小”；返回“零”，意味着“o1等于o2”；返回“正数”，意味着“o1大于o2”。

Comparator 和 Comparable 比较

Comparable是排序接口；若一个类实现了Comparable接口，就意味着“该类支持排序”。

而Comparator是比较器；我们若需要控制某个类的次序，可以建立一个“该类的比较器”来进行排序。

我们不难发现：Comparable相当于“内部比较器”，而Comparator相当于“外部比较器”。

```
public class ComparatorTest1 {
    public ComparatorTest1() {}
    public static void main(String[] args) {
        Person[] group = {
                new Person(10,"小明"),
                new Person(10,"小智"),
                new Person(12,"tom"),
                new Person(11,"美美")
        };
        Arrays.sort(group);//默认升序
        //按顺序打印Person信息
        for(Person p : group){
            System.out.println(p.getName() +"今年"+ p.getAge()+"岁~");
        }
    }
}
```

```
class Person implements Comparable<Person>{


 @Override
   //比较器中比较大小的依据
   public int compareTo(Person o) {
      if(this.age > o.age)
         return 1;
      else if(this.age < o.age)
         return -1;
      else{
         return this.name.compareTo(o.name);
      }
   }
}
```

```
public class ComparatorTest2 {
    public ComparatorTest2() {}
    public static void main(String[] args) {
        PersonN[] group = {
                new PersonN(10,"小明"),
                new PersonN(10,"小智"),
                new PersonN(12,"tom"),
                new PersonN(11,"美美")
        };
        PersonNComparator pc = new PersonNComparator();
        Arrays.sort(group,pc);//默认升序
        //按顺序打印Person信息
        for(PersonN p : group){
            System.out.println(p.getName() +"今年"+ p.getAge()+"岁~");
        }
    }
}
```

```
//PersonN类的比较器类
class PersonNComparator implements Comparator<PersonN>{
    @Override
    public int compare(PersonN o1, PersonN o2) {
        if(o1.getAge() > o2.getAge())
            return 1;
        else if(o1.getAge() < o2.getAge())
            return -1;
        else{
            return o1.getName().compareTo(o2.getName());
        }
    }
}
```

1、java.util.Collection 是一个集合接口。它提供了对集合对象进行基本操作的通用接口方法。Collection接口在Java 类库中有很多具体的实现。Collection接口的意义是为各种具体的集合提供了最大化的统一操作方式。

Collection

├List

| ├LinkedList

| ├ArrayList

| └Vector

| └Stack

└Set

2、java.util.Collections 是一个包装类。它包含有各种有关集合操作的静态多态方法。此类不能实例化，就像一个工具类，服务于Java的Collection框架。

Collections.sort：将集合按照自然顺序或者给定的顺序排序。

java.util.Arrays 包含了许多处理数据的实用方法：

Arrays.asList：可以从 Array 转换成 List。可以作为其他集合类型构造器的参数。

Arrays.sort：对整个数组或者数组的一部分进行排序。也可以使用此方法用给定的比较器对对象数组进行排序。

想要明白hashCode的作用，你必须要先知道Java中的集合。总的来说，Java中的集合（Collection）有两类，一类是List，再有一类是Set。 前者集合内的元素是有序的，元素可以重复；后者元素无序，但元素不可重复。

那么这里就有一个比较严重的问题了：要想保证元素不重复，可两个元素是否重复应该依据什么来判断呢？

这就是Object.equals方法了。但是，如果每增加一个元素就检查一次，那么当元素很多时，后添加到集合中的元素比较的次数就非常多了。也就是说，如果集合中现在已经有1000个元素，那么第1001个元素加入集合时，它就要调用1000次equals方法。这显然会大大降低效率。

于是，Java采用了哈希表的原理。哈希（Hash）实际上是个人名，由于他提出一哈希算法的概念，所以就以他的名字命名了。 哈希算法也称为散列算法，是将数据依特定算法直接指定到一个地址上。初学者可以这样理解，hashCode方法实际上返回的就是对象存储的物理地址（实际可能并不是）。 这样一来，当集合要添加新的元素时，先调用这个元素的hashCode方法，就一下子能定位到它应该放置的物理位置上。

如果这个位置上没有元素，它就可以直接存储在这个位置上，不用再进行任何比较了；如果这个位置上已经有元素了， 就调用它的equals方法与新元素进行比较，相同的话就不存了，不相同就散列其它的地址。所以这里存在一个冲突解决的问题。这样一来实际调用equals方法的次数就大大降低了，几乎只需要一两次。

所以，Java对于eqauls方法和hashCode方法是这样规定的：

1、如果两个对象相同，那么它们的hashCode值一定要相同；2、如果两个对象的hashCode相同，它们并不一定相同　上面说的对象相同指的是用eqauls方法比较。

hash code(散列码，也可以叫哈希码值)是对象产生的一个整型值。其生成没有规律的。二者散列码可以获取对象中的信息，转成那个对象的“相对唯一”的整型值。所有对象都有一个散列码，hashCode()是根类 Object 的一个方法。

钱进培训是哈法地区资深工程师组成的培训机构，通过各位老师的现身说法，帮助各位学员迅速掌握实战知识，为求职打下坚实的基础。电子邮件：jin.qian.canada@gmail.com钱老师报名、答疑微信号：qianjincanada，或扫描以下二维码添加：

钱老师

加拿大

扫一扫上面的二维码图案，加我微信