

第一章 认识java

1 历史

Sun Microsystems公司于1995年5月推出的Java程序设计语言和Java开发平台。Java是一种面向对象的编程语言，它的前身是詹姆斯·高斯林（James Gosling，人称java之父）等人于1990年代初开发的一种编程语言，最初被命名为Oak。

Java历史中的一些时间节点：

1. 1991年4月，由James Gosling博士领导的绿色计划（Green Project）开始启动，此计划的目的是开发一种能够在各种消费性电子产品（如机顶盒、冰箱、收音机等）上运行的程序架构。这个计划的产品就是Java语言的前身：Oak（橡树）。Oak当时在消费品市场上并不算成功，但随着1995年互联网潮流的兴起，Oak迅速找到了最适合自己的市场定位并蜕变成为Java语言。
2. 1995年5月23日，Oak语言改名为Java，并且在SunWorld大会上正式发布Java 1.0版本。Java语言第一次提出了“Write Once, Run Anywhere”的口号。
3. 1996年1月23日，JDK 1.0发布，Java语言有了第一个正式版本的运行环境。JDK 1.0提供了一个纯解释执行的Java虚拟机实现（Sun Classic VM）。JDK 1.0版本的代表技术包括：Java虚拟机、Applet、AWT等。
4. 1996年4月，10个最主要的操作系统供应商申明将在其产品中嵌入Java技术。同年9月，已有大约8.3万个网页应用了Java技术来制作。在1996年5月底，Sun公司于美国旧金山举行了首届JavaOne大会，从此JavaOne成为全世界数百万Java语言开发者每年一度的技术盛会。
5. 1997年2月19日，Sun公司发布了JDK 1.1，Java技术的一些最基础的支撑点（如JDBC等）都是在JDK 1.1版本中发布的，JDK 1.1版的技术代表有：JAR文件格式、JDBC、JavaBeans、RMI。Java语法也有了一定的发展，如内部类（Inner Class）和反射（Reflection）都是在这个时候出现的。
6. 1999年4月8日，JDK 1.1在此期间一共发布了1.1.0~1.1.8九个版本。从1.1.4之后，每个JDK版本都有一个自己的名字（工程代号），分别为：JDK 1.1.4 - Sparkler（宝石）、JDK 1.1.5 - Pumpkin（南瓜）、JDK 1.1.6 - Abigail（阿比盖尔，女子名）、JDK 1.1.7 - Brutus（布鲁图，古罗马政治家和将军）和JDK 1.1.8 - Chelsea（切尔西，城市名）。
7. 1998年12月4日，JDK迎来了一个里程碑式的版本JDK 1.2，工程代号为Playground（竞技场），Sun在这个版本中把Java技术体系拆分为3个方向，分别是面向桌面应用开发的J2SE（Java 2 Platform, Standard Edition）、面向企业级开发的J2EE（Java 2 Platform, Enterprise Edition）和面向手机等移动终端开发的J2ME（Java 2 Platform, Micro Edition）。
8. 1999年3月和7月，分别有JDK 1.2.1和JDK 1.2.2两个小版本发布。
9. 1999年4月27日，HotSpot虚拟机发布，HotSpot最初由一家名为“Longview Technologies”的小公司开发，因为HotSpot的优异表现，这家公司在1997年被Sun公司收购了。HotSpot虚拟机发布时是作为JDK 1.2的附加程序提供的，后来它成为了JDK 1.3及之后所有版本的Sun JDK的默认虚拟机。
10. 2000年5月8日，工程代号为Kestrel（美洲红隼）的JDK 1.3发布，JDK 1.3相对于JDK 1.2的改进主要表现在一些类库上（如数学运算和新的Timer API等）。JDK 1.3有1个修正版本JDK 1.3.1，工程代号为Ladybird（瓢虫），于2001年5月17日发布。自从JDK 1.3开始，Sun维持了一个习惯：大约每隔两年发布一个JDK的主版本，以动物命名，期间发布的各个修正版本则以昆虫作为工程名称。
11. 2002年2月13日，JDK 1.4发布，工程代号为Merlin（灰背隼）。JDK 1.4是Java真正走向成熟的一个版本，Compaq、Fujitsu、SAS、Symbian、IBM等著名公司都有参与甚至实现自己独立的JDK 1.4。
12. 2002年9月16日发布的工程代号为Grasshopper（蚱蜢）的JDK 1.4.1
13. 2003年6月26日发布的工程代号为Mantis（螳螂）的JDK 1.4.2。

14. 2004年9月30日，JDK 1.5 发布，工程代号Tiger（老虎）。从JDK 1.2以来，Java在语法层面上的变换一直很小，而JDK 1.5在Java语法易用性上做出了非常大的改进。例如，自动装箱、泛型、动态注解、枚举、可变长参数、遍历循环（foreach循环）等语法特性都是在JDK 1.5中加入的。在虚拟机和API层面上，这个版本改进了Java的内存模型（Java Memory Model，JMM）、提供了java.util.concurrent并发包等。
15. 2006年12月11日，JDK 1.6发布，工程代号Mustang（野马）。在这个版本中，Sun终结了从JDK 1.2开始已经有8年历史的J2EE、J2SE、J2ME的命名方式，启用Java SE 6、Java EE 6、Java ME 6 的命名方式。
16. 2006年11月13日，在JavaOne大会上，Sun公司宣布最终会将Java开源，并在随后的一年多时间内，陆续将JDK的各个部分在GPL v2（GNU General Public License v2）协议下公开了源码，并建立了OpenJDK组织对这些源码进行独立管理。除了极少量的产权代码（Encumbered Code，这部分代码大多是Sun本身也无权进行开源处理的）外，OpenJDK几乎包括了Sun JDK的全部代码，OpenJDK的质量主管曾经表示，在JDK 1.7中，Sun JDK和OpenJDK除了代码文件头的版权注释之外，代码基本上完全一样，所以OpenJDK 7与Sun JDK 1.7本质上就是同一套代码库开发的产品。
17. JDK 1.6发布以后，由于代码复杂性的增加、JDK开源、开发JavaFX、经济危机及Sun收购案等原因，Sun在JDK发展以外的事情上耗费了很多资源，JDK的更新没有再维持两年发布一个主版本的发展速度。JDK 1.6到目前为止一共发布了37个Update版本，最新的版本为Java SE 6 Update 37，于2012年10月16日发布。
18. 2009年2月19日，工程代号为Dolphin（海豚）的JDK 1.7完成了其第一个里程碑版本。根据JDK 1.7的功能规划，一共设置了10个里程碑。最后一个里程碑版本原计划于2010年9月9日结束，但由于各种原因，JDK 1.7最终无法按计划完成。
19. 2009年4月20日，Oracle公司宣布正式以74亿美元的价格收购Sun公司，Java商标从此正式归Oracle所有（Java语言本身并不属于哪间公司所有，它由JCP组织进行管理，尽管JCP主要是由Sun公司或者说Oracle公司所领导的）。由于此前Oracle公司已经收购了另外一家大型的中间件企业BEA公司，在完成对Sun公司的收购之后，Oracle公司分别从BEA和Sun中取得了目前三大商业虚拟机的其中两个：JRockit和HotSpot，Oracle公司宣布在未来1~2年的时间内，将把这两个优秀的虚拟机互相取长补短，最终合二为一。可以预见在不久的将来，Java虚拟机技术将会产生相当巨大的变化。
20. 2011年7月28日，Oracle公司发布Java SE 1.7
21. 2014年3月18日，Oracle公司发布Java SE 1.8
22. 2017年9月21日，Oracle公司发布Java9
23. 2018年3月20日，Oracle公司发布Java10
24. 2018年9月25日，Oracle公司发布Java11
25. 2019年3月19日，Oracle公司发布Java12
26. 2019年9月17日，Oracle公司发布Java13
27. 2020年3月17日，Oracle公司发布Java14

目前开发中使用的JDK，仍是以JDK8为主要的选择版本。

2 平台

1998年12月4日，Sun公司在发布的JDK1.2版本中，将Java技术体系拆分为3个方向（平台）：

1. J2SE（Java 2 Platform，Standard Edition），面向桌面应用开发
2. J2EE（Java 2 Platform，Enterprise Edition），面向企业级开发
3. J2ME（Java 2 Platform，Micro Edition），面向手机等移动终端开发

注意，我们所学习的corejava（核心java）相关的内容，就包含在J2SE之中。

- 2006年12月11日，Sun公司在发布的JDK 1.6版本中，启用Java SE、Java EE、Java ME的命名方式
- 2017年9月，oracle宣布将JavaEE所有权转交给Eclipse基金会
- 2018年3月，Eclipse基金会宣布将JavaEE更名为JakartaEE，Jakarta（雅加达）是一座城市的名字
- 2019年9月，Jakarta EE 8 完全开源
- 2020年6月，Jakarta EE 9 发布

3 JDK

3.1 相关名词

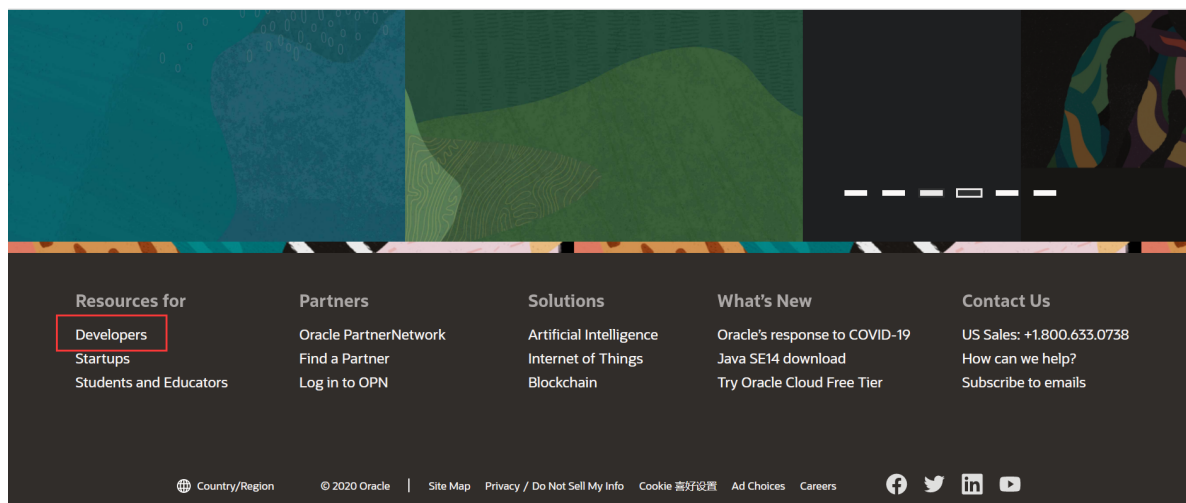
- SDK（software development kit），软件开发包，主要包含函数库或者工具等
- JDK（java development kit），java程序开发工具包，面向java程序的开发者
- JRE（java runtime environment），java程序运行环境，面向java程序的使用者
- API（application program interface），应用程序编程接口
- API Documentation，API说明文档，描述API中的类、方法等使用的方式

注意，JDK中会自带一个JRE的，也可以专门独立的只安装JRE，但开发人员一定是安装JDK

3.2 JDK下载

到oracle官网中，注册账号并登录，然后到相应的页面中，即可下载。

oracle官网



Technologies



Databases



Java



Cloud Native and Containers



Blockchain



Open Source



Chatbots



Javascript



Devops



API



Low Code



AI/ML



Linux

Developer Community and Events

Java Standard Edition (Java SE)

Java SE lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

- [OpenJDK Download](#)
- [Java SE Download](#)
- [Documentation](#)
- [Java SE Subscription](#)

Search Sign In Country/Region Contact

Oracle Technology Network / Java / Java SE / Overview

- Java SE
- Java EE
- Java ME
- Java SE Subscription
- Java Embedded
- Java Card
- Java TV
- Community
- Java Magazine

Overview **Downloads** Documentation Community Technologies Training

Java SE at a Glance

[General FAQs](#)

Java Platform, Standard Edition (Java SE) lets you develop and deploy Java applications on [desktops](#) and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

What's New
Java Platform, Standard Edition 14
Java SE 14.0.2 is the latest release of Java SE Platform. Oracle strongly recommends that all Java SE users upgrade to this release.

[Download](#)
[Release Notes](#) [Press Release](#)

Updates
Java SE 14
Java SE 14.0.2 is the latest release for Java SE Platform.

[Release Notes](#) [Download](#)

Java SE 11
Java SE 11.0.8 is the latest release for JDK 11

Java SDKs and Tools

- [Java SE](#)
- [Java EE and Glassfish](#)
- [Java ME](#)
- [Java Card](#)
- [NetBeans IDE](#)
- [Java Mission Control](#)

Java Resources



- [Java APIs](#)
- [Technical Articles](#)
- [Demos and Videos](#)
- [Forums](#)
- [Java Magazine](#)
- [Developer Training](#)
- [Tutorials](#)
- [Java.com](#)









Java SE 8u261

Java SE 8u261 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.

- [Documentation](#)
- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
 - [Binary License](#)
 - [Documentation License](#)
 - [BSD License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme Files](#)
 - [JDK ReadMe](#)
 - [JRE ReadMe](#)

Oracle JDK

-  [JDK Download](#)
-  [Server JRE Download](#)
-  [JRE Download](#)
-  [Documentation Download](#)
-  [Demos and Samples Download](#)


Linux x64 Compressed Archive	136.48 MB	 jdk-8u261-linux-x64.tar.gz
macOS x64	203.94 MB	 jdk-8u261-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	125.77 MB	 jdk-8u261-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	88.72 MB	 jdk-8u261-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.23 MB	 jdk-8u261-solaris-x64.tar.Z
Solaris x64	92.47 MB	 jdk-8u261-solaris-x64.tar.gz
Windows x86	154.52 MB	 jdk-8u261-windows-i586.exe
Windows x64	166.28 MB	 jdk-8u261-windows-x64.exe

JDK历史版本下载：


JDK历史版本

THE TZUPDATER TOOL IS PROVIDED BY ORACLE CORPORATION AND IS SUBJECT TO THE TERMS AND CONDITIONS OF THE LICENSE AGREEMENT.

- [Readme](#)
- [License](#)

TZ Updater
 [TZ Updater Download](#)

Java API Documentation Updater Tool 1.3
Java API Documentation Updater Tool repairs-in-place Java API Documentation created with javadoc versions included with JDK 5u45, 6u45, 7u21 and earlier. See the 7u25 release notes for more information.

Java API Documentation Updater Tool
 [Java API Documentation Updater Tool](#)

Java Archive
The [Java Archive](#) offers access to some of our historical Java releases. **WARNING:** These older versions of the JRE and JDK are provided to help developers debug issues in older systems. **They are not updated with the latest security patches and are not recommended for use in production.**

Java SE

Java EE

Java ME

Java FX

JAVA SE

Java SE 14	Java SE 8 (8u211 and later)	Java SE 1.3
Java SE 13	Java SE 8 (8u202 and earlier)	Java SE 1.2
Java SE 12	Java SE 7	Java SE 1.1
Java SE 11	Java SE 6	JRockit Family
Java SE 10	Java SE 5	Java SE Tutorials
Java SE 9	Java SE 1.4	JDK 1.3 Documentation
		JDK 1.4.2 Documentation

Java Client Technologies

Java 3D, Java Access Bridge, Java Accessibility, Java Advanced Imaging, Java Internationalization and Localization Toolkit, Java Look and Feel, Java Media Framework (JMF), Java Web Start (JAWS), JIMI SDK

Java Platform Technologies

Java Authentication and Authorization Service (JAAS), JavaBeans, Java Management Extension (JMX), Java Naming and

API在线文档

Java Standard Edition (Java SE)

Java SE lets you develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

[OpenJDK Download](#)

[Java SE Download](#)

[Documentation](#)

[Java SE Subscription](#)



Latest Release

JDK 14

Previous Releases

JDK 13

JDK 12

JDK 11

JDK 10

JDK 9

JDK 8

JDK 7

Java Platform, Standard Edition (Java SE) 8

[Home](#) [Client Technologies](#) [Embedded](#) [All Books](#)

About Java SE 8

- [What's New \(Features and Enhancements\)](#)
- [Commercial Features](#)
- [Compatibility Guide](#)
- [Known Issues](#)

Download and Install

- [Certified System Configurations](#)
- [Download and Installation Instructions](#)

Write Your First Application

- [Get Started with Java](#)
- [Get Started with JavaFX](#)

Learn the Language

- [Java Tutorials Learning Paths](#)

Monitor and Troubleshoot

- [Java Mission Control](#)
- [Java Flight Recorder](#)
- [Troubleshooting Guide](#)

HotSpot Virtual Machine

- [HotSpot Virtual Machine Garbage Collection Tuning Guide](#)
- [JRockit to HotSpot Migration Guide](#)

Deploy

- [Deployment Guide](#)

Reference

- [Java SE API Documentation](#)
- [JavaFX API Documentation](#)
- [Developer Guides](#)
- [Java Language and Virtual Machine Specifications](#)
- [Java SE Tools Reference for UNIX](#)
- [Java SE Tools Reference for Windows](#)

Release Notes

- [Java SE Release Notes](#)

API说明文档下载

Java SE 8u261

Java SE 8u261 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release.

- [Documentation](#)
- [Installation Instructions](#)
- [Release Notes](#)
- [Oracle License](#)
 - [Binary License](#)
 - [Documentation License](#)
 - [BSD License](#)
- [Java SE Licensing Information User Manual](#)
 - [Includes Third Party Licenses](#)
- [Certified System Configurations](#)
- [Readme Files](#)
 - [JDK ReadMe](#)
 - [JRE ReadMe](#)

Oracle JDK



JDK Download



Server JRE Download



JRE Download



Documentation Download



Demos and Samples Download

Java SE Development Kit 8 Documentation

Java SE Development Kit 8u261 Documentation

This software is licensed under the [Java SE Development Kit 8 Documentation License Agreement](#)

Product / File Description	File Size	Download
Documentation	118.75 MB	jdk-8u261-docs-all.zip

Development Kit 8 Documentation

Development Kit

under the J.

Description

You must accept the [Java SE Development Kit 8 Documentation License Agreement](#) to download this software.

☒ I reviewed and accept the Java SE Development Kit 8 Documentation License Agreement

Download jdk-8u261-docs-all.zip



Documentation

Development Kit 8 D

Development Kit

der the J.

tion

You must accept

☒ I reviewed

新建下载任务

网址: <https://download.oracle.com/otn-pub/java/jdk/8u261-b12/>

名称: [jdk-8u261-docs-all.zip](#) 压缩文件 118.75 MB

下载到: C:\Users\thinkpad\Desktop 剩: 21.55 GB 浏览

使用迅雷下载

直接打开

下载

取消

Documentation

3.3 JDK安装

Linux中安装，把下载好的压缩包，直接解压并解归档即可。

例如，解压到当前目录下，也可以解压到指定目录中：

```
briup@briup:~/jdk/test$ tar -zxvf jdk-8u74-linux-x64.tar.gz
```

Windows中安装，直接双击安装包，一直下一把即可，默认安装位置：C:\Program Files\Java，会在此目录中自动创建JDK的目录，并且目录的名字中携带JDK的版本号。

本地磁盘 (C:) > Program Files > Java

名称

- jdk1.7.0_79
- jdk1.8.0_74**
- jre1.8.0_74
- jre7

注意，安装过程中也会默认安装一个外部的jre，目录名字为jre1.8.0_74

注意，JDK按照目录里面，也自带了一个默认的内部的JRE，点击JDK目录后即可看到此JRE的目录

注意，电脑中可以按照多个不同版本的JDK，不会影响的，之后使用的时候，指定具体的版本即可。

3.4 环境变量

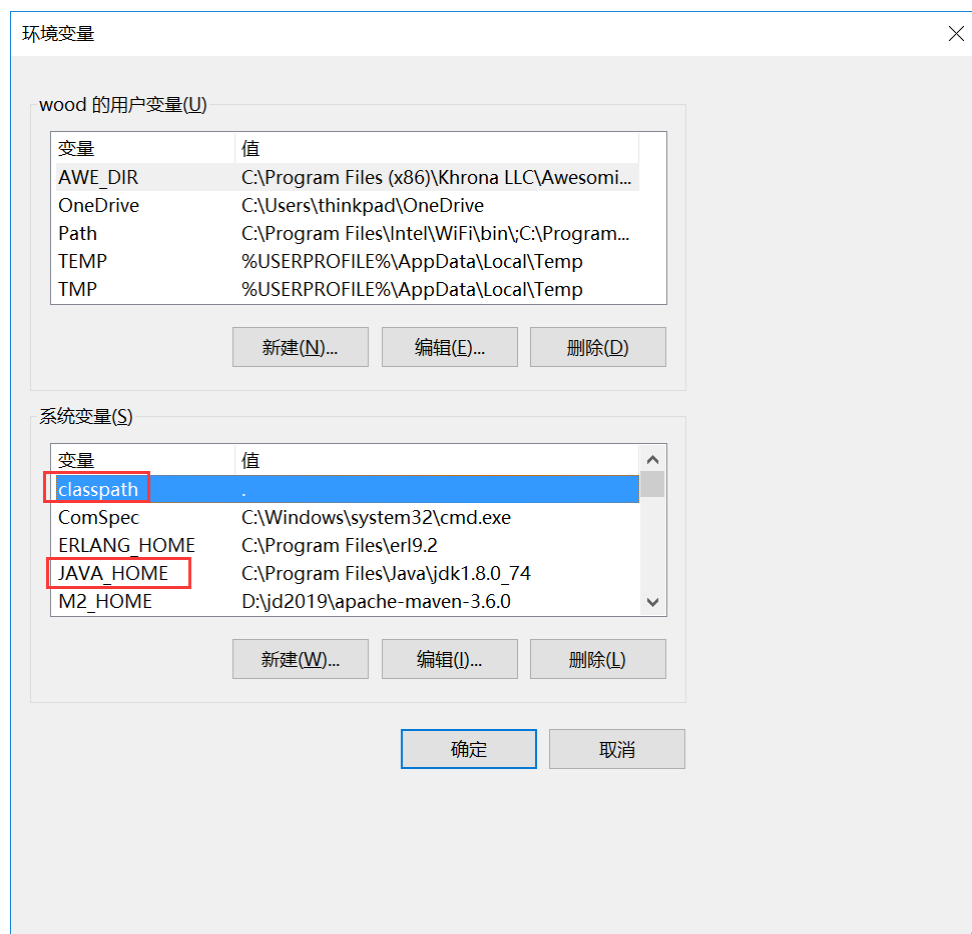
需要配置三个环境变量，JAVA_HOME、PATH、CLASSPATH，其中PATH是必须，否找不到java的相关命令所在的文件路径

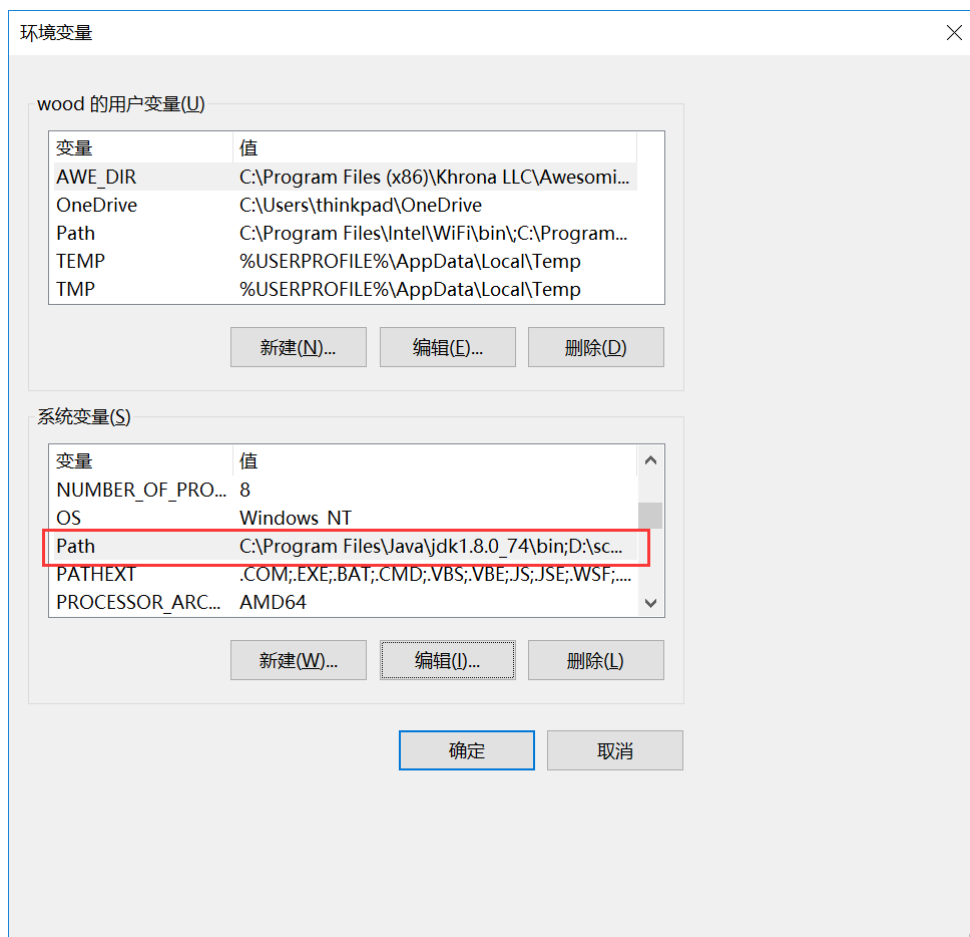
以Windows系统为例说明：

JAVA_HOME，指定JDK的安装目录，例如，JAVA_HOME=C:\Program Files\Java\jdk1.8.0_74

PATH，把JDK中java命令所在目录配置到原有的PATH中，可以配置到PATH的最前面。例如，PATH=%JAVA_HOME%\bin;...;...;...; 这里也可以不引用JAVA_HOME的变量值。

CLASSPATH，指定将来要运行加载的class文件所在位置，这个路径将来可能随时变换，可以先配置为当前路径，将来使用的时候再具体配置即可。例如，CLASSPATH=.





注意，在Windows中，可以使用where命令，查看java名字所在位置
例如，where java

Linux中的环境变量配置：

使用vi打开.bashrc文件进行配置

```
109 if [ -f /usr/share/bash-completion/bash_completion ]
110 then
111 . /usr/share/bash-completion/bash_completion
112 elif [ -f /etc/bash_completion ]; then
113 . /etc/bash_completion
114 fi
115
116 JAVA_HOME=/home/briup/jdk/jdk1.8.0_74
117 PATH=$JAVA_HOME/bin:$PATH
118 CLASSPATH=.
119
120 export JAVA_HOME PATH CLASSPATH
121
```

121,0-1 底端

配置完成之后，在命令窗口中，使用命令进行查看：

```
java -version
```

```
briup@briup:~$ java -version
java version "1.8.0_74"
Java(TM) SE Runtime Environment (build 1.8.0_74-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.74-b02, mixed mode)
briup@briup:~$
```

```
C:\Users\thinkpad>java -version
java version "1.8.0_74"
Java(TM) SE Runtime Environment (build 1.8.0_74-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.74-b02, mixed mode)
```

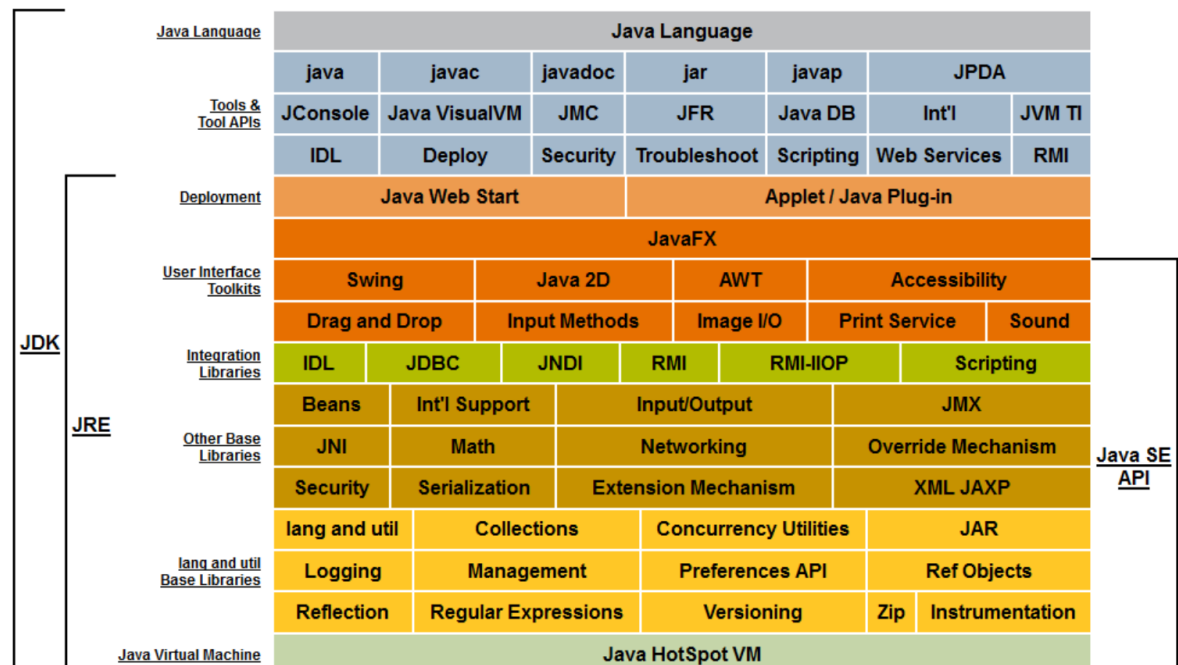
```
C:\Users\thinkpad>
```

注意，如果是Windows系统，配置好了之后，需要重新打开一个新的命令窗口进行测试

注意，如果是Linux系统，配置好了之后，可以先source命令让.bashrc文件生效，或者重写打开新的命令窗口

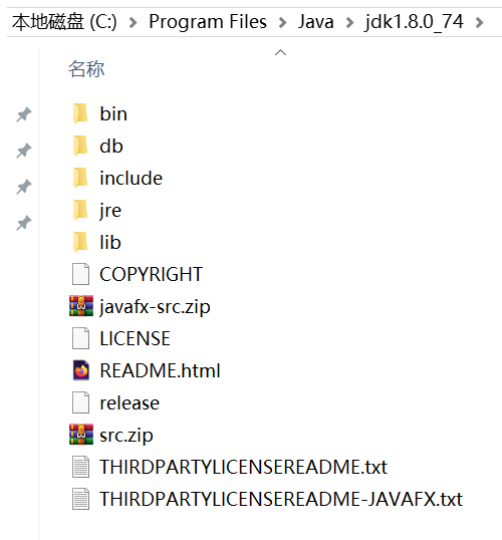
3.5 JDK结构

oracle官方文档中提供的JDK结构图：



3.6 JDK目录

这里以Windows中安装后的目录为例说明：



目录/文件	作用
bin目录	存放JDK中提供的java相关的命令，例如java、javac、javap等
db目录	JDK自带的一个小型数据库，纯java实现
include目录	JVM启动时需要引入一些C语言的头文件，该目录就是用于存放这些头文件的。
jre目录	JDK自带的一个java运行环境
lib目录	“library”的缩写，目录中提供了一些Java类库和库文件，也就是jar包。
src.zip文件	压缩文件，目录存放JDK核心类的源代码，也就是JavaSE-API的源代码

关于src.zip文件：

- 我们将来在代码中，所调用的JavaSE-api中的代码，大多数的源码就存放在这个压缩包中。
- 这里面都是java文件，这些java文件编译后生成的class文件，都存放在一个jar中，C:\Program Files\Java\jdk1.8.0_74\jre\lib\rt.jar
- 我们的代码在运行的之前，JVM会先从这个rt.jar中加载一些我们需要使用的类，例如String、Object、System等

sun公司程序员--编写基础的代码--》*.java --压缩-- 》src.zip --编译--》 *.class --归档--》 rt.jar

一定要记得src.zip和rt.jar中的内容是什么，和它们两者之间的关系。

4 API文档

将来在我们编写的代码中，要使用/调用JavaSE-API中所提供的代码，它的源代码存放在src.zip中，编译后的字节码存放在rt.jar中，这些类、接口的介绍和使用说明，都在JavaSE-API说明文档中，它就像是一本说明书，来指导我们该如何调用这些别人提供给我们的基本代码。

Java™ Platform
Standard Ed. 8

All ClassesAll Profiles

Packages

java.applet
java.awt
java.awt.color

All Classes

AbstractAction
AbstractAnnotationValueVisitor6
AbstractAnnotationValueVisitor7
AbstractAnnotationValueVisitor8
AbstractBorder
AbstractButton
AbstractCellEditor
AbstractChronology
AbstractCollection
AbstractColorChooserPanel
AbstractDocument
AbstractDocument.AttributeContext
AbstractDocument.Content
AbstractDocument.ElementEdit
AbstractElementVisitor6
AbstractElementVisitor7
AbstractElementVisitor8
AbstractExecutorService
AbstractInterruptibleChannel
AbstractLayoutCache
AbstractLayoutCache.NodeDimension

OVERVIEWPACKAGECLASSUSETREEDEPRECATEDINDEXHELP

PREVNEXTFRAMESNO FRAMES

Java™ Platform, Standard Edition 8
API Specification

This document is the API specification for the Java™ Platform, Standard Edition.
See: Description

Profiles

- compact1
- compact2
- compact3

Packages

Package	Description
java.applet	Provides the classes necessary to create an applet and the classes an applet uses to communicate with its applet context.
java.awt	Contains all of the classes for creating user interfaces and for painting graphics and images.
java.awt.color	Provides classes for color spaces.
java.awt.datatransfer	Provides interfaces and classes for transferring data between and within

java.awt.color
java.awt.im.spi
java.awt.image
java.awt.image.renderable
java.awt.print
java.beans
java.beans.beancontext
java.io
java.lang
java.lang.annotation

Long
Math
Number
Object
Package
Process
ProcessBuilder
ProcessBuilder.Redirect
Runtime
RuntimePermission
SecurityManager
Short
StackTraceElement
StrictMath
String
StringBuffer
StringBuilder
System
Thread
ThreadGroup
ThreadLocal
Throwable
Void

public final class System
extends Object

The System class contains several useful class fields and methods. It cannot be instantiated.

Among the facilities provided by the System class are standard input, standard output, and error output streams; access to externally defined properties and environment variables; a means of loading files and libraries; and a utility method for quickly copying a portion of an array.

Since:
JDK1.0

Field Summary

Fields	
static <code>PrintStream</code>	<code>err</code> The "standard" error output stream.
static <code>InputStream</code>	<code>in</code> The "standard" input stream.
static <code>PrintStream</code>	<code>out</code> The "standard" output stream.

Method Summary

5 java特点

我们平时所说的“java”，其实是一个综合的描述，它包含了一系列的很多东西。

- 它是一门编程语言
包含基本的语法、特性、思想等
- 它是开发环境
开发java程序时，需要一些java开发工具
- 它是应用环境
开发好的java程序要运行，需要它提供一些的运行环境
- 它是部署环境
将来在其他平台下开发出的java程序，都需要它提供一个最基础的部署环境

也就是说，只有是进行和java相关的一系列相关活动，就必须要先安装一个JDK，以保证这些最基础的环境、工具、语言支持等。

java语言的优点：

- 更纯粹的面向对象编程，加速开发的过程。
- 一次编写/编译，到处运行，代码可以跨平台
- 多线程的支持
- 代码中没有指针管理、内存管理，使得编程人员可以更加专注于系统的业务功能的开发
- 字节码的验证机制，保证代码的安全性
- 开源及强大的生态环境，社区活跃，第三方类库选择丰富

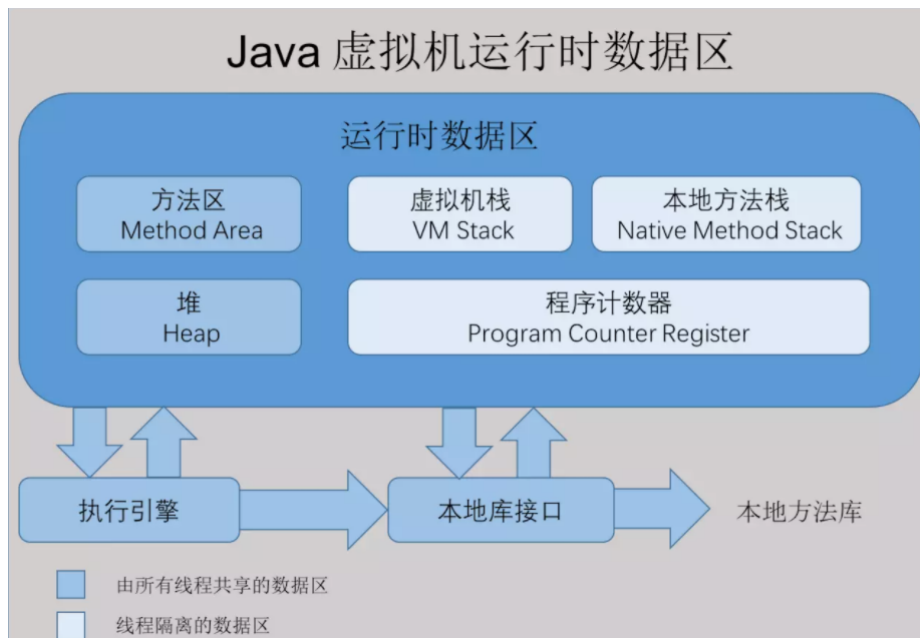
6 JVM

JVM是Java Virtual Machine (Java虚拟机) 的缩写，它是一个虚构出来的计算机规范结构，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。

JVM是java中最核心的一个东西，它在计算机的内存中，虚拟并提供了java代码可以在其中运行的基础环境。

思考：java代码编译后，可以在不同的操作系统平台中运行的原因是什么？（跨平台）

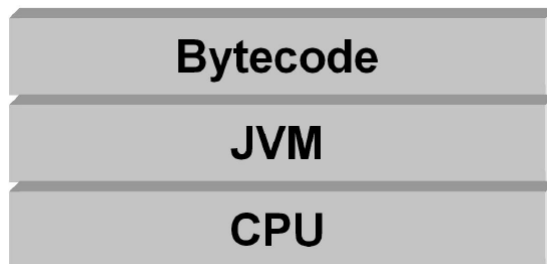
在JDK7中对JVM规范所给出的内存管理结构如下：



和我们程序中关系比较紧密的是堆区、栈区还有方法区，后面我们会经常提及这几个内存区域

我们将来编写好的java代码，编译成了class文件，这些class文件就需要加载到JVM中，再进行运行

JVM就是java代码和计算机之间的一个桥梁：



java代码编译后，计算机并不能直接运行，必须需要经过JVM进行解释后，再进行运行。
所以，java其实并不算是真正的编译语言。

注意，JVM本质上是一个规范，每个公司都可以按照这个规范实现自己的JVM虚拟机

我们现在默认使用的JVM就是oracle公司提供的实现，这个JVM叫做HotSpot，从上面的历史介绍中，也可以找到这个HotSpot虚拟机的来历。

```
C:\Users\thinkpad>java -version
java version "1.8.0_74"
Java(TM) SE Runtime Environment (build 1.8.0_74-b02)
Java HotSpot(TM) 64-Bit Server VM (build 25.74-b02, mixed mode)
```

当使用java命令来运行程序的时候，会先启动JVM，这个JVM在JDK中对应了一个dll或so文件：

- 如果是Windows系统下，这个文件在：%JAVA_HOME%\jre\bin\server\jvm.dll
- 如果是Linux系统下，这个文件在：\$JAVA_HOME/jre/lib/amd64/server/libjvm.so

注意，Linux下面的so文件就类似于Windows中的dll文件

7 垃圾回收器

在java语言中，编程人员不需要在代码中控制内存的开辟和释放。

java代码中，开辟要使用的内存空间，使用new关键字即可完成。

使用完之后，对内存的释放，在JVM中，由垃圾回收器（Garbage Collection，GC）来完成。

不同类型的GC，在JVM中，会根据不同的算法，对不同的内存区域内标记为垃圾的空间，进行回收释放。在这个过程中，是不需要编程人员干预的，它自己会主动的完成。

在代码中，我们也可以调用JavaSE-API提供的方法，通知GC现在去进行垃圾回收的工作：

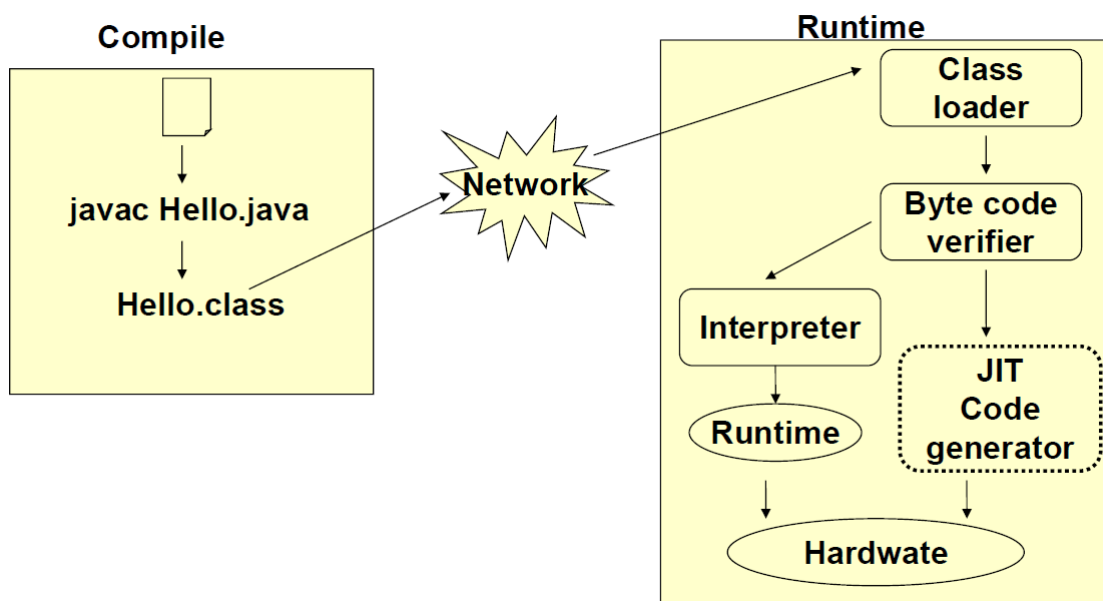
```
1 java.lang.System.gc();
2 java.lang.Runtime.gc();
```

注意，虽然可以主动通知，但是最后GC并不一定会真的立刻执行，因为这个垃圾回收的过程什么时候执行，最终还是要根据GC的具体算法和当前内存的使用情况来确定的

8 字节码验证

编写好的java代码，编译成class文件（字节码）后，再被JVM加载到内存中的时候，是需要做字节码验证的

编写并编译好的java代码，加载到JVM内存中：



加载到内存的途径有很多，这里描述的是通过网络，也可以是通过本地磁盘

一个class文件被加载到JVM内存之后，首先要经过字节码验证，主要包含：

- 检查当前class文件的版本和JVM的版本是否兼容
- 检查当前代码是否会破坏系统的完整性
- 检查当前代码是否有栈溢出的情况
- 检查当前代码中的参数类型是否正确
- 检查当前代码中的类型转换操作是否正确

验证通过，再确定哪些代码是解释执行的，哪些代码是JIT即时编译执行的：

- 解释执行
class文件内容，需要让JVM进行解释，解释成计算机可以执行的代码。整体效果就是JVM解释一行代码就执行一行代码。所以如果java代码全是这样的运行方式的话，效率会稍低一些。
- JIT (Just In Time) 即时编译
执行代码的另一种方式，JVM可以把java中的 **热点代码** 直接编译成计算机可以运行的代码，这样

接下来再调用这个热点代码的时候，就可以直接使用编译好的代码让计算机直接运行，可以提高运行效率。

9 编写java程序

创建文件，`mkdir -p code/day01`

如果是Windows，在桌面创建相应的文件夹即可。

`Hello.java`

```
1 public class Hello{
2
3     public static void main(String[] args){
4         System.out.println("hello world");
5     }
6
7 }
```

```
1 //public表示公共的，说明其他代码中也可以使用这个公共的类
2 //class是java中的关键字，表示定义一个类
3 //Hello是类的名字，这个类写在Hello.java文件中，类的名字和文件名字保持一致
4 public class Hello{
5     //定义一个方法，方法的名字叫 main，方法的参数名字叫args，参数的类型是String[]，表示字符串
    数组类型
6     //public表示这是个公共的方法
7     //static表示这是个静态的方法
8     //void表示main方法执行完，没有任何返回值
9     public static void main(String[] args){
10         //main方法中，执行这个语句代码
11         //该代码表示使用System类中的out属性的println方法，将字符串"hello world"进行打印输
    出
12         System.out.println("hello world");
13     }
14 }
```

这里的main方法，也叫做程序入口，代码无论写多少行，JVM运行的时候必须得有一个唯一的入口，也就是代码运行的起点。如果没有这个固定的入口，JVM是没有办法运行我们编写的代码的。

思考：程序运行的入口，是可以随便写么，也就是随便写一个方法，JVM就能把它当做运行的起点么？

编译Hello.java中，编写的代码：

```
javac Hello.java
```

```
briup@briup:~/code/day01$ javac Hello.java  
briup@briup:~/code/day01$ ls  
Hello.class Hello.java  
briup@briup:~/code/day01$
```

编译成功后，生成对应的class文件，也就是字节码文件，其实就是0101代码（二进制）

运行Hello.class文件中，编译生成的字节码：

```
java Hello
```

```
briup@briup:~/code/day01$ java Hello  
hello world  
briup@briup:~/code/day01$
```

字节码，是二进制的0101代码，但是计算机不能直接运行，需要JVM进行解释后再执行

java命令，会先启动JVM虚拟机，然后再进行加载、验证、解释/JIT、运行等一系列的过程

JVM要求的运行规则是，java命令后面加上要运行的类的名字即可，这个类的字节码一定是在class文件中的，同时要求这个类中，需要有一个程序入口，否则运行失败。

思考：JVM是怎么知道Hello这个类在哪个class文件中的？又是怎么知道这个class文件在什么地方的？

Hello这个类 不一定在Hello.java中，但是一定是在Hello.class中

10 常用的命令

- javac
编译命令
- java
运行命令
- javadoc
生成API文档命令
- javap
反解析命令，可以解析出class字节码文件的内容
- jar
打包命令

11 常用的包

在程序中，要区分一些东西，一般会采用【命名空间】的设计方式
在java中，package其实就是类的命名空间，用来唯一标识这个类的，同时也把类似功能的类组织到一个包中

JavaSE-API中常用的包有：

- java.lang
最常用的一个包，里面的类可以在我们的程序中直接使用，不需要import导入
- java.awt、javax.swing、java.awt.event
这三个包属于同一类型的，它们包下面的类都是用来做图形化界面的(GUI)
注意：javax开头的包，是sun公司后面又扩展进来的包，刚开始是没有的
- java.io
这个包下的类主要用于输入输出流的操作。也就是读数据/写数据
- java.net
这个包下的类主要用于网络编程。例如把计算机A和计算机B之间建立网络连接，然后进行信息传输。
- java.util
这个包下的类都是一些常用工具类，可以帮我们在代码中完成一些辅助的功能，例如表示日期、使用集合存储数据、使用工具类操作数组等

