理解Java语言程序设计

王少川 编著

清华大学出版社

理解Java语言程序设计

清华大学出版社
北京

内 容 简 介

本书是Java语言学习的指导书，不但讲述Java语言基本概念，而且介绍学习Java的方法，使读者学会分析问题、解决问题。本书内容共分为16章，分别是Java概述，Java语言要素，Java的基本数据类型、操作符和表达式，控制结构，类，类与类之间的关系，对象，接口，泛型，异常，包，输入和输出，reflection，算法、数据结构和Collection，多线程以及网络。

本书可作为科研人员和从事Java语言程序开发的人员的参考书和计算机相关专业本科生及爱好者的教材。

图书在版编目(CIP)数据

理解Java语言程序设计/王少川编著. —北京：清华大学出版社，2010.11
ISBN 978-7-302-22780-9

Ⅰ. ①理…　Ⅱ. ①王…　Ⅲ. ①JAVA语言—程序设计—高等学校—教材　Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第090427号

责任编辑：闫红梅　王冰飞
责任校对：时翠兰
责任印制：王秀菊
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦A座
http://www.tup.com.cn　　**邮　　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62795954，jsjjc@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：北京市世界知识印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：185×260　**印　张**：31.25　**字　数**：759千字
版　　次：2010年11月第1版　　**印　　次**：2010年11月第1次印刷
印　　数：1～3000
定　　价：49.00元

产品编号：036007-01

前言

本书主要是写给那些了解一些 Java 基础知识，又想进一步提高或正在学习 Java，需要一本参考书的读者。不过，如果你没有读过任何一本 Java 书，也不了解 Java 基础知识，也不要紧，本书作为你读的第一本 Java 的书，也是一种很好的选择。本书编写的宗旨是让初学者也能读懂，让有一定基础的人读了也有收获。

作者是完全通过自学学习 Java 语言的，一般的参考书中对一些基本概念只有一句简单的定义，要想比较深刻地理解有一定的困难，在学习过程中往往为了搞清一个问题，需要查找很多参考书，因此作者对学习过程中出现的问题比较熟悉，作者把这些问题基本上都写在书中了，并且讲述得比较详细，容易理解。这也是本书取名为《理解 Java 语言程序设计》的原因。

有的书籍是通过讲述实例的方法来介绍 Java 的，这样很快就能掌握 Java 的编程方法，缺点是系统性不太强，而且一些概念讲的不是很透彻。而有的书籍是通过讲述理论概念的方法来介绍 Java 的，系统性较强，但缺乏实践。本书吸取两者的长处，使用理论联系实际的方法。

本书不但讲述有关 Java 的基本概念，还介绍学习 Java 的方法，使读者学会分析问题、解决问题的方法。

希望读者在学习时把主要精力放在基本概念、基本规定和编程方法上。

考虑到篇幅，本书没有写很多的程序。本书的程序大部分都是说明性的，仅供参考。

本书没有讲 Java 的图形化用户界面，因为有一些更好的可替代的选择。如使用 Macromedia 的 Flex 编程系统的 Macromedia Flash，Flex 系统提供了非常强大的客户端编程环境。Flash Player 更小，下载速度可能更快，其“外观”可能更好，应用范围很广。还有一个选择是 Eclipse 的 Standard Widget Toolkit(SWT)。

感谢王正元、王新春的支持和鼓励，没有他们的支持和鼓励作者是不可能完成这本书的。

王少川

2010 年 5 月

目录

第 1 章　Java 概述 …… 1

1.1　Java 简介 …… 1
1.2　Java 运行环境 …… 2
1.2.1　Java API 类库 …… 2
1.2.2　Java 虚拟机 …… 2
1.2.3　编译器 …… 5
1.2.4　Java 程序的运行过程 …… 6
1.3　Java 开发工具集 …… 9
1.3.1　Java 开发环境 …… 9
1.3.2　环境变量 …… 11
1.4　application、applet 和 servlet …… 12
1.5　Java 平台 …… 13
1.6　面向对象 …… 14
1.7　类和对象的初步概念 …… 18
1.8　包 …… 21
1.9　EJB 简介 …… 22
1.9.1　EJB 的基本结构 …… 23
1.9.2　如何开发一个 enterprise bean …… 25
1.9.3　用 session bean 为工作流建模和提高性能 …… 25
1.10　中间件 …… 26
1.10.1　中间件简介 …… 27
1.10.2　CORBA …… 29
1.10.3　对象请求代理 …… 30
1.10.4　工作流 …… 31

第 2 章　Java 语言要素 …… 35

2.1　标志符 …… 35

2.2 各种符号值 …… 36
2.3 操作符 …… 36
2.4 分隔符 …… 37
2.5 保留词 …… 37
2.6 空白 …… 39
2.7 注释 …… 39
2.8 字符集 …… 41
2.9 类型 …… 41
2.10 变量 …… 43
2.11 常量 …… 46
2.12 修饰词 …… 46
2.13 表达式 …… 47
2.14 语句 …… 48

第 3 章 Java 的基本数据类型、操作符和表达式 …… 50

3.1 变量 …… 50
3.2 Java 的基本数据类型 …… 53
3.2.1 整数类型 …… 55
3.2.2 浮点数类型 …… 56
3.2.3 字符类型 …… 57
3.2.4 布尔数据类型 …… 61
3.2.5 正确选择数据类型 …… 62
3.3 对数据类型进行的操作 …… 66
3.3.1 后缀表达式 …… 67
3.3.2 一元操作符 …… 67
3.3.3 二元操作符 …… 68
3.3.4 移位操作符 …… 70
3.3.5 关系判断操作符 …… 71
3.3.6 相等操作符 …… 72
3.3.7 位操作符 …… 73
3.3.8 逻辑操作符 …… 75
3.3.9 条件操作符 …… 76
3.3.10 复合赋值操作 …… 77
3.3.11 增 1 操作符＋＋、减 1 操作符－－和副作用 …… 78
3.3.12 操作符的优先级 …… 80
3.4 表达式与赋值语句 …… 83
3.4.1 表达式 …… 83
3.4.2 赋值语句 …… 85
3.5 类型转换 …… 88

3.5.1 隐式类型转换 …… 88
3.5.2 显式类型转换 …… 90
3.5.3 利用基本数据类型的包装类中的方法实现类型转换 …… 91

第 4 章 控制结构 …… 92

4.1 if 选择结构 …… 93
4.2 if…else 选择结构 …… 94
4.3 switch 选择结构 …… 95
4.4 for 循环结构 …… 96
4.5 for each 循环结构 …… 98
4.6 while 循环结构 …… 98
4.7 do…while 循环结构 …… 99
4.8 break 和 continue 语句 …… 99
4.9 return 语句 …… 100

第 5 章 类 …… 101

5.1 类的声明 …… 102
5.2 类体 …… 104
5.3 字段 …… 106
5.3.1 实例变量和类变量的声明 …… 107
5.3.2 变量声明可能使用的修饰词 …… 107
5.3.3 字段的初始化 …… 108
5.3.4 变量的作用范围 …… 112
5.4 方法 …… 113
5.4.1 方法的定义 …… 115
5.4.2 把消息传递给方法(或构造器) …… 118
5.4.3 方法调用 …… 119
5.4.4 参数与参数的传递 …… 120
5.4.5 方法的重载 …… 122
5.4.6 方法的改写 …… 123
5.4.7 多态 …… 124
5.4.8 通过类方法掩盖 …… 126
5.4.9 main 方法 …… 126
5.4.10 final 方法 …… 127
5.4.11 abstract 方法 …… 127
5.4.12 static 方法 …… 127
5.4.13 Math 类方法 …… 128
5.5 构造器 …… 130
5.5.1 构造器的定义 …… 131

5.5.2 初始化 …… 133
5.5.3 构造器的调用 …… 136
5.5.4 类的引用(调用类的变量或方法) …… 137
5.5.5 类图 …… 138
5.6 Enum 类 …… 139
5.7 所有类的超类 Object 的方法 …… 142
5.8 包装类 …… 143
5.9 自动打包和自动拆包 …… 147
5.10 String 类 …… 148
5.11 涉及时间、日期数据类型的类 …… 152
5.12 抽象类 …… 153
5.12.1 抽象类的定义 …… 154
5.12.2 什么时候使用抽象类 …… 154

第 6 章 类与类之间的关系 …… 158

6.1 类的继承 …… 158
6.1.1 继承的实质是层次化 …… 160
6.1.2 上层类对象和下层类对象的关系 …… 163
6.1.3 里斯科夫代换原则 …… 166
6.1.4 下层类中使用构造器和终止方法 …… 170
6.1.5 下层类对象到上层类对象的转换 …… 170
6.1.6 null …… 171
6.1.7 this …… 171
6.1.8 super …… 172
6.1.9 继承数据成员和方法 …… 172
6.1.10 字段的掩盖和遮蔽 …… 173
6.1.11 方法的改写 …… 176
6.1.12 所有类的上层类 Object …… 178
6.2 合成 …… 179
6.2.1 合成简介 …… 179
6.2.2 合成/聚合优先于继承 …… 182
6.3 聚合 …… 183
6.4 依赖 …… 184
6.5 嵌套类 …… 185
6.5.1 静态嵌套类 …… 187
6.5.2 非静态嵌套类 …… 190
6.5.3 局部类 …… 194
6.5.4 匿名类 …… 196

第 7 章 对象 …… 200

7.1 链表、堆和栈 …… 200
7.1.1 链表 …… 200
7.1.2 堆和栈的概念 …… 201
7.1.3 内存 …… 201
7.1.4 基本数据变量和对象在内存中的表示 …… 203
7.2 创建对象 …… 204
7.3 对象的初始化 …… 205
7.4 对象变量 …… 208
7.5 instanceof、==和 equals 等的使用 …… 211
7.6 equels、hashCode 和 ToString 的改写 …… 214
7.7 实现 Comparable 接口 …… 215
7.8 对象的作用域规则 …… 215
7.9 对象的销毁 …… 216
7.10 强引用、软引用、弱引用和幻引用 …… 219
7.11 使用静态工厂方法创建对象 …… 220
7.12 Singleton 模式 …… 221
7.13 数组 …… 222
7.13.1 创建数组 …… 223
7.13.2 初始化数组 …… 225
7.13.3 字符数组 …… 227
7.13.4 多维数组 …… 227
7.13.5 访问数组中的每个元素使用表达式 …… 228
7.13.6 数组成员 …… 228
7.13.7 java.util.Arrays 的 static 方法 …… 229

第 8 章 接口 …… 230

8.1 接口简介 …… 230
8.1.1 什么是接口 …… 230
8.1.2 什么情况下使用接口 …… 231
8.2 接口的定义 …… 231
8.2.1 接口的成员 …… 232
8.2.2 接口中的字段(常量) …… 232
8.2.3 接口中的抽象方法 …… 233
8.2.4 多重接口 …… 233
8.3 实现接口 …… 234
8.3.1 一个接口实现多个接口 …… 234
8.3.2 接口的部分实现 …… 235

8.4 接口与抽象类的区别 …… 235
8.5 接口优于抽象类 …… 236
8.6 实现接口和继承类的区别 …… 237
8.7 通过接口引用对象 …… 237
8.8 注释类型 …… 237

第9章 泛型 …… 239

9.1 泛型的定义 …… 239
9.2 类型参数限定 …… 240
9.3 泛型方法 …… 241
9.4 类型擦除和原始类型 …… 242
9.5 桥接方法 …… 244
9.6 泛型的实例化 …… 245
9.7 用通配符作为类型实参 …… 247
9.8 对通配符的约束 …… 247
9.9 泛型类型的继承 …… 248
9.10 Class类、reflection(内省)类与泛型 …… 249

第10章 异常 …… 251

10.1 异常简介 …… 251
10.2 异常的分类 …… 252
10.2.1 Java对异常的分类 …… 252
10.2.2 Error …… 253
10.2.3 RuntimeException类及其子类 …… 254
10.2.4 Exception类中不是RuntimeException类的其他异常 …… 255
10.2.5 checked exception类和unchecked exception类 …… 255
10.3 为什么要使用异常 …… 256
10.4 使用throws子句声明方法可以抛出的受检查异常 …… 258
10.5 使用throw语句抛出异常 …… 259
10.6 异常处理子程序 …… 260
10.6.1 使用try语句块 …… 261
10.6.2 使用catch语句块 …… 261
10.6.3 使用finally语句块 …… 262
10.6.4 异常链接 …… 263
10.7 捕获异常后再次抛出异常 …… 264
10.8 Throwable类的方法 …… 264
10.9 使用异常处理子程序需要注意的问题 …… 265

第11章 包 …… 267

11.1 使用包的好处 …… 267

11.2 包的声明 …… 267
11.3 包的嵌套 …… 268
11.4 类型的导入 …… 268
11.5 静态导入 …… 269
11.6 Java 中常用的包 …… 272
11.7 设定 classpath …… 272
11.8 访问权限修饰词的使用 …… 273
11.9 java.lang.Package 类 …… 274

第 12 章 输入和输出 …… 275

12.1 概述 …… 275
12.1.1 数据的层次 …… 275
12.1.2 文件和流 …… 276
12.1.3 标准流 …… 281
12.2 java.io 包中输入、输出类的继承关系 …… 282
12.3 字节流 …… 284
12.3.1 InputStream 抽象类和 OutputStream 抽象类 …… 284
12.3.2 FileInputStream 类和 FileOutputStream 类 …… 287
12.3.3 FilterInputStream 类和 FilterOutputStream 类 …… 289
12.3.4 BufferedInputStream 类和 BufferedOutputStream 类 …… 291
12.3.5 DataInput 接口和 DataOutput 接口 …… 293
12.3.6 DataInputStream 类和 DataOutputStream 类 …… 294
12.3.7 PrintStream 类 …… 295
12.3.8 PipedInputStream 类和 PipedOutputStream 类 …… 296
12.3.9 Serializable 接口和 Externalizable 接口 …… 297
12.4 Reader 抽象类和 Writer 抽象类 …… 300
12.4.1 java.io.Reader 类 …… 301
12.4.2 java.io.Writer 类 …… 304
12.4.3 字符流与字节流的对照关系 …… 307
12.5 文件操作 …… 309
12.6 文件访问 …… 310
12.6.1 FileReader 类 …… 311
12.6.2 FileWriter 类 …… 311
12.6.3 RandomAccessFile 类 …… 312
12.6.4 访问文件的方式 …… 313
12.7 java.io 的分类 …… 319
12.7.1 java.io 主要类的分类 …… 319
12.7.2 处理数据流的类 …… 320
12.8 New I/O …… 322
12.8.1 Channels …… 322

12.8.2 FileChannel 类 …… 325
12.8.3 Buffers …… 326
12.8.4 缓冲区的位置、限制、容量和标记 …… 327
12.8.5 ByteBuffer …… 329
12.8.6 内存映射文件 …… 332
12.8.7 创建缓冲区 …… 334
12.8.8 缓冲区数据传送 …… 339
12.8.9 文件锁定 …… 341
12.9 Scanner 类 …… 342
12.10 文件的输入、输出 …… 344
12.10.1 写入文件 …… 344
12.10.2 读文件 …… 347
12.11 格式化输出 …… 351
12.12 简易输入 …… 354
12.12.1 规则(正则)表达式 …… 354
12.12.2 java.util.regex.Pattern 类 …… 354
12.12.3 规则(正则)表达式的规则 …… 355
12.12.4 java.util.regex.Matcher 类 …… 357
12.12.5 使用规则(正则)表达式进行匹配的步骤 …… 357
12.12.6 替换操作 …… 360

第 13 章 内省(reflection) …… 361

13.1 Class 类 …… 361
13.2 类型描述符 …… 363
13.3 获得 Class 对象的方法 …… 364
13.4 取得类的有关类型信息 …… 364
13.5 内省和泛型 …… 366

第 14 章 算法、数据结构和 Collection …… 368

14.1 算法和数据结构 …… 368
14.1.1 算法分析 …… 368
14.1.2 数据结构的分类 …… 371
14.1.3 线性数据结构 …… 371
14.1.4 分层数据结构 …… 376
14.1.5 集合 …… 380
14.1.6 映射 …… 380
14.2 对象集(Collection) …… 380
14.2.1 对象集框架 …… 381
14.2.2 使用对象集框架的好处 …… 383
14.2.3 Collection 接口 …… 384

14.2.4 Iterator<>接口 …… 385
14.2.5 Set 接口、SortedSet 接口和 NavigableSet 接口 …… 387
14.2.6 List 接口 …… 389
14.2.7 Queue<E>接口、Deque<E>接口 …… 390
14.2.8 Map(K,V)接口、SortedMap<K,V>接口和 NavigableMap 接口 …… 391
14.3 实现接口的类 …… 393
14.3.1 抽象类 …… 394
14.3.2 实现接口的类 …… 395
14.3.3 JDK 1.1 中的遗留对象集类型 …… 398
14.4 通过 Comparable 和 Comparator 排序 …… 401
14.5 Collections 类和算法 …… 401
14.6 同步包装 …… 403
14.7 不可修改的包装 …… 403
14.8 实现 …… 404
14.8.1 Set 接口 …… 405
14.8.2 List 接口 …… 405
14.8.3 Map 接口 …… 406
14.8.4 ArrayList 类 …… 406
14.8.5 LinkedList 类 …… 407
14.8.6 HashMap 类 …… 408

第 15 章 多线程 …… 409

15.1 线程的概念 …… 409
15.2 实现多线程的方法 …… 410
15.2.1 创建 Thread 的下层类 …… 411
15.2.2 实现 Runnable 接口 …… 412
15.2.3 线程池 …… 413
15.3 线程调度、线程的优先级和线程的状态 …… 415
15.3.1 线程调度 …… 415
15.3.2 线程的优先级 …… 415
15.3.3 线程的状态和线程的生命周期 …… 416
15.3.4 守护线程和用户线程 …… 418
15.4 同步 …… 418
15.4.1 使用 synchronized 来表示与对象的互锁关系 …… 419
15.4.2 锁对象 …… 424
15.4.3 生产者/消费者 …… 427
15.5 使用高效的映射、集合和队列 …… 430
15.6 同步器 …… 431
15.7 死锁 …… 433

15.8 优化 …… 434

第 16 章 网络 …… 436

16.1 socket …… 436
16.2 java.net 包中的类 …… 439
16.3 ServerSocket 类的应用 …… 442
16.3.1 创建 ServerSocket …… 442
16.3.2 ServerSocket 类的主要方法 …… 442
16.3.3 ServerSocket 类的应用 …… 443
16.4 Socket 类的应用 …… 448
16.4.1 创建 Socket …… 449
16.4.2 Socket 类的主要方法 …… 449
16.4.3 Socket 类的应用 …… 451
16.5 与 Socket 类相关的类 …… 454
16.6 Socket 套接字的异常处理 …… 454
16.7 DatagramPacket 类的应用 …… 455
16.7.1 创建 DatagramPacket …… 455
16.7.2 DatagramPacket 类的应用 …… 456
16.8 DatagramSocket 类的应用 …… 458
16.8.1 创建 DatagramSocket …… 458
16.8.2 接收(读取)UDP 包 …… 459
16.8.3 发送 UDP 包 …… 460
16.8.4 构建 UDP 客户端/服务器 …… 461
16.9 URL 的应用 …… 463
16.9.1 创建 URL …… 463
16.9.2 URL 和 URLConnection 的主要方法 …… 464
16.9.3 URL 的应用 …… 465
16.10 分布式系统 …… 466
16.10.1 Java RMI …… 468
16.10.2 CORBA …… 474
16.10.3 编写 CORBA 服务者和客户代码 …… 478
16.10.4 DCOM …… 480
16.10.5 CORBA、COM/DCOM 和 Java 三种技术的性能比较 …… 481
16.10.6 计算机网络模式的发展展望——跨越 Web …… 482

第1章

Java概述

1.1 Java 简介

Java 编程语言(Java Programming Language)的特点是什么?

简单说来特点就是:

- 简单、小巧(没有 C++的不适当特性,如指针运算。支持能够在小型机器上独立运行的软件。基本的解释器和类支持仅约 40KB,加上基础的标准类库约增加 175KB)。
- 面向对象(object-oriented)。
- 安全(内存管理机制)。
- 网络技能(Network-savvy)。
- 健壮。
- 体系结构中立(跨平台)。
- 可移植性。
- 解释性(现在使用即时编译器将字节码翻译成机器码)。
- 高性能(由于使用即时编译器,性能甚至超过传统编译器)。
- 多线程(可以有更好的交互响应和实时行为)。
- 动态性(库中可以自由地增加新方法和实例变量,而对客户端却没有影响。所以能够适应软件的不断发展)。

Java 不仅是一种编程语言,还是一个开发平台。

Java 的缺点是运行速度较慢和占用内存较多。目前 JVM 解释代码所花费的时间已大大减少,这要归功于即时(Just in time)编译程序的推出。预先(AOT ahead of time)编译程序能够预先解释 Java 代码,把 Java 代码转换成经过优化且面向具体平台的二进制代码,从而使运行速度大大加快(有报道说可把代码的执行速度提高 20 倍)。Java 1.4 版本通过共享"资源"和模糊中间层次的界限等技术使运行速度加快,比 Java 1.3.1 版本快了 60%。现在,Java 虚拟机使用了即时编译器,其代码的运行速度与 C++相差无几,甚至"一个较慢的 Java 程序与几年前相等快的 C++程序相比还要快一些"。据报道,完全使用 Java 编写的程序和使用 C 语言编写的相同程序的运行速度一样快。

另外,消耗内存多的垃圾回收程序和异常处理程序的改进也使运行时占用的内存减少。编写好的程序也可以加快运行速度和减少内存的占用。

1.2 Java 运行环境

Java 环境又可以分为运行环境和开发环境。

Java 运行环境就是 Java 程序运行时所需要的各种条件(包括硬件和软件方面的条件)。Java 开发环境是指开发 Java 程序时所需要的各种条件。

Java 平台由两个部分组成：一个是用于实现软件的假想计算机，即 Java 虚拟机(Java Vertual Machine,JVM)；另外一个是软件组件的集合，叫做 Java 应用程序编程接口(Java Application Programming Interface,Java API)。

Java 运行环境(Java Runtime Environment,JRE)是运行 Java 程序的用户使用的软件。

一般来说，Java 程序不能直接在计算机上执行，而是在一个称为 Java 6.0 平台的标准环境中运行。在各种计算机和操作系统上，这个标准环境就像一个普通的软件一样运行。

运行环境主要由 Java API、Java 虚拟机和编译器三个部分组成。

Java 运行环境是 JDK 的子集，包括 JVM、运行时的类库及执行字节码所需要的 Java 应用程序启动器(但是省略了编译器 Javac 这样的开发工具)。

1.2.1 Java API 类库

Java API 也叫 Java 类库，是 Java 预定义的一些类、接口和方法等代码组成的包。Java API 提供了丰富的类和方法的集合，它们提供了程序员所需要的许多功能模块，因此使得程序员的工作更加简单。Java API 方法以 Java 开发人员包(JDK)的形式提供。也可以说，Java 程序就是通过把程序员编写的新方法和新的类分别与 Java API“预定义包(pre-packaged)”中可用的方法和可用的类组合而形成的。

我们把自己编写的 API 称为导出的 API(exported API)，是指自己编写的一些可被外面访问的类、接口、构造方法、成员和序列化形式，用户和程序员通过它们可以访问一个类、接口或包。这里，类的成员(member)包括类的域(fied)、方法(method)、成员类(member class)和成员接口(member interface)。类、接口、构造方法、成员和序列化形式都叫做 API 的元素。任何用户都可以使用这些元素，而 API 的创建者负责维护这些元素。简单地讲，一个包的导出 API 就是包中所有的可被外面访问的公有(public)类、接口中公有的或受保护的(protected)成员和构造方法。

学习 Java 编程语言的难点是 Java API，要能够熟练运用 Java API。

1.2.2 Java 虚拟机

Java 与大多数编程语言不同的是，Java 编译器不为实际的处理器生成代码，而是为称为 Java 虚拟机的假想计算机生成称为字节码的代码。Java 虚拟机是 Java 系统的基石。正是因为 Java 虚拟机，Java 才与平台和操作系统无关，才能实现编写一次，到处运行。

如果说操作系统屏蔽了计算机硬件的差别，一个同样的操作系统可以安装在不同硬件组成的计算机上；那么，Java 虚拟机则可以屏蔽操作系统的 Java 虚拟机差别，使 Java 程序

可以在不同的操作系统上运行。

把一个 Java 的运行环境安装在一台计算机上，其实就是 Java 虚拟机。即开发环境中除了第一阶段编辑程序和第二阶段编译程序外，从类装载开始后面的各个阶段构成运行环境。

Java 虚拟机是计算机体系结构的一个软件实现。所有的 Java 程序都是在 Java 虚拟机上运行的。Java 虚拟机是以 Java 字节码为指令的软 CPU，Java 的许多功能都是依靠 Java 虚拟机来实现的。例如，在客户端，用户通过 Internet 下载 Web 服务器上的 applet，再依靠本地 Java 虚拟机对. class 文件进行解释和执行，然后在浏览器上显示出来。另外，Java 虚拟机是使. class 文件具有跨平台能力的核心结构。不管用户的计算机使用的是什么操作系统，只要安装了 Java 虚拟机，那么 Java 程序就能够在该计算机上运行。

JVM 在被编译的 Java 程序和底层的硬件平台与操作系统之间提供一个抽象的层。Java 虚拟机管理着内存、任务和其他资源。

1. JVM 的硬件组成

从硬件方面来讲，JVM 由 4 个部分组成：寄存器组(registers)、堆栈(stack)、垃圾回收堆(gabage-collecting heap)和方法区(method area)。

1) 寄存器组

JVM 中共有 4 个寄存器：pc(Java 程序计数器，包括下一字节码指令的地址)、optop(指向操作数栈顶端的指针)、frame(指向当前执行方法的执行环境的指针)和 vars(指向当前执行方法局部变量区第一个变量的指针，它们同微处理器的寄存器很相似)。每一个寄存器都存放一个 32 位的地址，分别指向存储空间的不同部分。由于方法区存放的是字节代码，因此它是以字节(8 位)为边界的，而在栈和堆中则是以字(32 位)为边界的。

2) 堆栈

Java 虚拟机是栈式的，它不定义或者使用寄存器来传递或接受参数，其目的就是为了保证指令集的简洁性和实现时的高效性(特别是对寄存器不多的处理器)。JVM 栈中存放的是 Java 栈帧(Java Stack Frame)。一个栈帧包括三个部分：局部变量、执行环境和操作数栈，分别由 vars、frame 和 optop 寄存器指示。它记录了 Java 程序当前方法的执行状态。

3) 垃圾回收堆

JVM 的垃圾回收堆是用来存放对象的。每次在 Java 程序中用 new 分配一个新的对象，JVM 就在垃圾回收堆中划出一块来存放该对象。当该对象不再被引用时，JVM 的垃圾回收堆将自动清除这一存储空间。

4) 方法区

JVM 中的方法区存放 Java 程序中方法的字节代码，pc 寄存器总是指向方法区中的某个位置，它表示下一条即将执行的字节代码，用于控制方法的执行顺序。

2. JVM 组件

一个给定的 JVM 可能是作为整块程序来实现的，不过却是按一组组件来设计的。Java 虚拟机的组件包括类装载器(class loader)、字节码解释器(bytecode interpreter)、安全管理