目录

[第二章 Java程序设计环境 2](#_Toc455319047)

[2.1 安装Java开发工具箱 2](#_Toc455319048)

[1、相关术语 2](#_Toc455319049)

[2、Java三类应用程序 2](#_Toc455319050)

[3、JDK、JRE、JVM区别与联系 2](#_Toc455319051)

[4、JDK安装配置步骤 3](#_Toc455319052)

[5、JDK目录结构 3](#_Toc455319053)

[2.2 选择开发环境 4](#_Toc455319054)

[2.3 使用命令行工具 4](#_Toc455319055)

[1、Java源代码编译及运行流程 4](#_Toc455319056)

[2.4 使用集成开发环境 4](#_Toc455319057)

[1、创建Java工程向导中JRE选项 4](#_Toc455319058)

[2.5 运行图形化应用程序 5](#_Toc455319059)

[2.6 建立并运行applet 6](#_Toc455319060)

[第三章 Java的基本程序设计结构 6](#_Toc455319061)

[3.1 一个简单的Java应用程序 6](#_Toc455319062)

[1、Java特性 6](#_Toc455319063)

[2、Java程序基本格式 6](#_Toc455319064)

[3.2 注释 7](#_Toc455319065)

[1、注释语法 7](#_Toc455319066)

[3.3 数据类型 7](#_Toc455319067)

[3.4 变量 7](#_Toc455319068)

[3.5 运算符 7](#_Toc455319069)

[3.6 字符串 8](#_Toc455319070)

[3.7 输入输出 8](#_Toc455319071)

[3.8 控制流程 8](#_Toc455319072)

[3.9 大数值 8](#_Toc455319073)

[3.10 数组 8](#_Toc455319074)

# 第二章 Java程序设计环境

## 2.1 安装Java开发工具箱

### 1、相关术语

Java Development Kit (JDK)——Java开发工具包

Java Runtime Environment (JRE)——Java运行环境

Standard Edition (SE)——标准版，用于桌面等标准应用程序开发

Enterprise Edition (EE)——企业版，用于企业级的复杂服务器应用程序开发

Micro Edition (ME)——微型版，用于手机和其他小型设备应用程序开发

Java 2 (J2)——过时，特指2006年之前的Java版本，J2EE、J2SE、J2ME也是当时的提法

Update (u)——Java的小幅度更新版

### 2、Java三类应用程序

（1）控制台程序

（2）图形化应用程序

（3）applet应用程序

### 3、JDK、JRE、JVM区别与联系

<http://playkid.blog.163.com/blog/static/56287260201372113842153/>

JVM, Java虚拟机，Java跨平台的核心部分，在JVM上运行的是Java编译后的目标代码（字节码）。

JRE, Java运行环境，JRE=JVM+类库，可直接运行.class文件。

JDK, Java编译环境，JDK=JVM+类库+编译工具，可编译Java源代码成.class文件并运行。



### 4、JDK安装配置步骤

（1）下载并安装JDK

（2）添加环境变量

（3）安装JDK标准库源文件

标准库源文件在JDK中以src.zip的形式发布，需要将其解压缩后使用。

cd jdk1.8.0\_92 #进入jdk目录

mkdir src #创建子目录src

cd src #进入子目录src

jar xvf ..\src.zip #解压缩src.zip至子目录src中

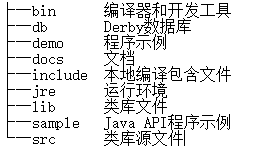
（4）安装JDK文档

JDK文档独立于JDK发布，需自行下载解压缩到jdk目录下。

cd jdk1.8.0\_92 #进入jdk目录

jar xvf jdk-8u92-docs-all.zip #解压缩jdk-8u92-docs-all.zip

### 5、JDK目录结构



/docs目录包含HTML格式的类库说明文档，用于学习参考。

/src目录包含Java类库公共部分的源代码，用于深入学习。

## 2.2 选择开发环境

集成开发环境IDE=编辑器+编译器+运行调试优化工具

NetBeans

Eclipse

## 2.3 使用命令行工具

### 1、Java源代码编译及运行流程

（1）javac程序将.java源文件编译成.class类文件

（2）java程序启动JVM，JVM执行.class文件中的字节码

javac xx.java #编译

java xx #运行

## 2.4 使用集成开发环境

### 1、创建Java工程向导中JRE选项



官方文档：

<http://help.eclipse.org/neon/index.jsp?topic=%2Forg.eclipse.jdt.doc.user%2Freference%2Fref-wizard-java-project.htm>

对于Eclipse而言，JRE可以安装多个，JDK有一个较新版本足够。

最简单的情况JDK与JRE版本相对。

在开发旧版本Java应用的场合，需要调低compiler compliance level。该选项应该与JRE版本相对应，指定了编译class时JDK的版本，保证高版本JDK编译出在指定JRE版本上运行的class文件，即编译过程中不使用高于compiler compliance level的语言特性。

JDK版本应不低于compiler compliance level。

例如，本机安装有JDK 1.8、JRE 1.8、JRE 1.7。若开发具有JDK 1.8特性的Java应用，compiler compliance level应设置为1.8，即用JDK 1.8编译在JRE 1.8上运行的Java应用。若开发在JRE 1.7上运行的Java应用，compiler compliance level应设置为1.7，即编译过程中屏蔽JDK 1.7以后的语言特性，用JDK 1.8编译在JRE 1.7上运行的Java应用。这就实现了一个JDK与多个JRE的对应问题。

**（1）使用一个执行环境的JRE**

指定构建用的JRE版本，同时compiler compliance自动更改为对应的编译级别。复选框给出了eclipse支持的所有运行环境选项，但只是选项而已，还是需要安装相应版本JRE。

Eclipse此选项试图实现兼容JRE开发，即在本机没有匹配JRE的情况下，借助本机中与目标JRE兼容的JRE开发调试应用程序。

此选项并不常用，JRE安装并不复杂且系统花销不大，不会有人甘冒兼容性风险而不去装对应版本的JRE。

**（2）使用项目指定的JRE**

指定构建用的JRE版本，同时compiler compliance自动更改为对应的编译级别。复选框给出了本机已安装的JRE列表，故JRE需提前安装。

JRE与JDK版本不同的场合使用此选项。

**（3）使用默认的JRE**

JRE可以安装多个，但当前默认的只能有一个。默认JRE一般为Eclipse运行的JRE环境，也可以自行改变。

JRE与JDK版本对应的场合使用此选项。

## 2.5 运行图形化应用程序

## 2.6 建立并运行applet

appletviewer程序，用于快速地测试applet，传入HTML文件。

javac xx.java #编译

appletviewer xx.html #运行

# 第三章 Java的基本程序设计结构

## 3.1 一个简单的Java应用程序

### 1、Java特性

（1）大小写敏感

（2）Java应用程序中的全部内容都必须放在类中，类可以理解为一个加载程序逻辑的容器。

（3）标准命名规范：

类名以大写字母开头，每个单词的首字母都大写，不能以数字开头，不能使用保留字，无长度限制，正确命名格式如，FirstSample

文件名与公共类名称应完全一致

### 2、Java程序基本格式

public class ClassName

{

public static void main(String[] args)

{

program statements;

}

}

（1）标准Java格式

一个Java源程序中有唯一的public class，唯一的public main，且public main在public class内，public class名称与文件名相同。

（2）main方法

一个Java源程序中有且只有一个public main方法，作为程序的入口。

main方法必须是静态的，且没有返回值。main方法没有给操作系统返回值，但Java程序会给系统返回值。如果main方法正常退出，返回值为0；调用System.exit()方法可以返回其他值。在Java中，main方法不再担负给操作系统返回值这一任务。

（3）主类

包含有public main方法的类称为主类，主类的概念是main方法的衍生。

一个Java源程序有且仅有一个main方法，因此，有且仅有一个主类，Java源程序名应与主类名相同。使用命令行“java xx”运行Java程序时，实际上是“java 主类名”，再在主类中找到public main方法，执行程序。因此，“java xx.class”是错误的。

（4）公共类

主类和公共类没有必然联系，主类不一定是public class，也可以是default class。一个Java源程序中公共类可以没有，最多有一个。

一般，主类就是是公共类。

## 3.2 注释

### 1、注释语法

（1）//注释单行

（2）/\*

注释多行，但不能嵌套

\*/

（3）/\*\*

\*注释自动生成文档

\*/

## 3.3 数据类型

## 3.4 变量

## 3.5 运算符

## 3.6 字符串

## 3.7 输入输出

## 3.8 控制流程

## 3.9 大数值

## 3.10 数组