面向对象 程序设计语言 Java 过程蓝图

计算机科学2000Vol.27№.7

88-91

Java 过程蓝图

Java Process Blueprint

刘建宾

(汕头大学工学院计算机系 汕头515063)

Abstract Java process blueprint is an oriented Java programming language, diagrammatized representation for program process logic. The engineering approach supports abstract representation of program at logic design and physical implementation two levels, which is a simple and practical, well-structured programming tool with high understandability. A formal model of Java process blueprint, graphical notations of abstract logic structure diagram, and a fundamental programming procedure are presented in the paper.

Keywords Java process blueprint, Abstract logic structure diagrams (ALSD), Java program representation, Java programming tool

1 引言

计算机应用正进入网络时代,Java 是一种广泛使用的网络编程语言,被称之为网络上的"世界语".Java 作为一种程序设计语言不仅具有简单、面向对象、分布式、解释执行、鲁棒、安全、平台无关、可移植、高性能、多线程以及动态性等特点,更重要的是它支持以网络为中心的新型计算模式——Java 计算模式,从而使 Java 技术成为当今世界上重要信息技术之一,给整个信息技术带来许多变化和影响,不仅在 Internet 软件开发中起重要作用,而且会在新世纪的软件开发技术中起重要作用。Java 应用程序的需求与日剧增,研究提高 Java 程序质量、可维护性及开发效率的方法与工具有重要意义。

Java 是一种面向对象的程序设计语言。一个 Java 程序是由一些类组成,类是由域和方法组成。程序执行时,由类生成一个个对象。对象的行为用方法来描述。方法基于过程,是对象中一个重要的构成要件。对方法的过程性编程方面,现有的工具大多把程序看成是字符的集合,主要提供文本编辑器予以支持.设计者需用 Java 语言直接写过程代码。这样容易使设计者过早陷入繁杂的实现细节,影响设计者有效地考虑解决问题的逻辑算法,导致编程效率低,容易出错,程序质量难以保证等问题。软件工程的研究表明,详细设计中把逻辑算法设计和具体实现细节分开有利于更有效地写程序。这当中需要解决的一个关键问题在于如何以更为有效和容易使用的方式来表现程序。设计表现形式与方法是设计人员表达思想,支持设计过程和实现语言语法,呈现设计结果的基本工具,也是研制 CASE 工具的基础,它对程序开发的效率和质量、程序员正确编程思想和风格的培养、程序的可维护性、可理解性,以及对工程化组织程序的生产都有着重要的影响和作用。

本文针对 Java 语言的特点提出一种工程化图表形式相结合的程序表现技术——Java 过程蓝图,它是在已有的图形化程序表示法基础上,进一步强调简单性、可理解性、逻辑设计独立性而设计出来的一种新的图表表现技术,用以支持 Java 对象中的方法设计。

2 形式化模型

为精确地描述 Java 过程蓝图,我们在这里给出一个形式化表示模型,该模型确立过程蓝图的表现形式、图形表示成分的绘制方法以及应该满足的有关条件。

2.1 程序的控制流和数据流

程序的控制流和数据流是构成程序处理逻辑的两个组成部分,是各种程序表示法都必须给出表示的主要内容.控制流是软件为完成处理或计算所执行的数据操作的次序,即数据操作的序列。控制流分为模块内的控制流和模块间的控制流。模块间的控制流即对象方法的调用和返回。模块内的控制包括顺序、循环、选

刘建宾 副教授,硕士生导师,主要从事软件方法与工具,软件工程,管理信息系统的研究工作。

择、跳转、例外处理等控制构造。

数据流表示对象数据域中的值怎样被求值、改变和移动，即对象状态值的序列。数据流是执行数据类型操作、数据存储操作以及数据流语句的结果。数据类型操作包括算术、关系及逻辑操作，还有字符串、位串等的处理操作。数据存储操作是对栈、队、表、数组、数据库、文件、持久对象等的操作。数据流语句包括明确赋值语句，I/O 操作以及条件测试等。

2.2 基本集合和函数

我们首先定义如下的集合和函数，为 Java 过程蓝图的定义建立基础。

(1)逻辑结点类型集合：T_n={SEQ，SYN，IFT，IFE，SWH，CAS，DFT，WHL，FOR，DOW，BLK，DCL，OPE，LAB，TRY，CAH，FNY，BRK，CON，THR，RET，UND}。T_{in}各元素的含义见表1。

(2)自然语言语句集合：S_{nn}={s|s∈自然语言语句}；

(3)受限自然语言语句集合：S_{df}={s|s∈S_{nn}∧verb(s)∈VDD∧noun(s)∈EDD∧conj(s)∈CDD}，其中VDD，EDD，CDD 分别表示命令动词词典、程序实体名词典和连接词词典；verb：S_{df}→VDD；noun：S_{df}→EDD；conj：S_{df}→CDD。

(4)Java 程序语言的基本操作语句和表达式集合：S_{pl}={s|s∈Java 语言表达式∨s∈JAVABCS}。其中，JAVABCS 为 Java 语言基本操作语句集，包括表达式语句、空语句、方法调用语句，不包括标号语句、语句块、synchronized 语句、三种选择语句(if，ifelse，switch)、三种循环语句(while，for，dowhile)、三种受限跳转语句(break，continue，return)，例外抛出语句(throw)，例外处理语句(try)，局部变量声明语句，注释语句。

2.3 Java 过程蓝图的定义

一个 Java 过程蓝图程序 jpb 是如下定义的三元组：

$$jpb=(t_1=(A_1=A_{11}\cup A_{12},dl),C_{df},f_{sc})$$

其中 t_1表示抽象逻辑结构图 ALSD，A_1表示逻辑结点的集合，C_{df}是用目标程序语言表示的操作表达式表，$C_{df}\subset S_{pl}$，f_{sc}：$A_{12}\to C_{df}$是带数据流的逻辑结点到目标语言表示的映射函数，下面我们给出 t_1的定义：

2.4 抽象逻辑结构图 ALSD

用 $t_1=(A_1=A_{11}\cup A_{12},dl)$表示 ALSD，其中 $A_{11}\subset(T_{in}-\{SEQ,BLK,DFT,LAB,TRY,FNY,BRK,CON,UND\})\times S_{df}$是带有数据流的逻辑结点集；$A_{12}\subset\{SEQ,BLK,DFT,LAB,TRY,FNY,BRK,CON,UND\}\times S_{nn}$是不带有数据流的逻辑结点集合；$dl:A_1\to 2^{A_1}$且满足如下定义的逻辑结点分解条件和 t_1树条件的逻辑结点分解函数。

表1 Java 过程蓝图处理逻辑结点类型

结构类型	结点表示	说 明
顺序	SEQ	概括结点，块
并发	SYN	Synchronized 语句
选择	IFT	If()…语句
	IFE	if()…else|elseif…语句
	SWH	Switch 多分支选择语句
	CAS	Case 情况分支
	DFT	default 否则情况分支
循环	WHL	while()…循环语句
	FOR	For()…循环语句
	DOW	do…while()循环语句
模块	BLK	过程块(段)
终结点	DCL	类型/类/函数声明语句
	OPE	基本操作语句
	LAB	标号结点
跳转	TRY	Try 例外处理语句
	CAH	Try 例外处理语句的 catch 拦截子句
	FNY	Try 例外处理语句的 finally 子句
	BRK	break 控制跳转语句
	CON	continue 控制跳转语句
	THR	Throw 例外抛出语句
	RET	return 语句
未定义	UND	未确定结点

·逻辑结点分解条件·对 $a\in A_1$，设 $k=|dl(a)|$，dl 满足下面的条件：

① 若 $a[T_{in}]\in\{BLK,DCL,OPE,LAB,BRK,CON,THR,RET,UND\}$，则有 $dl(a)=\Phi,k=0$；

② 若 $a[T_{in}]=$"IFE"，则有 $dl(a)\neq\Phi,k=2$，且对 $a_j\in dl(a)$，有 $a_j[T_{in}]\in T_{in}-\{CAS,DFT,CAH,FNY\},j=1,2$；

③ 若 $a[T_{in}]\in\{SEQ,SYN,IFT,CAS,DFT,WHL,FOR,DOW,CAH,FNY\}$，则有 $dl(a)\neq\Phi,k\geq 1$，且对 $a_j\in dl(a)$，有 $a_j[T_{in}]\in T_{in}-\{CAS,DFT,CAH,FNY\},j=1,\cdots,k$；

④ 若 $a[T_{in}]=$"SWH"，则有 $dl(a)\neq\Phi,k\geq 2$，且对 $a_j\in dl(a),j=1,2,\cdots,k$，有 $a_j[T_{in}]=$"CAS"，$j=1,2,\cdots,k-1$，$a_k[T_{in}]\in\{CAS,DFT\}$；

⑤ 若 $a[T_{in}]=$"TRY"，则有 $dl(a)\neq\Phi,k\geq 2$，且对 $a_j\in dl(a),j=1,2,\cdots,k$，有 $a_j[T_{in}]=T_{in}-\{CAS,DFT,CAH,FNY\}$，$a_j[T_{in}]=$"CAH"，$j=2,3,\cdots,k-1$，$a_k[T_{in}]\in\{CAH,FNY\}$。

·t_1树条件：给定(A_1,dl)，对每个 $a\in A_1$，用下面的方法产生结点分解图：用一个结点 n(a)表示 a，若 dl(a)=Φ，则该结点分解结束，否则对 $a_j\in dl(a)(j=1,$

2,…,k),用"L"形线依次把 n(a_i)连接到 n(a)的正下方,且保证各 n(a_i)处在同一垂直线上。若最终的分解图形成一棵树,则(A_1,dl)满足 t_1树条件。

2.5 操作表达式表 C_{dl}和映射函数 f_{sc}的构造

操作表达式表 C_{dl}给出逻辑算法的实现细节,映射函数 f_{sc}则在逻辑算法与实现细节之间建立起联系。我们可以从抽象逻辑结构图 t_1出发来构造 C_{dl}和 f_{sc}。

对抽象逻辑结构图 t_1上的每个带数据流的逻辑结点,即 A_{t1}中的每个结点 a($a\in A_{t1}$),进行如下处理:首先构造一个操作表达式表的表项 c,然后用 Java 语言写出结点 a 的数据流表现形式(表达式或基本操作语句)并填入 c 中,最后在结点 a 和表项 c 间建立联系,使 $f_{sc}(a)=c$。当 A_{t1}中的所有结点都构造完毕时,我们就得到与抽象逻辑结构图 t_1相对应的操作表达式表 C_{dl},以及 $A_{t1}\to C_{dl}$的映射函数 f_{sc},且 $C_{dl}=\{f_{sc}(a)\mid a\in A_{t1}\}$。

2.6 蓝图表现层次与方法

Java 过程蓝图以增量方式给出了处理逻辑在逻辑和实现两个层次上的表现形式。逻辑层的表示是开发者用汉语表达算法思想的工具,它是用"L"形连线将处理逻辑结点连接起来的一棵抽象树形图,用于描述抽象算法,它表达了一个处理的抽象控制结构和数据流的逻辑语义,并与具体实现细节无关。实现层的表示则是在保持逻辑层表示不变的情况下,通过操作表达式表进一步给出数据流的物理实现表示以及逻辑描述到物理实现表示的映射,从而提供了生成目标代码的必要细节。操作表达式表是逻辑处理数据流结点的实例化表示。一个逻辑层表示带有不同的操作表达式表可构成不同的实现层表示。

处理结点的描述包括结点类型,结点的逻辑语义及其物理实现表示,它表达了一个处理结点在逻辑层和物理实现层上的表示及逻辑语义到物理表示间的映射,处理结点按抽象程度的不同分为逻辑结点和物理结点。结点类型和语义说明构成了逻辑结点的表示。物理结点是对逻辑结点的实现,它是在逻辑结点的基础上进一步给出某种具体实现表示并完成逻辑描述到物理实现表示的映射。

在逻辑层和实现层上的表示分别用抽象逻辑结构图和操作表达式表来描述,描述的算法可称为逻辑算法和实现算法。蓝图的表示层次及方法如表2所示。

表2 Java 过程蓝图的表现层次与方法

表示层次	算法类别	表现形式	结点类别	表现内容	描述方式
逻辑层	逻辑算法	抽象逻辑结构图	逻辑结点	控制流	逻辑控制构造
				数据流	自然语言
实现层	实现算法	抽象逻辑结构图+操作表达式表	物理结点	控制流	逻辑控制构造
				数据流	Java 语言

3 抽象逻辑结构图的图形表示法

抽象逻辑结构图 ALSD 是 Java 过程蓝图中的图形成分,我们使用这种具有全部结构化控制构造的图形化表示来详细描述控制流,使用自然语言和受限自然语言描述逻辑结点的语义。抽象逻辑结构图是由逻辑结点构成的,从左到右、自上而下的线型树图,程序员使用它来进行逻辑算法设计,表示的算法与具体实现细节无关。抽象逻辑结构图是一种面向人和计算机的逻辑算法表示工具,它以一种较为自然直观的图形方式让人明白程序的工作原理,并以一种较为抽象的同构化表示方式让计算机获取算法的控制结构。

3.1 基本图形记法

基本图形记法见图1-13。

3.2 语法定义

用下列的 Backus 范式给出 ALSD 的图形语法定义:

〈ALSD〉::=〈正文〉|〈块〉|〈ALSD〉〈ALSD〉

〈块〉::=〈顺序〉|〈并发〉|〈选择〉|〈循环〉|〈例外处理〉|〈叶子〉

〈顺序〉::=〈L〉|〈块〉|〈顺序〉〈块〉

〈并发〉::=〈图2〉

〈选择〉::=〈图3〉|〈图4〉|〈图5〉

〈循环〉::=〈图8〉|〈图9〉|〈图10〉

〈例外处理〉::=〈图11〉

〈C 分支〉::=〈图6〉

〈D 分支〉::=〈图7〉

〈CD 分支〉::=〈C 分支〉〈D 分支〉

〈H 分支〉::=〈图12〉

〈Y 分支〉::=〈图13〉

〈HY 分支〉::=〈H 分支〉|〈Y 分支〉

〈叶子〉::=〈叶结点类型〉〈正文〉

〈叶结点类型〉::=BLK|DCL|OPE|LAB|BRK CON THR|RET|UND

〈TL〉::=〈块〉

〈T 分支〉::=〈块〉

〈F 分支〉::=〈块〉

〈正文〉::=〈自然语言〉|〈受限自然语言〉

4 基本程序设计过程

使用 Java 过程蓝图设计程序的基本过程是:

(1)使用抽象逻辑结构作图工具进行逻辑算法设计。抽象逻辑结构图不涉及语言实现细节，可使程序员将注意力集中于解决问题的算法上，使算法的设计更加有效。抽象逻辑结构图提供的程序块 BLK 结点和未定义 UND 结点可方便地支持自顶向下、自底向上、以及自顶向下和自底向上相结合的程序设计。设计中将逻辑 ALSD 中的未定义结点确定化，对未分解复合逻辑结点进行细化，每进行一步就得到一个新的进化逻辑 ALSD。当逻辑 ALSD 或进化 ALSD 满足形式化定义中的逻辑结点分解条件并且不含任何未定义结点时，我们就得到确定的逻辑程序，即确定逻辑 ALSD。

(2)将确定逻辑 ALSD 中带数据流的逻辑结点的数据流表示为 Java 程序语言形式，遵循 Java 语言的语法规则写出每个带数据流的逻辑结点的 Java 表达式或基本操作语句，构造并得到操作表达式表。操作表达式表具体给出程序与 Java 语言密切相关的实现细节。

(3)建立确定逻辑 ALSD 和操作表达式表的联系，在确定逻辑 ALSD 中每个带数据流的逻辑结点上标记操作表达式表中与其对应物理表示的表项号，得到具有实现形式映射关系的确定 ALSD。至此，我们已建立了一个完整的 Java 过程蓝图程序。

(4)生成程序源代码和实现 ALSD。从 Java 过程蓝图生成 Java 源程序是为了进一步编译并最终运行程序，而生成实现 ALSD 则是 Java 过程蓝图产生的一种附加文档，实现 ALSD 是将确定逻辑 ALSD 上的每个带数据流的逻辑结点语义替换为操作表达式表中对应 Java 语言实现表示后得到的树图。在 Java 过程蓝图基础上产生 Java 程序源代码和实现 ALSD 是非常机械的，使用代码生成工具和文档生成工具是完成此项工作的最好选择。

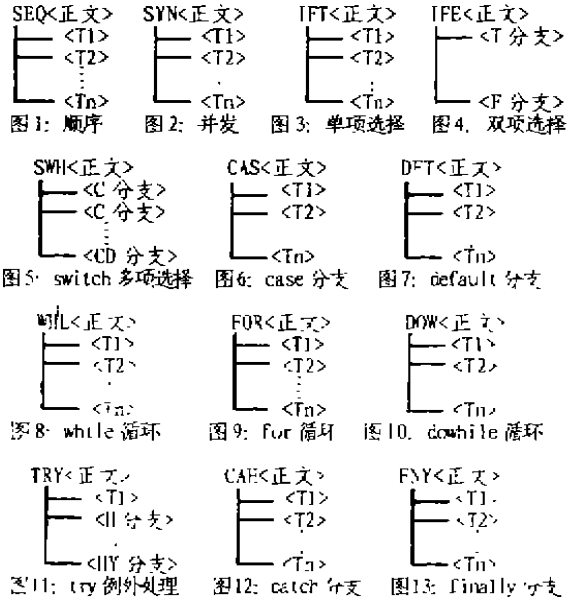

图1: 顺序　图2: 并发　图3: 单项选择　图4. 双项选择

图5: switch 多项选择　图6: case 分支　图7: default 分支

图8: while 循环　图9: for 循环　图10. double 循环

图11: try 例外处理　图12: catch 分支　图13: Finally 分支

参 考 文 献

1 刘建宾 一种工程化的程序表现技术 过程蓝图. 计算机工程与应用，1996，32(1)：31～34

2 Gosling J. et al. 著，蒋国新，等译. Java 语言规范. 北京 北京大学出版社，1997 12

3 Arnold K，Gosling J. 著，杨承高，等译 Java 程序设计语言. 北京 北京大学出版社，1998. 1

4 李明，李厚安 Java 程序设计教程. 北京 科学出版社，1997 8

5 Tripp L. L. A Survey of Graphical Notations for Program Design-An Update ACM SIGSOFT Software Engineering Notes，1988，13(1)

6 Chu Yaohan. Software Blueprint and Examples. Lexington D C Heath and Company，1982

(上接第49页)

一段距离，需要将很多细节落实到系统的设计中去。此外，在该模型中存在相当的数据冗余，在对模型的进一步探讨中可以考虑将重复的部分局限在控制信息和索引信息，而非空间对象信息的全体。另外考虑针对各种不同的应用环境，将该模型加以细化，使之更能适应各种具体系统的具体要求。

参 考 文 献

1 Newell D. Perspectives in GIS Database Architecture[C] SSD'1997

2 Aggarwal C C，et al The S-Tree，An Efficient Index for Multidimensional Objects[C] SSD'1997

3 Belussi A，et al. Manipulating Spatial Data in Constraint Databases[C] SSD'1997

4 詹奇波，张其善 电子地图数据库存贮文件的设计[J]. 计算机科学，1996，23(3)

5 Traynor C，Williams M G. Why Are Geographic Information Systems Hard to Use?[C]. CHI'95

6 Moore L，et al Network Access to Geographic Names Defense Mapping Agency Prototype to the Information Super Highway[C]. ACSM/ASPRS，1995