Java Semi-Formal

Specification Language

(JSF)

目录

1 JSF基本语法 3

1.1 JSF语法形式 3

1.2 基本表达式 3

1.2.1 量化布尔表达式 3

1.2.2其他关键字 3

1.2.3 逻辑运算符 4

1.2.4描述形式 5

1.2.5变量约束 5

2 JSF方法规格 5

2.1 REQUIRES ：前置条件 6

2.2 MODIFIES：副作用 6

2.3 EFFECTS：后置条件 7

3 多线程场景下的方法规格 7

4.1多线程共享前置规格 8

4.2 多线程共享后置规格 8

# 1 JSF基本语法

## 1.1 JSF语法形式

JSF规格的形式有如下两种：

**//@ <JSF>**

**//@ <JSF>**

**……**

**/\*@ <JSF>**

**@ <JSF>**

**……**

**@ <JSF> \*/**

在/\*@……\*/的注释中，@符号在规格的每一行的开头之前使用。

## 1.2 基本表达式

### 1.2.1 量化布尔表达式

使用关键字\*all*替代，以及\*exist*替代。用法如下：

* （\*all* T x; b）表示所有的T类型的x使得b成立。
* （\*all* T x; r; b）等价于（\*all* T x; r==>b），表示所有的r范围里的T类型的x，使得b成立。
* （\*exist* T x; b）表示存在T类型的x满足b。
* （\*exist* T x; r; b）等价于（\*exist* T x; r==>b），表示存在一个的r范围里的T类型的x，使得b成立。

**例1：**

**//@ REQUIRES:** **(\all int i, j;**

**//@   0 <= i & i < j & j < a.length;**

**//@   a[i] <= a[j]);**

**public int lookup(int elem) {…**

**表示所有在“0 <= i & i < j & j < a.length”范围里的、使得“a[i] <= a[j]”成立的整型i和j。**

### 1.2.2其他关键字

* （\sum T x; range a to b; exp）等价于(x)。
* \max，\min表达了最大值和最小值。

**例2：**

**//@ EFFECTS: \result == (\max int i; 0 <= i && i < arr.length; arr[i]);**

**表示后置条件中结果返回的是数组arr[]中的最大值。**

* \*result* M 表示方法M的返回值。
* \*old*(e) 与Object-Z中的e’同义。表示在方法被调用刚开始时表达式e的值。只能用于后置条件中。

**例3：**

**//@ EFFECTS: getR() == \old(getR()) + c;**

**表示方法getR()是进入本方法之前getR()的值加c。**

* 使用常规单词表达含义，而尽量减少缩写、和一些自定义的标识符。例如使用*length*表达数组长度，*size*表达数据结构的大小，*contain*(e).Q表示集合Q中包含元素e等等。

**例4：**

**//@ EFFECTS: size < limit && !contains(elem).Queue;**

**public void add(elem)**

**表示添加的元素elem在Queue中没有。**

### 1.2.3 逻辑运算符

* A==> B ：表示了A蕴含B（如果事件A发生必导致事件B发生，则称A蕴含了B）
* A<==B ：表示了A包含 B，B蕴含A
* A<==>B ：逻辑等价
* 逻辑与&&：为了与面向对象编程语言逻辑保持一致，不采用集合论中的&来表达逻辑与
* 逻辑或|| ：为了与面向对象编程语言逻辑保持一致，不采用集合论中的|来表达逻辑或
* 逻辑非! ：等价于
* 空集：{}

### 1.2.4描述形式

JSF中的规格描述可以使用Java程序中的常用方法来帮助表达，而不对语法进行限制。

**例6：**

**对于数组arr[]来说，描述其长度的语法是：arr.length；而对于字符串str来说，描述其长度的语法是：str.length()。当使用JSF描述时，只要识别到“length”，就表明是在描述长度，即允许规格中出现类似于“arr.length()”的写法。**

JSF在逻辑表达式中需要检查类型。比如C = A==>B中A和B都应该是相同的类型，C应该是布尔型。

### 1.2.5变量约束

JSF建议尽量少的定义在规格中使用的中间变量。

# 2 JSF方法规格

考虑到方法规格描述中可能会使用到类的属性、传入参数列表，故方法规格采用以下形式（不同于JML中将方法规格置于方法名之前）：

**<访问修饰符> <返回值类型> <方法名>（<参数列表>）{**

**/\* @ REQUIRES:**

**@ MODIFIES:**

**@ EFFECTS:**

**@ \*/**

**<方法体>**

**}**

**例6：**

**public boolean contains(int elem) {**

**/\* @ EFFECTS: \result == (\exists int i; 0 <= i && i < size; arr[i] == elem);**

**@ \*/**

**……}**

**public void remove(int elem) {**

**/\* @ REQUIRES: arr[] != {};**

**@** **EFFECTS: !contains(elem);**

**@ EFFECTS: (\forall int e;**

**@ e != elem;**

**@ contains(e) <==> \old(contains(e)));**

**@ EFFECTS: \old(contains(elem))**

**@ ==> size == \old(size) - 1;**

**@ EFFECTS: !\old(contains(elem)) ==> size == \old(size);**

**@ \*/**

**……}**

## 2.1 REQUIRES ：前置条件

前置条件，执行前过程对输入或系统状态的约束要求。方法规格的前置条件和后置条件实质上就是对输入进行划分。通过划分，简化了方法的计算逻辑——在前置条件范围内的，是方法必须要处理的输入，对应的处理结果必须满足相应的约束；而在这范围之外的可以不予考虑。

假定输入到方法中的参数都是合法的，无需开发人员在前置条件中加以赘述。前置条件必须是布尔表达式。

## 2.2 MODIFIES：副作用

表示一个方法在执行过程会对哪些用户能够感知到的数据进行修改。所谓“能够感知到的修改” 指的是函数内部与外部互动，产生运算以外的其他结果，包括：对this的修改、对传入对象的修改、对环境对象的修改（如数据库、外接设备等）、对全局变量的修改。

如果没有修改的值，则此项可写为“None”，即此方法为纯方法。纯方法就是没有副作用的方法，强调没有副作用，意味着函数要保持独立，所有功能就是返回一个新的值，没有其他行为，尤其是不得修改外部变量的值。

## 2.3 EFFECTS：后置条件

执行后返回结果或系统状态应该满足的要求，也就是过程在所有未被前置条件排出的输入下给出的结果。一种方法只有在遵守前提条件的情况下才能保留其后置条件。如果在不符合前提条件的情况下进行调用，则无法保证。

JSF对后置条件细化，针对不同的情况分为*normal\_behavior*和*exceptional\_behavior*(E) P。*normal\_behavior*表示方法正常终止情况下的后置条件，也可以省略不写。JSF还允许在异常情况下使用*exceptional\_behavior*，来表示如果方法因为E类型异常出现的终止，相当于java语言中的try-catch语句，出现异常时需要执行P。E是Throwable的子类。

在后置条件描述中，每一句都应该是布尔表达式，且应该体现出用集合论和一阶逻辑谓词表达设计层面的逻辑，也可以采用自然语言描述规格。

# 3 多线程场景下的方法规格

多线程场景下，任何一个类都可能被多个线程共享。因此，除了功能视角的规格之外，还需要从线程共享控制视角来明确相应的规格。

共享对象方法的规格说明形式如下：

**<访问修饰符> [synchronized] <返回值类型> <方法名>（<参数列表>）{**

**/\* @ REQUIRES:**

**@ MODIFIES:**

**@ EFFECTS:**

**@ THREAD\_REQUIRES:**

**@ THREAD\_EFFECTS:**

**@ \*/**

**<方法体>**

**}**

**其中，**

## 4.1多线程共享前置规格

THREAD\_REQUIRES**描述是否要求线程主动对该方法的调用进行锁控制，使用**关键字\*locked*(…)表示。如果不写，则表明对使用者不做任何多线程同步控制要求。对于线程安全设计而言，不应对使用者提出此要求。\locked(…)中的参数可以是\this或者方法传入的对象变量**，表示要求调用者对括号中的共享对象进行锁控制。**

## 4.2 多线程共享后置规格

THREAD\_EFFECTS专门用来描述一个方法否对多线程并发执行进行了控制，使用关键词\locked(x)。如果不写，则表明该方法不对多线程并发执行进行任何控制。如果括号后面没有任何变量，则表明对整个方法进行同步控制；如果给定了变量，则表明该方法仅对相应变量的操作访问进行了同步控制。注意，\locked(x)中的x只能是方法输入参数中的对象，或者所处类的成员对象。对于线程安全设计而言，这部分一定要写。