Intermediate Java Language

Our goal is to motivate each addition to the language with a compelling example that shows why a new language feature has been included in the language, what benefits it provides for the programmer, and what are the pitfalls of using the new feature.

Some concepts may be expressed easily in one programming language, but may require a lot more work in another language. We want to highlight such situations, so that you understand what one may expect in a language, and how to get around the language shortcommings.

While all of what we have said above seems to revolve around *language features*, our focus is on the fundamental concepts in program design, making sure that your thinking about program design transcends the language and focuses on understanding the connection between information, its representation as data, and the program that manipulates the data.

Beginner ProfessorJ

The *Beginner ProfessorJ* language is specified by the following data definitions. The ellipsis (...) means that there can be zero or more data items of the given kind. You can think about it as a list of zero or more items.

Program = *Import ... Def ...*

Import = import *Name* ;
import *Name.** ;

Def = class *Id* { *Member Member ...* }
class *Id* implements *Id* { *Member Member ...* }
interface *Id* { *Signature ...* }

Signature = *Type Id* (*Type Id* , ...);

Member = *Field* ;
Constructor ;
Method ;

Field = *Type Id* = *Expression* ;
Type Id ;

Constructor = *Id* (*Type Id* ,...) { *Init ...* }

Method = *Type Id* (*Type Id* ,...) { *Statement* }

Init = this.*Id* = *Id* ;

Statement = if (*Expression*) { *Statement* } else { *Statement* } ;
return *Expression* ;

Expression = - *Expression*
! *Expression*
this
Expression.*Id*
Expression.*Id* (*Expression* ,...)
new *Id* (*Expression* ,...)
check *Expression* expect *Expression*
check *Expression* expect *Expression* within *Expression*
(*Expression* ,...)
Id
Number
ICharacter
String
true
false

Name = *Id*. ...
Id

Op = one of: `+ - * / < <= == > >= && ||`

Type = *Id*
int
char
double
float
long
byte
short

Id is a sequence of letters, digits, _, and $ and it must start with either a letter or with _

String = " any sequence of characters, including none "

Additions to the Beginner ProfessorJ Language

We have added the following to the *Beginner ProfessorJ* language:

- Visibility modifiers `public` and `private` can be used with constructors, and method definitions.
- We allow overloading of constructors and methods. This means that in one class we can define several constructors as long as their argument lists are different (differing in the types of arguments, not the argument names). Similarly, we can define in one class several methods with the same name as long as the argument lists are different.
- A class can `extend` a super class.
- A class can implement more than one `interface`, though we have not seen an example of this yet.
- A class can be declared `abstract`. We expect that `abstract` class will contain at least one `abstract` method.
- We allow `abstract` method declarations in `abstract` classes. Such method declarations have only purpose statements and headers, just like method declarations in `interface`s.
- We modify the definition of **Statement** as follows:

  Statement = if (*Expression*) { *Statement* } else { *Statement* } ;
  if (*Expression*) *Statement* else *Statement* ;
  return *Expression* ;

  It means, when the statements that follow `if` or `else` are short and simple, you do not need to enclose them in braces. However, every `if` statement must be followed by an `else` clause.
- The **Expression** can no longer have the format

  Expression = check *Expression* expect *Expression*
  check *Expression* expect *Expression* within *Expression*

  Instead, you can use any of the `check...` methods in the *tester* library.