3. The imperative core $\text{Java}_I$ of Java

In this chapter we define the basic model $\text{execJava}_I$, which defines the semantics of the sequential imperative core of Java with statements (appearing in method bodies) and expressions (appearing in statements) over the primitive types of Java. Each machine is started with an arbitrary but fixed Java program it has to interpret. In Sect. 3.1 we describe the signature of $\text{Java}_I$ and the static semantics of the input programs. We explain the form in which these programs are supposed to appear to the ASM, namely as annotated abstract syntax trees, resulting from parsing and elaboration. For future use in the proofs we also list the constraints which are imposed on the syntax of programs and on the types of the constructs appearing in them. In Sect. 3.2 we define the ASM rules for the dynamic semantics of $\text{Java}_I$ programs.

3.1 Static semantics of $\text{Java}_I$

The primitive types of $\text{Java}_I$ are: boolean, byte, short, int, long, float, double, char. Types are denoted by capital letters $A$, $B$, $C$. The types byte, short, int and long are called integral types. The types float and double are called floating point types. Numeric types are integral types or floating point types. External representations of values of primitive type are called literals. Table 3.1 contains examples of literals.

A binary relation $\leq$ is defined between primitive types. In terms of the JLS, the relation $A \leq B$ means that there exists an identity conversion or a widening primitive conversion from $A$ to $B$. In traditional terms, the relation $A \leq B$ means that $A$ is a subtype of $B$, i.e., each value of type $A$ can be used as a value of type $B$. In some cases, however, information may be lost. For example, if a 64-bit value of type long is converted to a 32-bit value of type float, then some precision may be lost.

Definition 3.1.1. The relation $\leq$ is the least relation on the set of primitive types which is reflexive, transitive and has the following properties:

$$\text{byte} \leq \text{short} \leq \text{int} \leq \text{long} \leq \text{float} \leq \text{double}, \quad \text{char} \leq \text{int}.$$  

Reflexive means that $A \leq A$ for each primitive type $A$. Transitive means that, if $A \leq B$ and $B \leq C$, then $A \leq C$.  

R. F. Stärk et al., *Java and the Java Virtual Machine*  
© Springer-Verlag Berlin Heidelberg 2001
Table 3.1 The primitive types of Java

<table>
<thead>
<tr>
<th>Type</th>
<th>Size</th>
<th>Default</th>
<th>Literals</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>false</td>
<td>true, false</td>
<td></td>
</tr>
<tr>
<td>byte</td>
<td>8 bit</td>
<td>(byte)0</td>
<td></td>
</tr>
<tr>
<td>short</td>
<td>16 bit</td>
<td>(short)0</td>
<td></td>
</tr>
<tr>
<td>int</td>
<td>32 bit</td>
<td>0</td>
<td>11, 1969, 0xff00, 017</td>
</tr>
<tr>
<td>long</td>
<td>64 bit</td>
<td>0L</td>
<td>11L, 0x1000L, 0777L</td>
</tr>
<tr>
<td>float</td>
<td>32 bit</td>
<td>0.0f</td>
<td>3.141f, 1.2e+23f</td>
</tr>
<tr>
<td>double</td>
<td>64 bit</td>
<td>0.0d</td>
<td>3.141, 1e-9, 0.1e10</td>
</tr>
<tr>
<td>char</td>
<td>16 bit</td>
<td>'\u0000'</td>
<td>'a', '?' , '\n', '\uFFFF'</td>
</tr>
</tbody>
</table>

3.1.1 Syntax of Java

The syntax of the imperative core of Java is defined in Fig. 3.1. It can also be viewed as defining corresponding domains (also called universes) of Java. Although in our ASMs we will extend some of these domains by a small number of auxiliary constructs which do not appear in the syntax of Java, we use the names of Java constructs also as names for the corresponding extended ASM universes. Usually we denote domains by words beginning with a capital letter and write dom for elements of Dom, i.e., assuming without further mentioning that dom ∈ Dom.

Fig. 3.1 uses universes which represent basic syntactic constructs of Java, namely:

- Exp ...... expressions,
- Lit ...... literals,
- Asgn .... assignments,
- Loc ...... local variables,
- Stm ...... statements,
- Uop ...... unary operators,
- Block .... blocks,
- Bop ...... binary operators,
- Bstm .... block statements,
- Lab ...... labels.

Local variables and labels are identifiers, sequences of letters and digits starting with a letter. Java programs are written in the Unicode 16-bit character set and so letters and digits may be drawn from the entire Unicode character set.

The unary operators Uop are listed in Table 3.2 and the binary operators Bop in Table 3.3. The function ‘max’ in the column ‘Result type’ of the two tables denotes the maximum of types with respect to the subtype relation ⊑. Although the set of primitive types is not linearly ordered by the relation ⊑, the maximum always exists for the special cases in Table 3.2 and Table 3.3. The type cast operator ‘(B)’ is considered as a unary operator in Table 3.2 provided that B is a primitive type. Binary operators are associated to the left: e₁ bop e₂ bop e₃ is read as (e₁ bop e₂) bop e₃.