3

A Quick Tour of Java

Earlier, we introduced key object-oriented concepts such as objects, methods and classes and how these may be ultimately used in a computer model. In this chapter, we see how the Java programming language is used to construct our object model of the problem domain. This approach is advantageous in that it allows our model to operate or “come alive” under computer control.

3.1 Primitive Types

The Java programming language allows the creation of objects that will ultimately participate in message communication. We have seen that objects may have diverse behavior and that it is more convenient to specify objects via classification, that is, class constructs.

Before examining class definitions for user-specified objects, we should be mindful that Java also provides primitive values from which other (bigger) objects may be described in terms of and constructed from. For example, a complex number may be seen as being comprised of two numbers representing the real and imaginary parts.

The primitive types byte, short, int and long defined in the Java language allow for the representation of discrete integer values of widths 8, 16, 32, and 64 bits, respectively. These in turn correspond to the representation of numeric ranges –128 to 127, –32768 to 32767, –2147483648 to 2147483647, and –9223372036854775808 to 9223372036854775807, respectively.

The primitive types float and double allow for the representation of single and double precision floating-point real values with representational widths of 32 and 64 bits, respectively. The adopted IEEE 754 standard includes both positive and negative sign-magnitude numbers, both positive and negative zeros and infinities, and unique not-a-number representations.
Values of type `float` are of the form $s \cdot m \cdot 2^e$, where $s$ is either $+1$ or $-1$, $m$ is a positive integer less than $2^{24}$, and $e$ is an integer between $-149$ and $104$. Similarly, values of type `double` have the similar form $s \cdot m \cdot 2^e$, but $m$ is a positive integer less than $2^{53}$, and $e$ is an integer between $-1075$ and $970$.

Finally, the primitive types `char` and `boolean` allow for 16-bit multi-byte characters and `false/true` boolean values, respectively.

### 3.2 Object Definition

Building upon the primitive values supported by the language proper, other entities to be manipulated are user-designed objects which are defined via class constructs. A class construct in Java consists of the `class` keyword followed by the class name and braces `{ }` which delimit the declaration of attributes and methods for its instances. The Counter class introduced in Chapter 2 would have the following form in Java:

```java
class Counter {
    attribute and method declarations
}
```

Object attributes are, in turn, either nested component objects or primitive types used to represent the object. An *instance method* manipulates the object by altering its attribute values. The `number` attribute and `add()` method in the Counter class below are representative of an object’s *state* and *operation*, respectively:

```java
class Counter {
    int number;
    void add() {
        number = number +1;
    }
}
```

The `number` attribute is also known as an *instance variable* because it occurs in every object or instance of the Counter class. This further implies that an attribute in one instance is independent from that in another instance. In the same vein, a method manipulates object attributes in the same instance. This occurs when a method is invoked and the corresponding code in its body is executed. In our recent example, invoking the `add()` method of an object will increment the corresponding `number` attribute.

### 3.2.1 Variable Definitions

Variable definitions in Java take the form below, where the type name $T$ precedes the variable name $v$:

```
T v;
```