Before we begin our bottom-up exploration of Java syntax, let's take a moment for a top-down overview of a Java program. Java programs consist of one or more files, or *compilation units*, of Java source code. Near the end of the chapter, we describe the structure of a Java file and explain how to compile and run a Java program. Each compilation unit begins with an optional `package` declaration followed by zero or more `import` declarations. These declarations specify the namespace within which the compilation unit will define names, and the namespaces from which the compilation unit imports names. We'll see `package` and `import` again later in this chapter in "Packages and the Java Namespace" on page 94.

The optional `package` and `import` declarations are followed by zero or more reference type definitions. We will meet the full variety of possible reference types in Chapters 3 and 4, but for now, we should note that these are most often either `class` or `interface` definitions.

Within the definition of a reference type, we will encounter *members* such as *fields*, *methods*, and *constructors*. Methods are the most important kind of member. Methods are blocks of Java code composed of *statements*.

With these basic terms defined, let's start by approaching a Java program from the bottom up by examining the basic units of syntax—often referred to as *lexical tokens*.

Excerpts from Flanagan, D. (2019). *Java in a Nutshell*. Sebastopol, CA. O'Reilly.

Java programs are written using Unicode. You can use Unicode characters anywhere in a Java program, including comments and identifiers such as variable names. Unlike the 7-bit ASCII character set, which is useful only for English, and the 8-bit ISO Latin-1 character set, which is useful only for major Western European languages, the Unicode character set can represent virtually every written language in common use on the planet.

If you do not use a Unicode-enabled text editor, or if you do not want to force other programmers who view or edit your code to use a Unicode-enabled editor, you can embed Unicode characters into your Java programs using the special Unicode escape sequence `\u`*xxxx*—that is, a backslash and a lowercase u, followed by four hexadecimal characters. For example, `\u0020` is the space character, and `\u03c0` is the character π.

Java is a case-sensitive language. Its keywords are written in lowercase and must always be used that way. That is, `While` and `WHILE` are not the same as the `while` keyword. Similarly, if you declare a variable named `i` in your program, you may not refer to it as `I`.

Java ignores spaces, tabs, newlines, and other whitespace, except when it appears within quoted characters and string literals. Programmers typically use whitespace to format and indent their code for easy readability, and you will see common indentation conventions in this book's code examples.

59

Comments are natural-language text intended for human readers of a program. They are ignored by the Java compiler. Java supports three types of comments. The first type is a single-line comment, which begins with the characters // and continues until the end of the current line. For example:

```
int i = 0;   // Initialize the loop variable
```

The second kind of comment is a multiline comment. It begins with the characters /* and continues, over any number of lines, until the characters */. Any text between the /* and the */ is ignored by javac. Although this style of comment is typically used for multiline comments, it can also be used for single-line comments. This type of comment cannot be nested (i.e., one /* */ comment cannot appear within another). When writing multiline comments, programmers often use extra * characters to make the comments stand out. Here is a typical multiline comment:

```
/*
 * First, establish a connection to the server.
 * If the connection attempt fails, quit right away.
 */
```

The third type of comment is a special case of the second. If a comment begins with /**, it is regarded as a special *doc comment*. Like regular multiline comments, doc comments end with */ and cannot be nested. When you write a Java class you expect other programmers to use, provide doc comments to embed documentation about the class and each of its methods directly into the source code. A program named javadoc extracts these comments and processes them to create online documentation for your class. A doc comment can contain HTML tags and can use additional syntax understood by javadoc. For example:

```
/**
 * Upload a file to a web server.
 *
 * @param file The file to upload.
 * @return <tt>true</tt> on success,
 *         <tt>false</tt> on failure.
 * @author David Flanagan
 */
```

Excerpts from Flanagan, D. (2019). *Java in a Nutshell*. Sebastopol, CA. O'Reilly.

60

The following words are reserved in Java (they are part of the syntax of the language and may not be used to name variables, classes, and so forth):

```
abstract   const      final      int        public     throw
assert     continue   finally    interface  return     throws
boolean    default    float      long       short      transient
break      do         for        native     static     true
byte       double     goto       new        strictfp   try
case       else       if         null       super      void
catch      enum       implements package    switch     volatile
char       extends    import     private    synchronized while
class      false      instanceof protected  this
```

An *identifier* is simply a name given to some part of a Java program, such as a class, a method within a class, or a variable declared within a method. Identifiers may be of any length and may contain letters and digits drawn from the entire Unicode character set. An identifier may not begin with a digit.

In general, identifiers may not contain punctuation characters. Exceptions include the dollar sign ($) as well as other Unicode currency symbols such as £ and ¥.

Literals are sequences of source characters that directly represent constant values that appear as-is in Java source code. They include integer and floating-point numbers, single characters within single quotes, strings of characters within double quotes, and the reserved words true, false, and null. For example, the following are all literals:

```
1    1.0    '1'    1L    "one"    true    false    null
```

Java also uses a number of punctuation characters as tokens. The Java Language Specification divides these characters (somewhat arbitrarily) into two categories, separators and operators. The 12 separators are:

```
(   )   {   }   [   ]

...  @  ::

;   ,   .
```

The operators are:

```
+   -   *    /    %    &   |    ^    <<   >>   >>>
+=   -=   *=   /=   %=   &=  |=   ^=   <<=  >>=  >>>=
=    ==   !=   <    <=   >    >=
!    ~    &&   ||   ++   --   ?    :    ->
```