Random Fact 9.2

Programming Languages

Many hundreds of programming languages exist today, which is actually quite surprising. The idea behind a high-level programming language is to provide a medium for programming that is independent from the instruction set of a particular processor, so that one can move programs from one computer to another without rewriting them. Moving a program from one programming language to another is a difficult process, however, and it is rarely done. Thus, it seems that there would be little use for so many programming languages.

Unlike human languages, programming languages are created with specific purposes. Some programming languages make it particularly easy to express tasks from a particular problem domain. Some languages specialize in database processing; others in "artificial intelligence" programs that try to infer new facts from a given base of knowledge; others in multimedia programming. The Pascal language was purposefully kept simple because it was designed as a teaching language. The C language was developed to be translated efficiently into fast machine code, with a minimum of housekeeping overhead. The C++ language builds on C by adding features for object-oriented programming. The Java language was designed for securely deploying programs across the Internet.

The initial version of the C language was designed around 1972. As different compiler writers added incompatible features, the language sprouted various dialects. Some programming instructions were understood by one compiler but rejected by another. Such divergence is an immense pain to a programmer who wants to move code from one computer to another, and an effort got underway to iron out the differences and come up with a standard version of C. The design process ended in 1989 with the completion of the ANSI (American National Standards Institute) Standard. In the meantime, Bjarne Stroustrup of AT&T added features of the language Simula (an object-oriented language designed for carrying out simulations) to C. The resulting language was called C++. From 1985 on, C++ grew by the addition of many features, and a standardization process was completed in 1998. C++ has been enormously popular because programmers can take their existing C code and move it to C++ with only minimal changes. In order to keep compatibility with existing code, every innovation in C++ had to work around the existing language constructs, yielding a language that is powerful but somewhat cumbersome to use.

In 1995, Java was designed by James Gosling to be conceptually simpler and more internally consistent than C++, while retaining the syntax that is familiar to millions of C and C++

James Gosling, Designer of the Java Programming Language

programmers. The Java language was an immediate success, not only because it eliminated many of the cumbersome aspects of C++, but also because it included a powerful library.

However, programming language evolution has not come to an end. There are aspects of programming that Java does not handle well. Computers with multiple processors are becoming increasingly common, and it is difficult to write *concurrent* Java programs that use multiple processors correctly and efficiently. Also, as you have seen in this chapter, it can be rather cumbersome in Java to deal with small blocks of code such as the code for measuring an object. In Java, you have to make an interface with a method for that code, and then provide a class that implements the interface. In *functional* programming languages, you can manipulate functions in the same way as objects, and such tasks becomes much easier. For example, in Scala, a hybrid functional/object-oriented language, you can simply construct a `DataSet` with a function, without having to use an interface:

```
data = new DataSet((x : Rectangle) => x.getWidth() * x.getHeight())
```

Could the Java language be enhanced for concurrent and functional programming? Some attempts have been made, but they were not encouraging because they interacted with existing language features in complex ways. Some new languages (such as Scala) run on the same virtual machine as Java and can easily call existing Java code. Nobody can tell which languages will be most successful in the years to come, but most software developers should expect to work with multiple programming languages during their career.