**Using the PrintWriter Class**

There are several different approaches to writing text files in Java, but the simplest approach is to use the **PrintWriter** class, which contains methods with familiar names like **print()** and **println()**. As you probably suspect, these methods stream output to a file instead of the monitor. As you work through this lesson, think about all of the information you write on a daily basis. Can you list any that are analogous to writing files to the hard drive?

Based on your experience reading files, you probably suspect that information can be printed to a file as tokens using a loop. The **for** loop is often used to write files, although a **while** loop works just as well. When the last token is printed to the file, an EOF marker is created so the file can be re-read at another time.

It is good programming practice while you are writing code to test frequently, in order to avoid an accumulation of errors. When you write to text files, it is hard to tell if the program is executing properly because nothing shows up on the screen. It is a good strategy to use temporary print statements to display the information to the screen. This tells you that the program is actually doing something, even if it is not what you intended. Once your program works, you can comment out the statement that prints to the screen, or delete it.

### Part 1

Reading and writing files requires great attention to detail, so Java requires extra coding to ensure output streams are correctly handled. The code may appear somewhat unclear at first, but as long as you follow the templates provided, there won't be any problems. The details of file handling will become less obscure as you progress farther into the course.

Before moving on, be sure that you have tried all the suggested modifications and can answer the questions posed in the Virtual Lecture Notes.

When a file is opened, always remember to close it. This is especially important when writing files. The physical act of writing the file to disk occurs when the **close()** method is called.

### Part 2

The programs you are writing during this introductory phase of the course are still fairly simple, and you are developing the foundation skills necessary for more advanced topics coming soon. An often overlooked skill is the ability to read and understand code; it is also a critical debugging technique.

Some events or phenomena are too costly, too dangerous, or too complicated to study or observe directly, so they must be simulated. The Monte Carlo Method involves repeated random simulations to approximate quantities that cannot be quantified by direct observation or experimentation. This method is applied in numerous areas such as computer chip design, search and rescue operations, containment of oil spills, evaluation of investment opportunities, product marketing, etc.

Let's say a company holds a contest and says you have a 1 in 6 chance of winning a prize. Can you simulate the contest and test their claim? Sure! By using a standard six-sided die, you can pick one number to be the "winner." Let's pick the number 1. Each time a 1 is rolled, you win. Since you have a 1 in 6 chance of rolling a 1, the average number of throws before landing on a 1 be approximately six. The Monte Carlo Method can be used to simulate situations like this.