Special Topic 11.3

Command Line Arguments

Depending on the operating system and Java development system used, there are different methods of starting a program—for example, by selecting "Run" in the compilation environment, by clicking on an icon, or by typing the name of the program at a prompt in a terminal or shell window. The latter method is called "invoking the program from the command line". When you use this method, you must type the name of the program, but you can also type in additional information that the program can use. These additional strings are called *command line arguments*.

For example, it is convenient to specify the input and output file names for the `LineNumberer` program on the command line:

```
java LineNumberer input.txt numbered.txt
```

When you launch a program from the command line, you can specify arguments after the program name. The program can access these strings by processing the `args` parameter of the `main` method.

The strings that are typed after the Java program name are placed into the `args` parameter of the `main` method. (Now you finally know the use of the `args` parameter that you have seen in so many programs!)

For example, with the given program invocation, the `args` parameter of the `LineNumberer.main` method has the following contents:

- `args[0]` is `"input.txt"`
- `args[1]` is `"numbered.txt"`

The `main` method can then process these parameters, for example:

```
if (args.length >= 1)
   inputFileName = args[0];
```

It is entirely up to the program what to do with the command line argument strings. It is customary to interpret strings starting with a hyphen (-) as program options. For example, we may want to enhance the `LineNumberer` program so that a `-c` option places line numbers inside comment delimiters; for example

```
java LineNumberer -c HelloWorld.java HelloWorld.txt
```

If the `-c` option is missing, the delimiters should not be included. Here is how the `main` method can analyze the command line arguments:

```
for (String arg : args)
{
   if (arg.startsWith("-")) // It's an option
   {
      if (arg.equals("-c")) useCommentDelimiters = true;
```

```
      }
      else if (inputFileName == null) inputFileName = arg;
      else if (outputFileName == null) outputFileName = arg;
   }
```

Should you support command line interfaces for your programs, or should you instead supply a graphical user interface with file chooser dialog boxes? For a casual and infrequent user, the graphical user interface is much better. The user interface guides the user along and makes it possible to navigate the application without much knowledge. But for a frequent user, graphical user interfaces have a major drawback—they are hard to automate. If you need to process hundreds of files every day, you could spend all your time typing file names into file chooser dialog boxes. But it is not difficult to call a program multiple times automatically with different command line arguments. Productivity Hint 7.3 discusses how to use shell scripts (also called batch files) for this purpose.