**Command-Line Arguments**

A Java application can accept any number of arguments from the command line. This allows the user to specify configuration information when the application is launched.

The user enters command-line arguments when invoking the application and specifies them after the name of the class to be run. For example, suppose a Java application called Sort sorts lines in a file. To sort the data in a file named friends.txt, a user would enter:

java Sort friends.txt

When an application is launched, the runtime system passes the command-line arguments to the application's main method via an array of Strings. In the previous example, the command-line arguments passed to the Sort application in an array that contains a single String: "friends.txt".

**Echoing Command-Line Arguments**

The [Echo](http://docs.oracle.com/javase/tutorial/essential/environment/examples/Echo.java) example displays each of its command-line arguments on a line by itself:

public class Echo {

 public static void main (String[] args) {

 for (String s: args) {

 System.out.println(s);

 }

 }

}

The following example shows how a user might run Echo. User input is in italics.

*java Echo Drink Hot Java*

Drink

Hot

Java

Note that the application displays each word — Drink, Hot, and Java — on a line by itself. This is because the space character separates command-line arguments. To have Drink, Hot, and Java interpreted as a single argument, the user would join them by enclosing them within quotation marks.

*java Echo "Drink Hot Java"*

Drink Hot Java

**Parsing Numeric Command-Line Arguments**

If an application needs to support a numeric command-line argument, it must convert a String argument that represents a number, such as "34", to a numeric value. Here is a code snippet that converts a command-line argument to an int:

int firstArg;

if (args.length > 0) {

 try {

 firstArg = Integer.parseInt(args[0]);

 } catch (NumberFormatException e) {

 System.err.println("Argument" + args[0] + " must be an integer.");

 System.exit(1);

 }

}

parseInt throws a NumberFormatException if the format of args[0] isn't valid. All of the Number classes — Integer, Float, Double, and so on — have parseXXX methods that convert a String representing a number to an object of their type.