Processing Command-Line Options in Java Programs

Java passes command-line arguments to programs in the array that you name in the `main()` declaration. The following declaration uses `args` for that array:

```
public static void main (String[] args)
```

A `Getopt` class for parsing arguments in Java is available here:

http://www.urbanophile.com/arenn/coding/download.html

Install the *jar* file somewhere and make sure that it is named in the value of your `CLASSPATH` environment variable. Then you can use `Getopt` as shown in the following example program:

```
// Cmdline.java - demonstrate command-line option parsing in Java

import java.io.*;
import java.sql.*;
import gnu.getopt.*;  // need this for the Getopt class

public class Cmdline
{
  public static void main (String[] args)
  {
    Connection conn = null;
    String url = null;
    String hostName = null;
    String password = null;
    String userName = null;
    LongOpt[] longOpt = new LongOpt[3];
    int c;

    longOpt[0] =
      new LongOpt ("host", LongOpt.REQUIRED_ARGUMENT, null, 'h');
    longOpt[1] =
      new LongOpt ("password", LongOpt.REQUIRED_ARGUMENT, null, 'p');
    longOpt[2] =
      new LongOpt ("user", LongOpt.REQUIRED_ARGUMENT, null, 'u');

    // instantiate option-processing object, and then
    // loop until there are no more options
    Getopt g = new Getopt ("Cmdline", args, "h:p:u:", longOpt);
    while ((c = g.getopt ()) != -1)
    {
      switch (c)
      {
      case 'h':
        hostName = g.getOptarg ();
        break;
      case 'p':
        password = g.getOptarg ();
        break;
      case 'u':
        userName = g.getOptarg ();
        break;
      case ':':       // a required argument is missing
      case '?':       // some other error occurred
        // no error message needed; getopt() prints its own
        System.exit (1);
      }
    }

    try
    {
```

```
        // construct URL, noting whether hostName was
        // given; if not, MySQL will assume localhost
        if (hostName == null)
          hostName = "";
        url = "jdbc:mysql://" + hostName + "/cookbook";
        Class.forName ("com.mysql.jdbc.Driver").newInstance ();
        conn = DriverManager.getConnection (url, userName, password);
        System.out.println ("Connected");
      }
      catch (Exception e)
      {
        System.err.println ("Cannot connect to server");
      }
      finally
      {
        if (conn != null)
        {
          try
          {
            conn.close ();
            System.out.println ("Disconnected");
          }
          catch (Exception e) { }
        }
      }
    }
  }
```

As the example program demonstrates, you prepare to parse arguments by instantiating a new `Getopt` object to which you pass the program's arguments and information describing the options the program allows. Then you call `getopt()` in a loop until it returns `-1` to indicate that no more options are present. Each time through the loop, `getopt()` returns a value indicating which option it's seen, and `getOptarg()` may be called to obtain the option's argument, if necessary. (`getOptarg()` returns `null` if no following argument was provided.)

When you create an instance of the `Getopt()` class, pass it either three or four arguments:

- The program name; this is used for error messages.
- The argument array named in your `main()` declaration.
- A string listing the short option letters (without leading dashes). Any of these may be followed by a colon (`:`) to indicate that the option requires a following argument, or by a double colon (`::`) to indicate that a following argument is optional.
- An optional array that contains long option information. To specify long options, you must set up an array of `LongOpt` objects. Each of these describes a single option, using four parameters:
  - The option name as a string (without leading dashes).
  - A value indicating whether the option takes a following argument. This value may be `LongOpt.NO_ARGUMENT`, `LongOpt.REQUIRED_ARGUMENT`, or `LongOpt.OPTIONAL_ARGUMENT`.
  - A `StringBuffer` object or `null`. `getopt()` determines how to use this value based on the fourth parameter of the `LongOpt` object.
  - A value to be used when the option is encountered. This value becomes the return value of `getopt()` if the `StringBuffer` object named in the third parameter is `null`. If the buffer is non-`null`, `getopt()` returns zero after placing a string representation of the fourth parameter into the buffer.

The example program uses `null` as the `StringBuffer` parameter for each long option object and the corresponding short option letter as the fourth parameter. This is an easy way to cause `getopt()` to return the short option letter for both the short and long options, so that you can handle them with the same `case` statement.

After `getopt()` returns `-1` to indicate that no more options were found in the argument array, `getOptind()` returns the index of the first argument following the last option. The following code fragment shows one way to access the remaining arguments:

```
for (int i = g.getOptind (); i < args.length; i++)
  System.out.println (args[i]);
```

The `Getopt` class offers other option-processing behavior in addition to what I've described here. Read the documentation included with the class for more information.