Table of Contents

Preface

General

This module is part of a collection of modules designed to make object-oriented programming concepts accessible to blind students.

Blind students should not be excluded from computer programming courses because of inaccessible textbooks. Because of its text-based nature, computer programming is fundamentally an accessible technology. However, many textbooks adopt and use high-level integrated development environments with graphical user interfaces that greatly reduce that accessibility.

The modules in this collection present object-oriented programming concepts in a format that blind students can read using tools such as an audio screen reader and an electronic line-by-line Braille display.

In an effort to get and keep the student's interest, the more advanced modules in this collection also make heavy use of programming projects that provide sensory feedback through the use of sound.

The collection is intended to supplement but not to replace the textbook in an introductory course in high school or college object-oriented programming.

This module explains how to get started programming in Java in a format that is designed to be accessible to blind students.

Prerequisites

In addition to an Internet connection and a browser, you will need the following tools *(as a minimum)* to work through the exercises in these modules:

- An audio screen reader that is compatible with your operating system, such as the NonVisual Desktop Access program (NVDA), which is freely available at http://www.nvda-project.org/.
- A refreshable Braille display capable of providing line by line tactile output of information displayed on the computer monitor is recommended. Such a display is described at http://www.userite.com/ecampus/lesson1/tools.php, is recommended.
- The Oracle Java Development Kit (JDK) *(See http://www.oracle.com/technetwork/java/javase/downloads/index.html)*
- Documentation for the Oracle Java Development Kit (JDK) *(See http://download.oracle.com/javase/8/docs/api/)*
- A simple IDE or text editor for use in writing Java code.

The minimum prerequisites for understanding the material in these modules include:

- A cursory understanding of algebra.
- An understanding of the material covered in the *Programming Fundamentals* modules that you will find in two formats at the following URLs. These modules provide fundamental programming concepts using the Java programming language in a format that should be accessible.
    - http://cnx.org/content/m45179/latest/?collection=col11441/latest
    - http://cnx.org/contents/fb64661c-5b3f-4ea8-97c6-e48df112438a

Viewing tip

I recommend that you open another copy of this document in a separate browser window and use the following links to easily find and view the listings while you are reading about them.

Listings

Discussion

Considering that during the past fifteen years, I have published several hundred online programming tutorials *(see http://www.dickbaldwin.com/toc.htm),* one might wonder why I am taking the time and expending the effort to publish still another online programming tutorial.

Accessibility is the keyword

When writing and publishing the earlier tutorials, I made no attempt to make them accessible to blind students. Some of them are probably accessible, simply because I used a relatively simple HTML format and didn't include any inaccessible content such as images. However, many of the earlier tutorials make heavy use of images and will therefore be inaccessible to blind students.

In writing this collection of modules *(tutorials)*, I will make a concentrated effort to make them accessible to blind students.

Some of the earlier tutorials that include images do so because the purpose of the tutorial was to teach students how to manipulate graphic images.

Other tutorials, however, included images in sample programs simply as a way to provide sensory feedback to the students and give them a feeling of accomplishment. I believe that providing sensory feedback in sample programs makes the process of learning how to program more interesting for the students.

Sound and music

In this collection of modules, I will use sound instead of images to provide sensory feedback. The sound will be of a form that is often referred to as *sampled sound*. In sampled sound, the actual waveform of the sound is sampled as a series of numeric values. At playback time, those numeric values are applied in sequence to a device that reproduces the original waveform and feeds that waveform to amplifiers, speakers, etc. This is the type of sound that is commonly found on a music CD.

While I am not a musician, I will also show you how to write a Java program that you can use to compose and play melodies based on the nomenclature used in sheet music (A, A#, B, C, C#, etc.). In fact, if you have a cursory knowledge of how to read sheet music *(or the accessible equivalent of sheet music)*, you will be able to copy the information from a piece of sheet music into a text file and play it on your computer using the program that you will write.

Here is a link to a file named MaryLamb.au containing a simple melody produced using that program. *(If you have musical talent, you can produce much more complex music*

than this simple melody.) You should be able to download and play this melody using any standard media player that supports sound files of type AU. However, you may need to revert to the *legacy* format of this module to download the file.

Writing, compiling, and running Java programs

Writing Java code

Fortunately, writing Java code is straightforward. You can write Java code using any plain text editor. You simply need to cause the output file to have an extension of .java.

There are a number of high-level *Integrated Development Environments (IDEs)* available, such as Eclipse and NetBeans, but they tend to be overkill for the relatively simple Java programs described in these modules.

There are also some low-level IDEs available, such as JCreator and DrJava, which are very useful for sighted students. However, I don't know anything about their level of accessibility. I normally use a free version of JCreator, mainly because it contains a color-coded editor, but that feature wouldn't be useful for a blind student.

So, just find an editor that you are happy with and use it to write your Java code.

Preparing to compile and run Java code

Perhaps the most complicated thing is to get your computer set up for compiling and running Java code in the first place.

Downloading the java development kit *(JDK)*

You will need to download and install the free Java JDK from the Oracle website. As of August 2014, you will find that website at
http://www.oracle.com/technetwork/java/javase/downloads/index.html

Installing the JDK

As of August 2014, you will find installation instructions for JDK 8 at
http://docs.oracle.com/javase/8/docs/technotes/guides/install/install_overview.html.

A word of caution

If you happen to be running Windows Vista, you may need to use something like the following when updating the PATH Environment Variable

```
;C:\Program Files (x86)\Java\jdk1.8.0\bin
```

in place of

```
;C:\Program Files\Java\jdk1.8.0\bin
```

as shown in the installation instructions.

I don't have any experience with any Linux version. Therefore, I don't have any hints to offer there.

The JDK documentation

It is very difficult to program in Java without access to the documentation for the JDK.

Several different types of Java documentation are available online at http://docs.oracle.com/javase/8/docs/.

Specific documentation for classes, methods, etc., for JDK 8 is available online at http://docs.oracle.com/javase/8/docs/api/.

It is also possible to download the documentation and install it locally if you have room on your disk. As of August 2014, you can download the documentation for JDK 8 from http://www.oracle.com/technetwork/java/javase/documentation/jdk8-doc-downloads-2133158.html.

Compiling and running Java code

There are a variety of ways to compile and run Java code. The way that I will describe here is the most basic and, in my opinion, the most reliable. These instructions apply to a Windows operating system. If you are using a different operating system, you will need to translate the instructions to your operating system.

Write your Java program

Begin by using your text editor to write your Java program *(see Listing 2 for example)* into one or more text files, each with an extension of .java. *(Files of this type are often referred to as source code files.)* Save the source code files in an empty folder somewhere on your disk. Make sure that the name of the **class** containing the **main** method *(which you will learn about in a future module)* matches the name of the file in which that class is contained *(except for the extension of .java on the file name, which does not appear in the class name)*.

Create a batch file

Use your text editor to create a batch file *(or whatever the equivalent is for your operating system)* containing the text shown in Listing 1 *(with the modifications discussed below)* and store it in the same folder as your Java source code files..

Then execute the batch file, which in turn will execute the program if there are no compilation errors.

Listing 1. Windows batch file.

```
echo off
cls

del *.class

javac -cp .; hello.java
java -cp .; hello

pause
```

Comments regarding the batch file

The commands in the batch file of Listing 1 will

- Open a command-line screen for the folder containing the batch file.
- Delete all of the compiled class files from the folder. *(If the folder doesn't contain any class files, this will be indicated on the command-line screen.)*
- Attempt to compile the program in the file named **hello.java.**
- Attempt to run the compiled program using a compiled Java file named **hello.class**.
- Pause and wait for you to dismiss the command-line screen by pressing a key on the keyboard.

If errors occur, they will be reported on the command-line screen and the program won't be executed.

If your program is named something other than **hello**, *(which it typically would be)* substitute the new name for the word **hello** where it appears twice in the batch file.

Don't delete the pause command

The **pause** command causes the command-line window to stay on the screen until you dismiss it by pressing a key on the keyboard. You will need to examine the contents of the window if there are errors when you attempt to compile and run your program, so don't delete the pause command.

Translate to other operating systems

The format of the batch file in Listing 1 is a Windows format. If you are using a different operating system, you will need to translate the information in Listing 1 into the correct format for your operating system.

A test program

The test program in Listing 2 can be used to confirm that Java is properly installed on your computer and that you can successfully compile and execute Java programs.

Listing 2. A test program.

```
class hello {
  public static void main(String[] args){
    System.out.println("Hello World");
  }
}
```

Instructions

Copy the code shown in Listing 2 into a text file named **hello.java** and store in an empty folder somewhere on your disk.

Create a batch file named **hello.bat** containing the text shown in Listing 1 and store that file in the same folder as the file named **hello.java**.

Execute the batch file.

If everything is working, a command-line screen should open and display the following text:

```
Hello World
Press any key to continue . . .
```

Congratulations

If that happens, you have just written, compiled and executed your first Java program.

Oops

If that doesn't happen, you need to go back to the installation instructions and see if you can determine why the JDK isn't properly installed.

If you get an error message similar to the following, that probably means that you didn't set the **path** environment variable correctly.

```
'javac' is not recognized as an internal or external command,
operable program or batch file.
```

Beyond that, I can't provide much advice in the way of troubleshooting hints.

Miscellaneous

This section contains a variety of miscellaneous information.

Housekeeping material

- Module name: Getting Started
- File: Jbs1000.htm
- Revised: 08/10/14
- Keywords:
    - object-oriented programming
    - accessible
    - accessibility
    - blind
    - Java
    - screen reader
    - refreshable Braille display

Baldwin.

Affiliation: Prof. Baldwin is a professor of Computer Information Technology at Austin Community College in Austin, TX.

-end-