# Accessible Objected-Oriented Programming Concepts for Blind Students

**By:**
Richard Baldwin

# Accessible Objected-Oriented Programming Concepts for Blind Students

**By:**
Richard Baldwin

**Online:**
< http://cnx.org/content/col11349/1.7/ >

**C O N N E X I O N S**

**Rice University, Houston, Texas**

Collection structure revised: January 17, 2013
PDF generated: January 17, 2013
For copyright and attribution information for the modules contained in this collection, see p. 47.

# Table of Contents

**1 Getting Started** ..... 1
**2 A Gentle Introduction to Java Programming** ..... 9
**3 A Gentle Introduction to Methods in Java** ..... 17
**4 Java comments** ..... 25
**5 Java Data Types** ..... 31
**Index** ..... 46
**Attributions** ..... 47

# Chapter 1

# Getting Started[1]

## 1.1 Table of Contents


- Preface (p. 1)
  - General (p. 1)
  - Prerequisites (p. 2)
  - Viewing tip (p. 2)
    - Listings (p. 2)
  - Supplemental material (p. 3)
- Discussion (p. 3)
  - Accessibility is the keyword (p. 3)
  - Sound and music (p. 3)
    - Sampled sound (p. 3)
    - MIDI sound (p. 3)
- Writing, compiling, and running Java programs (p. 4)
  - Writing Java code (p. 4)
  - Preparing to compile and run Java code (p. 4)
    - Downloading the java development kit *(JDK)* (p. 4)
    - Installing the JDK (p. 4)
    - The JDK documentation (p. 5)
  - Compiling and running Java code (p. 5)
    - Write your Java program (p. 5)
    - Create a batch file (p. 5)
    - A test program (p. 6)
- Resources (p. 7)
- Miscellaneous (p. 7)


## 1.2 Preface

### 1.2.1 General

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

[1] This content is available online at <http://cnx.org/content/m40794/1.1/>.

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, these modules make heavy use of programming projects that provide sensory feedback through the use of both sampled sound and MIDI 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.

See http://cnx.org/content/col11349/latest/ [2] for the main page of the collection. If you can see the main page, there is a Table of Contents on the left side of the page. I'm not sure how your screen reader will treat that Table of Contents relative to the other parts of the page.

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

### 1.2.2 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/ [3] .
- A refreshable Braille display capable of providing a line by line tactile output of information displayed on the computer monitor is recommended ( http://www.userite.com/ecampus/lesson1/tools.php [4] ).
- The Sun/Oracle Java Development Kit (JDK) *(See http://www.oracle.com/technetwork/java/javase/downloads/index.* [5] *)*
- Documentation for the Sun/Oracle Java Development Kit (JDK) *(See http://download.oracle.com/javase/7/docs/api/* [6] *)*
- A simple IDE or text editor for use in writing Java code.

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

- An understanding of algebra.
- An understanding of all of the material covered in the earlier modules in this collection.

### 1.2.3 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.

#### 1.2.3.1 Listings

- Listing 1 (p. 5) . Windows batch file.
- Listing 2 (p. 6) . A test program.

[2] http://cnx.org/content/col11349/latest/
[3] http://www.nvda-project.org/
[4] http://www.userite.com/ecampus/lesson1/tools.php
[5] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[6] http://download.oracle.com/javase/7/docs/api/

### 1.2.4 Supplemental material

I recommend that you also study the other lessons in my extensive collection of online programming tutorials. You will find a consolidated index at www.DickBaldwin.com [7] .

## 1.3 Discussion

Considering that during the past fifteen years, I have published several hundred online programming tutorials *(see http://www.dickbaldwin.com/toc.htm* [8] *)*, one might wonder why I am taking the time and expending the effort to publish still another online programming tutorial.

### 1.3.1 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 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.

### 1.3.2 Sound and music

In this collection of modules, I will use sound instead of images to provide sensory feedback.

#### 1.3.2.1 Sampled sound

In some cases, 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 numerical 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.

#### 1.3.2.2 MIDI sound

In other cases, I will use MIDI sound. Pronounced *middy* , this is an acronym for *musical instrument digital interface*. MIDI is a standard adopted by the electronic music industry for controlling devices, such as synthesizers and sound cards, that emit music. For example, this is the scheme that is usually employed by digital keyboards used in rock bands.

While I am not a musician, I will show you how to write Java code to play some simple melodies. Perhaps after completing this series of modules, you can continue learning on your own to create serious music using MIDI.

[7] http://www.dickbaldwin.com/toc.htm
[8] http://www.dickbaldwin.com/toc.htm

## 1.4 Writing, compiling, and running Java programs

### 1.4.1 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.

### 1.4.2 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.

#### 1.4.2.1 Downloading the java development kit *(JDK)*

You will need to download and install the free Java JDK from the Oracle/Sun website. As of August 2011, you will find that website at http://www.oracle.com/technetwork/java/javase/downloads/index.html [9]

While writing this module in August of 2011, I noticed that JDK 7 has been recently released. While I expect that it will work just fine, I haven't tried it yet. I am still running Java SE 6 Update 26. I plan to hold off for a few months before downloading and installing JDK 7.

Also there is a 64-bit version of the JDK, but I haven't tried it yet either because the computers in the labs at the college where I teach can't support it. I am still using the 32-bit version. I probably won't start using the 64-bit version until the computers in those labs are upgraded.

Whether you elect to use JDK 6 or JDK 7 in either the 32-bit or 64-bit version is strictly up to you. Any one of them should do the job very nicely.

#### 1.4.2.2 Installing the JDK

As of August 2011, you will find installation instructions for JDK 7 at http://download.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html [10] and you will find installation instructions for JDK 6 at http://www.oracle.com/technetwork/java/javase/index-137561.html [11] .

The installation instructions for JDK 7 are more complete than the installation instructions for JDK 6. Even if you are installing JDK 6, I recommend that you read the instructions for JDK 7 and note the additional information that you will find there, particularly the information having to do with setting the **path** environment variable.

**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\jdk1.6.0_26\bin
```

in place of

[9] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[10] http://download.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html
[11] http://www.oracle.com/technetwork/java/javase/index-137561.html

```
;C:\Program Files\Java\jdk1.7.0\bin
```

as shown in the installation instructions.

I don't have any experience with Windows 7 yet, so I don't have any hints regarding Windows 7. I don't have any experience with any Linux version. Therefore, I don't have any hints to offer there either.

#### 1.4.2.3 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://www.oracle.com/technetwork/java/javase/documentation/index.html [12] .

Specific documentation for classes, methods, etc., for JDK 7 is available online at http://download.oracle.com/javase/7/docs/api/ [13] . Similar documentation for JDK 6 is available at http://download.oracle.com/javase/6/docs/api/ [14] .

It is also possible to download the documentation and install it locally if you have room on your disk. The download links for JDK 6 and JDK 7 documentation are also shown on the page at http://www.oracle.com/technetwork/java/javase/downloads/index.html [15] .

### 1.4.3 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.

#### 1.4.3.1 Write your Java program

Begin by using your text editor to write your Java program 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)* .

#### 1.4.3.2 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 (p. 5) *(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 1.1: Windows batch file.**

```
echo off
cls

del *.class

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

[12] http://www.oracle.com/technetwork/java/javase/documentation/index.html
[13] http://download.oracle.com/javase/7/docs/api/
[14] http://download.oracle.com/javase/6/docs/api/
[15] http://www.oracle.com/technetwork/java/javase/downloads/index.html

```
pause
```

**Comments regarding the batch file**

The commands in the batch file of Listing 1 (p. 5) 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 (p. 5) is a Windows format. If you are using a different operating system, you will need to translate the information in Listing 1 (p. 5) into the correct format for your operating system.

#### 1.4.3.3 A test program

The test program in Listing 2 (p. 6) can be used to confirm that Java is properly installed on your computer and that you can successfully compile and execute Java programs.

**Listing 1.2: A test program.**

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

**Instructions**

Copy the code shown in Listing 2 (p. 6) 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 (p. 5) 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.

## 1.5 Resources

I will publish a module containing consolidated links to resources on my Connexions web page and will update and add to the list as additional modules in this collection are published.

## 1.6 Miscellaneous

This section contains a variety of miscellaneous information.

NOTE: **Housekeeping material**

- Module name: Getting Started
- File: Jb1000.htm
- Revised: 08/18/11
- Keywords:
  - object-oriented programming
  - accessible
  - accessibility
  - blind
  - Java
  - screen reader
  - refreshable Braille display

NOTE: **Disclaimers:** **Financial** : Although the Connexions site makes it possible for you to download a PDF file for this module at no charge, and also makes it possible for you to purchase a pre-printed version of the PDF file, you should be aware that some of the HTML elements in this module may not translate well into PDF.

I also want you to know that I receive no financial compensation from the Connexions website even if you purchase the PDF version of the module.

**Affiliation** : I am a professor of Computer Information Technology at Austin Community College in Austin, TX.

-end-

# Chapter 2

# A Gentle Introduction to Java Programming[1]

## 2.1 Table of Contents


- Preface (p. 9)
  - General (p. 9)
  - Prerequisites (p. 10)
  - Viewing tip (p. 10)
    - Figures (p. 10)
    - Listings (p. 10)
  - Supplemental material (p. 10)
- Discussion and sample code (p. 10)
  - Introduction (p. 10)
  - Compartments (p. 11)
  - Checkout counter example (p. 11)
  - Sample program (p. 14)
- Run the program (p. 15)
- Resources (p. 15)
- Miscellaneous (p. 15)


## 2.2 Preface

### 2.2.1 General

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

See http://cnx.org/content/col11349/latest/ [2] for the main page of the collection. If you can see the main page, there is a Table of Contents on the left side of the page. I'm not sure how your screen reader will treat that Table of Contents relative to the other parts of the page.

This module provides a gentle introduction to Java programming in a format that is accessible to blind students.

[1] This content is available online at <http://cnx.org/content/m40812/1.1/>.
[2] http://cnx.org/content/col11349/latest/

### 2.2.2 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/ [3] .
- A refreshable Braille display capable of providing a line by line tactile output of information displayed on the computer monitor is recommended ( http://www.userite.com/ecampus/lesson1/tools.php [4] ).
- The Sun/Oracle Java Development Kit (JDK) *(See http://www.oracle.com/technetwork/java/javase/downloads/index.* [5] *)*
- Documentation for the Sun/Oracle Java Development Kit (JDK) *(See http://download.oracle.com/javase/7/docs/api/* [6] *)*
- A simple IDE or text editor for use in writing Java code.

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

- An understanding of algebra.
- An understanding of all of the material covered in the earlier modules in this collection.

### 2.2.3 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 figures and listings while you are reading about them.

#### 2.2.3.1 Figures

- Figure 1 (p. 12) . A checkout counter algorithm.

#### 2.2.3.2 Listings

- Listing 1 (p. 14) . Program named Memory01.
- Listing 2 (p. 14) . Batch file for Memory01.

### 2.2.4 Supplemental material

I recommend that you also study the other lessons in my extensive collection of online programming tutorials. You will find a consolidated index at www.DickBaldwin.com [7] .

## 2.3 Discussion and sample code

### 2.3.1 Introduction

All data is stored in a computer in numeric form. Computer programs do what they do by executing a series of calculations on numeric data. It is the order and the pattern of those calculations that distinguishes one computer program from another.

[3] http://www.nvda-project.org/
[4] http://www.userite.com/ecampus/lesson1/tools.php
[5] http://www.oracle.com/technetwork/java/javase/downloads/index.html
[6] http://download.oracle.com/javase/7/docs/api/
[7] http://www.dickbaldwin.com/toc.htm

**Avoiding the detailed work**

Fortunately, when we program using a high-level programming language such as Java, much of the detailed work is done for us behind the scenes.

**Musicians or conductors**

As programmers, we are more like conductors than musicians. The various parts of the computer represent the musicians. We tell them what to play, and when to play it, and if we do our job well, we produce a solution to a problem.

### 2.3.2 Compartments

As the computer program performs its calculations in the correct order, it is often necessary for it to store intermediate results someplace, and then come back and get them to use them in subsequent calculations later. The intermediate results are stored in memory, often referred to as RAM or *Random Access Memory* .

**A mechanical analogy**

We can think of random access memory as being analogous to a metal rack containing a large number of compartments. The compartments are all the same size and are arranged in a column. Each compartment has a numeric address printed above it. No two compartments have the same numeric address. Each compartment also has a little slot into which you can insert a name or a label for the compartment. No two compartments can have the same name.

**Joe, the computer program**

Think of yourself as a computer program. You have the ability to write values on little slips of paper and to put them into the compartments. You also have the ability to read the values written on the little slips of paper and to use those values for some purpose. However, there are two rules that you must observe:

- You may not remove a slip of paper from a compartment without replacing it by another slip of paper on which you have written a value.
- You may not put a slip of paper in a compartment without removing the one already there.

### 2.3.3 Checkout counter example

In understanding how you might behave as a human computer program, consider yourself to have a job working at the checkout counter of a small grocery store in the 1930s.

You have two tools to work with:

- A mechanical adding machine
- The rack of compartments described above

**Initialization**

Each morning, the owner of the grocery store tells you to insert a name in the slot above each compartment and to place a little slip of paper with a number written on it inside each compartment. *(In programming jargon, we would refer to this as initialization.)*

Each of the names on the compartments represents a type of grocery such as

- Beans
- Apples
- Pears

No two compartments can have the same name.

No compartment is allowed to have more than one slip of paper inside it.

**The price of a can of beans**