
Examples for *On Java 8* by Bruce Eckel

If you want to experiment with the code examples from the book *On Java 8*, you're in the right place.

These examples are automatically extracted directly from the book. This repository includes tests to verify that the code in the book is correct.

NOTE: Do not attempt to use JDK 16 or greater with gradle. This produces a [BUG!](#) message from Gradle, which is broken for those versions.

Contents

- Building From the Command Line: Quick Version
- Building From the Command Line: Detailed Instructions
 - Install Java
 - ★ Windows
 - ★ Macintosh
 - ★ Linux
 - Verify Your Installation
 - Installing and Running the Book Examples
- Appendix A: Command-Line Basics
 - Editors
 - The Shell
 - ★ Starting a Shell
 - ★ Directories
 - ★ Basic Shell Operations
 - ★ Unpacking a Zip Archive
- Appendix B: Testing
- Appendix C: Troubleshooting

Building From the Command Line: Quick Version

Before you can run the examples from this repository, you must install JDK8, the *Java Development Kit* for version 8 of the language.

If you just want to download and check the code, Download Here and unzip it into your destination directory. Open a shell/command window and move into the root of that directory. You'll know you are in the right directory if you see the files `gradlew` and `gradlew.bat`.

You'll need an Internet connection the first time you compile the code, because Gradle needs to first install itself, then all the support libraries. Once these are installed you can perform additional compiling and running offline.

On Mac/Linux, enter:

```
1 ./gradlew test
```

(If you get a *Permission denied* error, run `chmod +x ./gradlew`)

On Windows, enter

```
1 gradlew test
```

If all goes well, the tests will run. Everything should complete without errors.

All the book examples are in the subdirectory `Examples` in subdirectories corresponding to the atom names.

Building From the Command Line: Detailed Instructions

If you are not familiar with the command line, first read Command-Line Basics.

Install Java

You must first install the *Java Development Kit* (JDK).

Windows

1. Follow the instructions to install Chocolatey.
2. At a shell prompt, type: `choco install jdk8` (you may also select a more recent version, like `jdk11`). The installation process takes some time, but when it's finished Java is installed and the necessary environment variables are set.

Macintosh

The Mac comes with a much older version of Java that won't work for the examples in this book, so you'll need to update it to (at least) Java 8.

1. Follow the instructions at this link to Install HomeBrew
2. At a shell prompt, first type `brew update`. When that completes, enter `brew cask install java`.

NOTE: Sometimes the default version of Java that you get with the above installation will be too recent and not validated by the Mac's security system. If this happens you'll either need to turn off the security by hand or install an earlier version of Java. For either choice, you'll need to Google for answers on how to solve the problem (often the easiest approach is to just search for the error message produced by the Mac).

Linux

Use the standard package installer with the following shell commands:

Ubuntu/Debian:

1. `sudo apt-get update`
2. `sudo apt-get install default-jdk`

Fedora/Redhat:

```
1 su -c "yum install java-1.8.0-openjdk"
```

Verify Your Installation

Open a new shell and type:

```
1 java -version
```

You should see something like the following (Version numbers and actual text will vary):

```
1 openjdk version "11" 2018-09-25
2 OpenJDK Runtime Environment 18.9 (build 11+28)
3 OpenJDK 64-Bit Server VM 18.9 (build 11+28, mixed mode)
```

If you see a message that the command is not found or not recognized, review the installation instructions. If you still can't get it to work, check StackOverflow.

Installing and Running the Book Examples

Once you have Java installed, the process to install and run the book examples is the same for all platforms:

1. Download the book examples from the GitHub Repository.
2. Unzip the downloaded file into the directory of your choice.
3. Use the Windows Explorer, the Mac Finder, or Nautilus or equivalent on Linux to browse to the directory where you unzipped [OnJava8-Examples](#), and open a shell there.
4. If you're in the right directory, you should see files named `gradlew` and `gradlew.bat` in that directory, along with numerous other files and directories. The directories correspond to the chapters in the book.
5. At the shell prompt, type `gradlew test` (Windows) or `./gradlew test` (Mac/Linux).

The first time you do this, Gradle will install itself and numerous other packages, so it will take some time. After everything is installed, subsequent builds and runs will be much faster.

Note that you must be connected to the Internet the first time you run `gradlew` so that Gradle can download the necessary packages.

Appendix A: Command-Line Basics

Because it is possible for a “dedicated beginner” to learn programming from this book, you may not have previously used your computer’s command-line shell. If you have, you can go directly to the installation instructions.

Editors

To create and modify Java program files—the code listings shown in this book—you need a program called an *editor*. You’ll also need the editor to make changes to your system configuration files, which is sometimes required during installation.

Programming editors vary from heavyweight *Integrated Development Environments* (IDEs, like Eclipse, NetBeans and IntelliJ IDEA) to more basic text manipulation applications. If you already have an IDE and are comfortable with it, feel free to use that for this book.

Numerous explanations in this book are specific to IntelliJ IDEA so if you don’t already have an IDE you might as well start with IDEA. There are many other editors; these are a subculture unto themselves

and people sometimes get into heated arguments about their merits. If you find one you like better, it's not too hard to change. The important thing is to choose one and get comfortable with it.

The Shell

If you haven't programmed before, you might be unfamiliar with your operating system *shell* (also called the *command prompt* in Windows). The shell harkens back to the early days of computing when everything happened by typing commands and the computer responded by displaying responses—everything was text-based.

Although it can seem primitive in the age of graphical user interfaces, a shell provides a surprising number of valuable features.

To learn more about your shell than we cover here, see [Bash Shell for Mac/Linux](#) or [Windows Shell](#).

Starting a Shell

Mac: Click on the *Spotlight* (the magnifying-glass icon in the upper-right corner of the screen) and type “terminal.” Click on the application that looks like a little TV screen (you might also be able to hit “Return”). This starts a shell in your home directory.

Windows: First, start the Windows Explorer to navigate through your directories:

- *Windows 7:* click the “Start” button in the lower left corner of the screen. In the Start Menu search box area type “explorer” and then press the “Enter” key.
- *Windows 8:* click Windows+Q, type “explorer” and then press the “Enter” key.
- *Windows 10:* click Windows+E.

Once the Windows Explorer is running, move through the folders on your computer by double-clicking on them with the mouse. Navigate to the desired folder. Now click the file tab at the top left of the Explorer window and select “Open Windows Powershell.” This opens a shell in the destination directory.

Linux: To open a shell in your home directory:

- *Debian:* Press Alt+F2. In the dialog that pops up, type ‘gnome-terminal’
- *Ubuntu:* Either right-click on the desktop and select ‘Open Terminal’, or press Ctrl+Alt+T
- *Redhat:* Right-click on the desktop and select ‘Open Terminal’
- *Fedora:* Press Alt+F2. In the dialog that pops up, type ‘gnome-terminal’

Directories

Directories are one of the fundamental elements of a shell. Directories hold files, as well as other directories. Think of a directory as a tree with branches. If `books` is a directory on your system and it has two other directories as branches, for example `math` and `art`, we say that you have a directory `books` with two *subdirectories* `math` and `art`. We refer to them as `books/math` and `books/art` since `books` is their *parent* directory. Note that Windows uses backslashes rather than forward slashes to separate the parts of a directory.

Basic Shell Operations

The shell operations shown here are approximately identical across operating systems. For the purposes of this book, here are the essential operations in a shell:

- **Change directory:** Use `cd` followed by the name of the directory where you want to move, or `cd ..` if you want to move up a directory. If you want to move to a different directory while remembering where you came from, use `pushd` followed by the different directory name. Then, to return to the previous directory, just say `popd`.
- **Directory listing:** `ls` (`dir` in Windows) displays all the files and subdirectory names in the current directory. Use the wildcard `*` (asterisk) to narrow your search. For example, if you want to list all the files ending in “.kt,” you say `ls *.kt` (Windows: `dir *.kt`). If you want to list the files starting with “F” and ending in “.kt,” you say `ls F*.kt` (Windows: `dir F*.kt`).
- **Create a directory:** use the `mkdir` (“make directory”) command (Windows: `md`), followed by the name of the directory you want to create. For example, `mkdir books` (Windows: `md books`).
- **Remove a file:** Use `rm` (“remove”) followed by the name of the file you wish to remove (Windows: `del`). For example, `rm somefile.kt` (Windows: `del somefile.kt`).
- **Remove a directory:** use the `rm -r` command to remove the files in the directory and the directory itself (Windows: `deltree`). For example, `rm -r books` (Windows: `deltree books`).
- **Repeat a command:** The “up arrow” on all three operating systems moves through previous commands so you can edit and repeat them. On Mac/Linux, `!!` repeats the last command and `!n` repeats the *n*th command.
- **Command history:** Use `history` in Mac/Linux or press the F7 key in Windows. This gives you a list of all the commands you’ve entered. Mac/Linux provides numbers to refer to when you want to repeat a command.

Unpacking a Zip Archive

A file name ending with `.zip` is an archive containing other files in a compressed format. Both Linux and Mac have command-line `unzip` utilities, and it's possible to install a command-line `unzip` for Windows via the Internet.

However, in all three systems the graphical file browser (Windows Explorer, the Mac Finder, or Nautilus or equivalent on Linux) will browse to the directory containing your zip file. Then right-mouse-click on the file and select "Open" on the Mac, "Extract Here" on Linux, or "Extract all ..." on Windows.

Appendix B: Testing

The test system is built in so that we (the authors) can verify the correctness of what goes into the book.

You don't need to run the tests, but if you want to, you can just run `gradlew test` (on Windows) or `./gradlew test` (Mac/Linux).

To compile and run everything, the command is:

```
gradlew run
```

If you are on a Unix/Linux based system, you must select the local directory for all commands, for example:

```
./gradlew run
```

To only compile everything, the command is:

```
gradlew compileJava
```

To compile only a single chapter (including dependencies), use for example:

```
gradlew :strings:compileJava
```

To run only a single chapter, say:

```
gradlew :strings:run
```

Gradle can also be used to run a single program. Here, we run the **ReplacingStringTokenizer.java** program in the **strings** chapter subdirectory:

```
gradlew :strings:ReplacingStringTokenizer
```

However, if the file name is unique throughout the book (the majority are), you can just give the program name, like this:

gradlew ReplacingStringTokenizer

Note that all commands are run from the base directory where the example code is installed, and where you find the `gradlew` script.

You can learn about other options by just typing `gradlew` with no arguments.

Appendix C: Troubleshooting

If any terminology or processes described here remain unclear to you, you can usually find explanations or answers through Google. For more specific issues or problems, try StackOverflow. Sometimes you can find installation instructions on YouTube.

Sometimes a Gradle build will be unable to connect to the internet and download the necessary components, producing an error message containing:

```
1 javax.net.ssl.SSLException: java.lang.RuntimeException: Unexpected
  error: java.security.InvalidAlgorithmParameterException: the
  trustAnchors parameter must be non-empty
```

Normally this means you have multiple Java installations on your machine (applications built with Java ordinarily install their own version of Java), and somehow the `cacerts` security file is interfering with the `cacerts` file for the Java you have installed. It can be difficult to know which `cacerts` file is interfering with yours. The brute-force approach is to search for all the `cacerts` files on your machine and begin uninstalling the associated applications—or in some cases, simply removing the directory containing the `cacerts` file—until the Gradle build begins to work. You might also need to adjust some environment variables and/or your path. Once you get the Gradle build working successfully, you should be able to reinstall any applications you removed in the process.