OpenJDK Cookbook

OpenJDK is one of the most widely used open source implementations of the Java platform. It is used to change, customize, and tune core application internals and provide a way to extend the application internals according to your requirements.

OpenJDK Cookbook begins by introducing you to OpenJDK and IcedTea builds for various virtual machine implementations and how to deploy OpenJDK on multiple platforms. Furthermore, the book digs deeper into the development concepts, JVM internals, and techniques to make robust improvements or customizations to OpenJDK. Essentially, the book covers the best practices for accessing and using the core features of OpenJDK to build advanced Java solutions by utilizing the more complex and nuanced parts of OpenJDK.

What this book will do for you...

- Set up a development environment and start hacking OpenJDK
- Understand advanced features such as adding new intrinsics to JVMs
- Build different versions of OpenJDK for different operating systems and architectures
- Write reliable tests using jtreg and get a good understanding of the tool
- Create and submit limit patches with improvements and bug fixes
- Understand valuable techniques, native code debugging, and incremental builds
- Prepare VirtualBox machines with different operating systems
- Understand the process behind JEPs


Over 80 recipes to build and extend your very own version of the Java platform using OpenJDK project

Alex Kasko
Stanislav Kobylyanskiy
Alexey Mironchenko

Prices do not include local sales tax or VAT where applicable.
In this package, you will find:

- The author biography
- A preview chapter from the book, Chapter 1 "Getting Started with OpenJDK"
- A synopsis of the book’s content
- More information on OpenJDK Cookbook

About the Author

Alex Kasko is a participant in the OpenJDK project. He maintains unofficial OpenJDK builds on his GitHub account and has 8 years’ experience in enterprise and high performance programming. He works in an OpenJDK development team at Red Hat Inc.

Stanislav Kobylyanskiy is a software developer with years of Java experience. He started his career with C++ and system programming in the late 90s with the Aelita software (now DELL). After a few years, he switched to Java and then moved to telecom. At that time, he joined T-Mobile, UK, to rebuild their Customer Service Web Portal, which lasted for about 4 years. Currently, he is with an investment bank where he is working on a strategic algorithmic trading platform. He is continuously looking for new challenges and to extend his knowledge of core Java technologies.

I want to say thank you to my family—my two lovely daughters, Alyssa and Alexandra, and to my beloved wife, Natalia—for all their help and support in everything I do.

Alexey Mironchenko is a software developer with experience in scalable enterprise projects, involving Java EE, NoSQL databases, and various other frameworks. He has a mathematical background with some COQ and Maxima experience, and his hobby is to test cutting-edge technologies that are open source or in early access.
OpenJDK Cookbook

OpenJDK is a unique project that opens numerous and exciting opportunities for people who want to dive into the huge and complicated infrastructure behind the JVM. There is an incredible amount of things to learn and to explore. Almost anyone can find something in it as per their interest, starting from HTTP, Web, software dependency problems, and ending with hardware-specific JIT optimization techniques, and concurrency challenges. Such variety is very unique and it would be true to say that there is no other open source project that can provide something similar. The other factor is that there are not so many other open source projects on that scale; possibly only the Linux core. Such scale requires a non-trivial organizational approach and, to be involved in that process, to see how it works, is a very interesting insight.

This cookbook will lead you through steps to take you into the world of OpenJDK as smoothly as possible. It starts by explaining how to download the source code and how to build the different versions of OpenJDK, how to set it up on a machine, and what different options are available. Then, you will learn how to set up the development environment (IDE) required for editing and debugging C++ and Java source code, and how to start making changes. It will go through some examples, which you may decide to change in various parts of OpenJDK. Further, it will cover the tools available for testing, benchmarking, and ensuring that the changes you have made are not breaking the existing functionality. As OpenJDK is a big project with its own rules and processes, there will be a part covering the procedures that are involved in making changes or fixing bugs, the lifecycle of projects, JSRs, JEPs, and so on. At the end, there will be a section about future work that is planned to be included in forthcoming releases; that part will be the most interesting section for anyone who is interested in the future direction of OpenJDK and wants to try something new, which is not yet available in the stable product.

In addition, this book contains many practical examples which should be useful to any developer who is working with OpenJDK or any other Java technology. They are available in simple form, which allows you to quickly copy and use them for your own project.

What This Book Covers

Chapter 1, Getting Started with OpenJDK, provides an overview of OpenJDK, explains what it is, and covers the basic steps required to have OpenJDK running and properly configured on the machine.

Chapter 2, Building OpenJDK 6, covers the steps required to build OpenJDK Version 6. This build is very different from OpenJDK 7 and OpenJDK 8 and requires more manual work to be done.
Chapter 3, Building OpenJDK 7, covers the steps required to build OpenJDK Version 7. Building OpenJDK 7 is an easier and more enjoyable process, compared to OpenJDK 6.

Chapter 4, Building OpenJDK 8, covers the steps required to build OpenJDK Version 8.

Chapter 5, Building IcedTea, teaches you how to build a set of tools that are developed apart from OpenJDK. These tools are replaced with some proprietary bits that are not available as open source, which include a browser plugin, Java WebStart, and so on.

Chapter 6, Building IcedTea with Other VM Implementations, covers some interesting VM projects, which also can benefit from the features provided by IcedTea, and how to build that product using these VMs and non-x86 CPUs.

Chapter 7, Working with WebStart and the Browser Plugin, will cover the configuration and installation of WebStart and browser plugin components, which are the biggest parts of the Iced Tea project.

Chapter 8, Hacking OpenJDK, covers some bits which are required to start digging into the OpenJDK source code. Such things are the installation and setup of IDE, debugging, and updating HotSpot source code. There are also some useful examples of what the developer can do, for example, implementing your own intrinsic details.

Chapter 9, Testing OpenJDK, will go through an approach used in OpenJDK to test the source code and, since writing code is not enough, we need to write high quality product on which the code has to be tested. This chapter will also show you some examples to use the latest available tools.

Chapter 10, Contributing to OpenJDK, explains how OpenJDK is changing and evolving, how changes are executed, and what one needs to do to participate or to facilitate changes in OpenJDK. Some of these changes, if they are big enough, can take years to appear in the production version.

Chapter 11, Troubleshooting, teaches you about one of the most important parts of any project: bug fixing. It is important to understand which tools and processes are involved. In this chapter, we will cover some important steps, from submitting defects, to pushing the fix into the shared repository.

Chapter 12, Working with Future Technologies, covers some future developments in OpenJDK. As with any big project, OpenJDK has a roadmap with some exciting and promising projects for the next releases. That is exactly what this chapter is about. It lists all the steps required to download sources, build, and run some examples, where possible.

Chapter 13, Build Automation, provides some useful tips for automating the build process. It will be useful for those developers who make frequent changes in OpenJDK or always want to have the build with all the latest changes.
1

Getting Started with OpenJDK

In this chapter, we will cover the following topics:

- Distinguishing OpenJDK from Oracle JDK
- Installing OpenJDK on Windows
- Configuring OpenJDK on Windows
- Installing OpenJDK on Linux
- Configuring OpenJDK on Linux
- Navigating through OpenJDK groups and projects

Introduction

OpenJDK is now an official Java 7 reference implementation, and is now one for Java 8 as well. This means that the most essential projects of the Java ecosystem are now open source. This also means that OpenJDK can be installed in various ways—from building from source to installing the binary package from the package manager, if any.

Sun’s effort to release an open source JDK was the beginning of the project publicly stated in 2006, during the JavaOne conference. HotSpot was released under the GPLv2 license. The complete source code of the Java Class Library (JCL) was released in May 2007 under the GPL, except for several proprietary parts with GPL-incompatible licenses.

However, some proprietary parts (from 4 to 1 percent of the total code lines, depending on the update number) were required in OpenJDK in a separate proprietary bundle in the OpenJDK 7 b53 update in April 2009.
One may think that the initial installation and configuration are quite simple and do not need some sort of detailed explanation. In many ways, that's true; but there are some difficulties along the way.

We will start by distinguishing OpenJDK from Oracle JDK. The latter is based on the former, but not entirely. Each one has its own advantages and drawbacks. The primary OpenJDK advantage is the fact that it's open source while Oracle JDK is always recommended and is ready-to-use. Besides, OpenJDK 6 is the only maintainable Java 6 realization left, after the Java 6 discontinuation.

Then we will cover the installation process for Windows and possible issues with some Windows versions. After that, we will describe some typical profiles to configure an installed instance of OpenJDK for various needs, such as a server instance and developer instance.

Then we will go in to more complicated matters, such as installing OpenJDK on various Linux systems. There are at least two common ways to do it: the distribution-recommended way that depends on the distribution itself, and another way, which is common for all Linux systems.

Linux configuration is more or less the same as that of Windows, but there are some differences that need to be covered. The differences are, mainly, related to system philosophy, namely the way it's done and then what exactly is done.

Then we will proceed to OpenJDK internal structures—in an introductory way. We will consider OpenJDK projects that are already in use, and will learn how to use instruments that we will need later. Also, we will briefly look at OpenJDK groups and find out what they are doing and how they may influence OpenJDK's further evolution.

Last but not least, you will learn how to benefit from the Adopt OpenJDK program, which is also a part of the OpenJDK community. Adopt OpenJDK is an effort to improve OpenJDK usability readiness, test new language releases, and do whatever it needs to make OpenJDK more useful and welcoming among users as well as developers.

This chapter is written with an introductory purpose, and does not cover some details that are common to Oracle Java. However, it provides a necessary basis to work with.

We will use Java 7, as it is stable and the latest Java version available. All screenshots and processes are assuming that we use Java 7, if another is not explicitly mentioned.

If you already have OpenJDK built and installed as default and you are aware of the differences between OpenJDK and Oracle JDK, as well as of the existence of Adopt OpenJDK, you may skip this chapter entirely.
Distinguishing OpenJDK from Oracle JDK

Though OpenJDK is an official reference implementation for the Java platform, certain Oracle-provided software are not open source. The most famous of them is the Java browser plugin, but there are a lot more differences than just that. This recipe will show you how to distinguish OpenJDK from Oracle JDK.

Getting ready

To follow this recipe, you will need an installed OpenJDK instance. It will be good if you have an Oracle JDK instance as well, to feel the difference. Also, we will assume that you have a Linux installation and an `update-java-alternatives` command installed and ready to use. To know how to install OpenJDK on various systems, see the later recipes in this chapter. To know how to switch the system Java version, if you do not have `update-alternatives` installed (for Fedora, Gentoo, and so on), visit the Configuring OpenJDK on Linux recipe or refer to your distribution documentation/forums.

How to do it...

Please take a look at the following procedures to know the difference between OpenJDK and Oracle JDK:

1. We will open a terminal and type the following command:
   ```
   update-java-alternatives --list
   ```

2. We will see a full list of installed Java implementations:
   ```
   $ update-java-alternatives --list
   java-1.6.0-openjdk-amd64 1061 /usr/lib/jvm/java-1.6.0-openjdk-amd64
   java-1.7.0-openjdk-amd64 1071 /usr/lib/jvm/java-1.7.0-openjdk-amd64
   java-6-oracle 1073 /usr/lib/jvm/java-6-oracle
   java-7-oracle 1081 /usr/lib/jvm/java-7-oracle
   java-8-oracle 1082 /usr/lib/jvm/java-8-oracle
   ```

3. Let’s set Oracle Java as default. We will run the following command with root access:
   ```
   update-java-alternatives --set java-7-oracle
   ```
Getting Started with OpenJDK

This command may produce errors such as "no alternatives for apt". It's OK, just ignore them.

4. Then we will go to https://www.java.com/en/download/installed.jsp?detect=jre and check our browser plugin version. We will see the activate link (following the name of the activating entity).

We can see from the result of our actions that the Java browser plugin has been installed.

5. Let's try to set OpenJDK as the default Java environment (the actual instance name may differ in your case):

   `update-java-alternatives --set java-1.7.0-openjdk-amd64`

6. Then we will go to our browser page and refresh it. It may be necessary to restart the browser so that the changes can take effect, as shown in the following screenshot:

   ![Detecting Java on your computer](image)

   We can see that the plugin is not from the JDK itself but from a project named IcedTea.

   IcedTea is an open source project, whose goal is to replace proprietary parts of the Java ecosystem as much as possible. The plugin itself is from IcedTea-Web, an open source implementation of the Java Web Start and Java browser plugins.

   In most distributions, the IcedTea plugin is installed by default. But it is necessary to keep in mind that it's an open source plugin, and definitely not a referenced one. This means that its functionality might be slightly different from the Oracle plugin. It is also possible that some features may not work.

How it works...

Oracle JDK still has some proprietary components, and the browser plugin is one example. All we need in this chapter is to see the difference between the work of OpenJDK and Oracle JDK components that are different.
Also, the huge difference between OpenJDK and Oracle JDK lies in the license. OpenJDK is open source, while Oracle JDK contains proprietary pieces, and thus it is licensed under the Oracle binary license. The fact that OpenJDK is open source provides a whole new range of benefits (and exciting discoveries) through the ability to study and modify its source code. It is also worth mentioning that more than 90 percent of Oracle JDK is based on OpenJDK source code. This means the OpenJDK quality is not compromised in any way. The browser plugin is not the only thing that is missed in OpenJDK compared to Oracle JDK.

See also

- In Chapter 5, Building IcedTea, there is a detailed explanation of how to build IcedTea from source.

Installing OpenJDK on Windows

Windows is the most commonly used OS in the world, and many developers are using it as their primary system. Despite its popularity, Windows doesn't have such strong support by the OpenJDK development community, and installation of the product is not as easy as in Linux. This recipe will cover the steps required to install OpenJDK on Windows.

This recipe provides an easy but decentralized way to install programs, although the most recent versions provide package repositories of their own. However, on Windows, the only official way to install an up-to-date OpenJDK is to build it from source.

Getting ready

To follow this recipe, we will need an installed Windows system. Windows 7 or Windows 8 will be best, because Windows XP is already officially discontinued by Microsoft.

How to do it...

There is an official build of OpenJDK on Windows, but it exists for referential purposes only. It is official and easy to install, but it doesn't have any security updates or improvements. However, there are unofficial builds, maintained by Alex Casco. We will try to install OpenJDK in both ways:

1. We will start with an official reference build. To get it, we need to go to https://jdk7.java.net/java-se-7-ri/ and accept the license terms. Then, download and run the installer.
Getting Started with OpenJDK

Though the OpenJDK source code is licensed by an open license, this official build is licensed by the Oracle Binary Code license and by GPLv2. If you want to keep your OpenJDK open source, please use one licensed by GPLv2.

2. Unpack the downloaded file in a location you prefer. Let's name it C:/OpenJDK.

3. Open the Windows command line by navigating to Start | Run, type cmd, and click on the Run button.

4. Run the following command:
   ```
   C:\OpenJDK\bin\java.exe -version
   ```
   It will output the Java version information. The output should look like this:
   ```
   openjdk version 1.7.0
   OpenJDK Runtime Environment <build 1.7.0-b146>
   OpenJDK Client VM <build 21.0-b16, mixed mode>
   ```

Congratulations! We've just installed the OpenJDK official binary.

How it works...

The reference implementation is the only available official binary build of OpenJDK. But it lacks security and is used only for reference purposes. It is a simple archive that needs to be unpacked to use it.

To bypass this unpleasantness and give Windows users an opportunity to install OpenJDK as a binary without building it from source, one of the OpenJDK contributors established a completely unofficial but very useful OpenJDK build set for various platforms.

Moreover, this binary build, unlike the official one, is open source and licensed over GPL. So we can use it even in a completely open source environment without adding any proprietary pieces that will possibly get us in trouble.

There's more...

Though an official reference binary is outdated, there is an unofficial project that provides OpenJDK builds from sources that are up to date.
Now we will install OpenJDK 7 from unofficial builds:

2. Select an appropriate build for Windows and download it.
3. Unpack it and run `install.exe`.
4. Click on the Run button when the preceding message appears.
5. Carefully read and accept the license and then click on Next.
6. Select the installation path in the next window. It will point to your home directory by default, so be careful—such an installation may be available to no one but yourself.
7. If the target directory does not exist, let the installer create it.
8. Check the red highlighted checkboxes in the preceding screenshot only if you want to set this JDK as default for all users in the system, not just for you. You may uncheck the fourth box if you don't need this JDK to be default at all.

9. Then click on the Next button and wait until the installation is finished.

10. Then click on the Next button for the last time.

11. Then click on Done.

See also

Although the simplest way to install OpenJDK is to unpack binaries, manually or in an automatic fashion, there is no doubt that working with the source code will give us more flexibility at all possible levels.

To know more, read the following chapters:

- Chapter 2, Building OpenJDK 6 up to Chapter 4, Building OpenJDK 8 to learn about building OpenJDK from source
- Chapter 6, Building IcedTea with Other VM Implementations to build OpenJDK using other VMs
- Chapter 13, Build Automation to work with future technologies, which will be unavailable in binary form for quite some time

Configuring OpenJDK on Windows

Although the initial configuration is sufficient for most tasks, it may still be required to do some configuration. In the case of OpenJDK, this is performed by setting system variables. Here we will touch only on the case that often occurs when JDK is unpacked manually—how to set it as default.

Getting ready

To follow this recipe, we will need an OpenJDK instance installed on our Windows system. Windows 7 or Windows 8 will be best, because Windows XP is already officially discontinued by Microsoft.
How to do it...

At first, we need to install our OpenJDK implementation as the default Java instance. This is often necessary for development:

1. In order to do so, we will go to **Start** | **Control Panel** | **System** | **Advanced** | **Environment Variables** | **User Variables** (or **System Variables** for system-wide configuration) and add the path to the Java executable to the **PATH** system variable, as shown:

   ![Image of environment variables settings](image)

   If there are other paths to other Java executables, we may need to delete them as well, but it will be better to remember them, since we may need to restore our old default Java settings.

2. If we were installing OpenJDK from unofficial builds, there may be no need to change the **PATH** variable at all.
3. To validate our newly configured variable, we will go to the Command Prompt and type the following:
   java -version

4. The expected output is the version of our newly installed build.

**How it works...**

In order to set a newly installed OpenJDK instance as the default JDK, we need to change the system variable. After that change, our Java executables will be visible to the system.

**There's more...**

The same procedure is followed to set the CLASSPATH variable. It is not very necessary, and if you are using other libraries such as GNU classpath, you probably know about it.

**Installing OpenJDK on Linux**

The Linux operating system allows for many internal tweaks, as well as for changes to the system's source code. It is also known as a complicated OS, and not all distributions are user-friendly. There are many people using it, and it's open source, such as OpenJDK itself. The installation process varies between chosen distributions, and we will go through the process for the three most-used package managers, as well as through the process that will work for virtually all x86 Linux distributions.

**Getting ready**

To follow this recipe, you will need an installed Linux system. It will be better if it has the kernel version 2.6 or higher, though OpenJDK is reported workable on 2.4 kernels as well. Also, if you have the .deb, .rpm, or .ebuild package manager, the recommended way to install any package is to install it using those.

**How to do it...**

When the installation of various packages is concerned, the process is dependent on our Linux distribution.

For a Debian-based distribution:

1. Open a terminal and type:
   apt-get install openjdk-7-jdk
We should have root permissions or use *sudo* to gain access to system files.

2. This will trigger the installation automatically. If we get an error message, indicating that the package is not found, we should Google an appropriate name for an OpenJDK package for our distribution.

For an RPM-based distribution, we'll need to first search for the package names, because package names are varied between different distributions, as shown here:

```
$ yum search openjdk
```

You will see an output like this:

```
java-1.6.0-openjdk.x86_64 : OpenJDK Runtime Environment
java-1.6.0-openjdk-demo.x86_64 : OpenJDK Demos
java-1.6.0-openjdk-devel.x86_64 : OpenJDK Development Environment
java-1.6.0-openjdk-javadoc.x86_64 : OpenJDK API Documentation
java-1.6.0-openjdk-src.x86_64 : OpenJDK Source Bundle
java-1.7.0-openjdk.x86_64 : OpenJDK Runtime Environment
java-1.7.0-openjdk-demo.x86_64 : OpenJDK Demos
java-1.7.0-openjdk-devel.x86_64 : OpenJDK Development Environment
java-1.7.0-openjdk-javadoc.noarch : OpenJDK API Documentation
java-1.7.0-openjdk-src.x86_64 : OpenJDK Source Bundle
```

You may install all of the packages that have the desired version. Then, we will run another command, using the package name we've just found:

```
$ yum install <a found package name>
```

This will also trigger an automatic download and installation.

If we have a Gentoo-based distribution, just type the following:

```
$ emerge openjdk-1.7
```

This will, depending on your distribution, unpack and install a binary package or, more probably, automatically build this package from source.

**There's more...**

Aside from the recommended ways, there is a generic installation procedure. It is quite simple, though it may do some damage to your operating system, so don't use it unless you really know what you're doing:

```
1. This is the way to unpack the OpenJDK system and then install it yourself. To get
the builds, we will refer again to the unofficial build page, https://github.com/
alexkasko/openjdk-unofficial-builds.
```
Getting Started with OpenJDK

2. Then unpack the downloaded package into a folder and run the following command from it:
   ```
   java -jar ./install.jar
   ```

3. A GUI installer window will appear. Read and accept the license, choose the directory, and allow OpenJDK to be created, if it does not exist, as shown:

   ![Select Installation Packages](image)

   - Check the preceding checkbox if you want to make this installation the default one. Then click on the **Next** button.
   - Wait for the installation to complete, and click on **Next** for the last time.
   - Then click on **Done**.

Configuring OpenJDK on Linux

Linux configuration profiles differ from the Windows ones, because those systems are working with resources as well as with hardware in a slightly different fashion. Here, we will briefly explain those differences and a way to overcome them. Moreover, different Linux distributions, as always, have different ways to deal with the configuration. We will try to pass through the most noticeable ones.
Getting ready

To follow this recipe, you will need an OpenJDK instance installed on a Linux system. The deb, rpm, or ebuild distributions will suite us really well, although we will see a method for generic Linux configuration also.

Also, we will need the bash startup files to be installed properly.

In most Linux distributions, the generic way to configure anything that needs root access is not recommended, and the results of such an approach tend to vanish with each update. Usually, there are distribution-recommended how-to's where the problem solution is described.

How to do it...

First let's check whether your bash startup files are installed. The simplest possible way is to configure your OpenJDK using them. They are system-wide and easy to use, though there are drawbacks in their usage, such as update conflicts:

1. Type the following line in your terminal:
   ```bash
cat /etc/profile
   ```
   If the file exists and contains some kind of shell script, then your bash startup file's setup is probably correct. If not, please set it up by following your distribution instructions.

2. Then add the `/etc/profile.d/openjdk.sh` file.

3. In order to configure different things, write the following:
   ```bash
   To set JAVA_HOME
   JAVA_HOME=<youJDK installation directory>
   export JAVA_HOME

   To append JAVA_HOME to PATH
   pathappend $JAVA_HOME/bin PATH

   To adjust CLASSPATH directory
   AUTO_CLASSPATH_DIR=<classpath dir>
   pathprepend . CLASSPATH

   for dir in `find ${AUTO_CLASSPATH_DIR} -type d 2>/dev/null`; do
     pathappend $dir CLASSPATH
   done
   ```
The CLASSPATH env variable should be avoided as much as possible. It is generally used by legacy Java applications mostly configured for JDK 1.2 and below. Use `-classpath` with `java` and `javac` commands instead.

The preceding code is quite simple—it just appends all JAR files to the classpath.

How it works...

This script is called during shell initialization, so whenever you perform shell initialization, these variables will be exported. The variables are thus system-wide, so be careful while playing with them, as they can cause your Java to fail permanently if you make some errors in this file.

There's more...

On Linux, you can see the directory structure of the installed OpenJDK using the `tree` command.

To do so, install the `tree` package (use your distribution's documentation if possible) and type:

```
tree -L 1 <path-to-openjdk> -lah
```

You will see something like the following:

```
/usr/lib/jvm/java-7-openjdk-amd64
├── [ 22] ASSEMBLY_EXCEPTION -> jre/ASSEMBLY_EXCEPTION
├── [4.0K] bin
├── [ 41] docs -> ../../share/doc/openjdk-7-jre-headless
├── [4.0K] include
├── [4.0K] jre
```
Chapter 1

This is the first-level directory structure in which:

- `ASSEMBLY_EXCEPTION` is about licensing, and so is `THIRD_PARTY_README`.
- The `docs` folder is for various OpenJDK documentation (changelog, copyrights, authors, and so on).
- The `include` directory is to include paths (for JNI, for example).
- The `jre` directory is where the Java Runtime is placed.
- The `lib` directory is where various OpenJDK libraries are placed (such as Jigsaw or CORBA support; mainly, it consists of all OpenJDK code).
- The `man` command is a manual pages entry for OpenJDK. It contains OpenJDK classes, javadocs, and other manual entries. It may be extremely useful in the highly improbable event of Internet connection loss.

**Navigating through OpenJDK groups and projects**

OpenJDK is not one huge project. It consists of a large number of subprojects, and is developed by relatively small groups of developers. We will look at them and realize what is going on under the hood of OpenJDK.

**Getting ready**

To follow this recipe, you will need an OpenJDK instance installed and an established Internet connection. The recipe is more for the initial understanding of the process rather than for practical use, so if you are familiar with these matters, don’t hesitate to skip this recipe entirely.

**How to do it...**

We will see what OpenJDK consists of:

2. In the right column, there are overviews of groups as well as projects.
3. We will select one of them to get through the process. The process will be described in detail in Chapter 8, *Hacking OpenJDK*.

4. Let the selected project be JDK9.

5. Go to the JDK9 project page at http://openjdk.java.net/projects/jdk9/.

There's not very much to see, because there is only the basic on-boarding information. Most of the project business is in the bug tracker.

After we read the information about the project on an official site, we will go to JDK JIRA to see what's happening here. We will go to the JDK9 part of JIRA at https://bugs.openjdk.java.net/browse/JDK/fixforversion/14949.

Here we can see various issues related directly to JDK9 and see how the process is going.

### How it works...

Groups are sets of developers who may work on different projects but in one large scope. Developers participate in chosen projects, and projects are sponsored by groups.

To participate in a group and become a contributor, follow the instructions at http://openjdk.java.net/contribute/.

There are four major types of projects:

- Feature
- Improvement
- Replacement
- Portability

For example, the JDK9 project is a featured one. The graphics rasterizer project is a replacement one, while the Swing group is a whole group that is focused on the improvement of Swing.

Various ports' projects are obviously the portability ones.

### See also

- See Chapter 8, *Hacking OpenJDK*, Chapter 11, *Troubleshooting*, and Chapter 13, *Build Automation*—they will suit you just well
- Tails in Chapter 4, *Building OpenJDK 8*
Where to buy this book

You can buy OpenJDK Cookbook from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.