The JFreeChart Class Library

Version 1.5.6

Introductory Guide

Written by David Gilbert

May 23, 2025

IMPORTANT NOTICE:
I work hard to make this document as accurate and informative as I can, but cannot guarantee that it is error-free.

Contents

Chapter 1

Introduction

1.1 What is JFreeChart?

1.1.1 Overview

JFreeChart is a free chart library for the Java(tm) platform. It is designed for use in client-side applications (JavaFX and Swing) and server-side applications. JFreeChart is distributed with complete source code subject to the terms of the GNU Lesser General Public Licence, which permits JFreeChart to be used in proprietary or free software applications (see Appendix C for details).

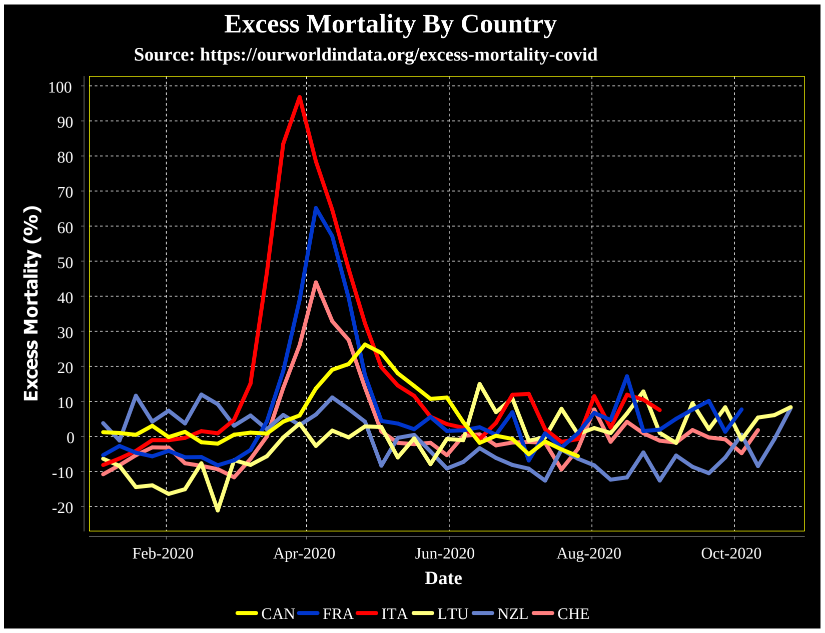


Figure 1.1: A sample chart

Figure 1.1 shows a typical chart created using JFreeChart. Many more examples are shown in later sections of this document.

1.1.2 Features

JFreeChart can generate pie charts, bar charts (regular and stacked), line charts, scatter plots, time series charts (including moving averages, high-low-open-close charts and candlestick plots), Gantt charts, meter charts (dial, compass and thermometer), symbol charts, wind plots, combination charts and more.

Additional features include:

- data is accessible from any implementation of the defined interfaces;
- export to PNG and JPEG image file formats (or you can use Java's ImageIO library to export to any format supported by ImageIO);
- export to any format with a `Graphics2D` implementation including:
  - PDF via JFreePDF (`https://github.com/jfree/jfreepdf`);
  - SVG via JFreeSVG (`https://github.com/jfree/jfreesvg`);
- tool tips;
- interactive zooming (drag region and/or mouse-wheel) and panning;
- chart mouse events (these can be used for drill-down charts or information pop-ups);
- annotations;
- HTML image map generation;
- distributed with complete source code subject to the terms of the GNU Lesser General Public License (LGPL);

JFreeChart is written entirely in Java, and should run on any implementation of the Java 2 platform (JDK 1.8.0 or later). JavaFX support is available—see `https://github.com/jfree/jfreechart-fx`.

1.1.3 Home Page

The JFreeChart project is hosted at GitHub:

`https://github.com/jfree/jfreechart`

Here you will find the latest JFreeChart releases, the source code, an issue tracker and other information about the project.

There is also a JFreeChart home page at:

`http://www.jfree.org/jfreechart/`

Here you will find more information about JFreeChart, including sample charts, Javadocs, an historical discussion forum (read only) and more.

1.2 This Document

1.2.1 Versions

Two versions of this document are available:

- a free version, the "JFreeChart Introductory Guide", is available from the JFreeChart home page, and contains basic information about using JFreeChart;
- a premium version, the "JFreeChart Developer Guide", is available only to those that have sponsored the JFreeChart project, and includes additional tutorial chapters and reference information.

If you wish to sponsor the JFreeChart project, please visit the following site:

`https://github.com/sponsors/jfree`

I'd like to thank everyone that has supported JFreeChart in the past by purchasing the JFreeChart Developer Guide!

1.2.2 Disclaimer

Please note that I have put in considerable effort to ensure that the information in this document is up-to-date and accurate, but I cannot guarantee that it does not contain errors. You must use this document *at your own risk* or *not use it at all*.

1.3 Acknowledgements

JFreeChart contains code and ideas from many people. At the risk of missing someone out, I would like to thank the following people for contributing to the project:

> Eric Alexander, Richard Atkinson, David Basten, David Berry, Chris Boek, Zoheb Borbora, Anthony Boulestreau, Jeremy Bowman, Daniel Bridenbecker, Nicolas Brodu, Jody Brownell, David Browning, Brian Cabana, Søren Caspersen, Chuanhao Chiu, Brian Cole, Pascal Collet, Martin Cordova, Paolo Cova, Michael Duffy, Don Elliott, Rune Fausk, Jonathan Gabbai, Serge V. Grachov, Daniel Gredler, Hans-Jurgen Greiner, Joao Guilherme Del Valle, Nick Guenther, Aiman Han, Cameron Hayne, Jon Iles, Wolfgang Irler, Sergei Ivanov, Adrian Joubert, Darren Jung, Xun Kang, Bill Kelemen, Norbert Kiesel, Gideon Krause, Pierre-Marie Le Biot, Arnaud Lelievre, Wolfgang Lenhard, David Li, Yan Liu, Tin Luu, Craig MacFarlane, Achilleus Mantzios, Thomas Meier, Aaron Metzger, Jim Moore, Jonathan Nash, Barak Naveh, David M. O'Donnell, Krzysztof Paz, Tomer Peretz, Xavier Poinsard, Andrzej Porebski, Luke Quinane, Viktor Rajewski, Eduardo Ramalho, Michael Rauch, Cameron Riley, Klaus Rheinwald, Dan Rivett, Scott Sams, Michel Santos, Thierry Saura, Andreas Schneider, Jean-Luc Schwab, Bryan Scott, Tobias Self, Mofeed Shahin, Pady Srinivasan, Greg Steckman, Roger Studner, Gerald Struck, Irv Thomae, Eric Thomas, Rich Unger, Daniel van Enckevort, Laurence Vanhelsuwé, Sylvain Vieujot, Jelai Wang, Mark Watson, Alex Weber, Richard West, Matthew Wright, Benoit Xhenseval, Christian W. Zuckschwerdt, Hari and Sam (oldman).

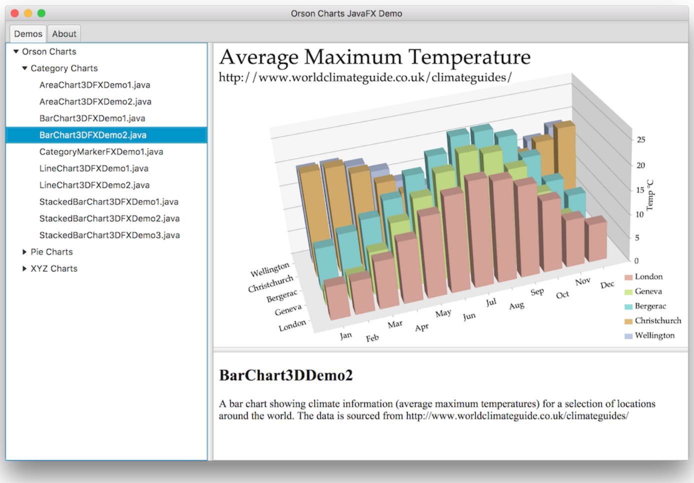
1.4 Comments and Suggestions

If you have any comments or suggestions regarding this document, please send e-mail to:

```
david.gilbert@jfree.org
```

1.5 Orson Charts

David Gilbert is also the author of a 3D chart library, Orson Charts, that provides an excellent complement to the 2D charts of JFreeChart.



BarChart3DDemo2

A bar chart showing climate information (average maximum temperatures) for a selection of locations around the world. The data is sourced from http://www.worldclimateguide.co.uk/climateguides/

Figure 1.2: Orson Charts 3D

Orson Charts features:

- multiple chart types: pie charts, bar charts (regular and stacked), line charts, area charts and scatter plots;
- a built-in lightweight 3D rendering engine based on Java2D (no OpenGL or other dependencies, therefore easy deployment);
- a mouse-enabled chart viewer provides 360 degree rotation and zooming for precise end-user view control;
- flexible data sources;
- auto-adaptive axis labeling;
- support for PDF, SVG and PNG export of charts for reporting;
- a clean and well-documented API with a high degree of chart configurability.

To find out more, please visit:

```
https://github.com/jfree/orson-charts
```

Chapter 2

JFreeChart Samples

2.1 Overview

This section shows some sample charts created using JFreeChart, to give an overview of the range of charts that JFreeChart can generate. For other examples, please run the demo application that you can find on the JFreeChart samples page:

```
http://www.jfree.org/jfreechart/samples.html
```

After downloading the jar file, you can run it with the following command:

```
java -jar jfreechart-demo-1.5.6-with-dependencies.jar
```

The complete source code for the demo application is available with certain sponsorship tiers.[1]

2.2 Pie Charts

JFreeChart can create *pie charts* using any data that conforms to the `PieDataset` interface. Figure 2.1 shows a simple pie chart.

[1]See `https://github.com/sponsors/jfree` for details.

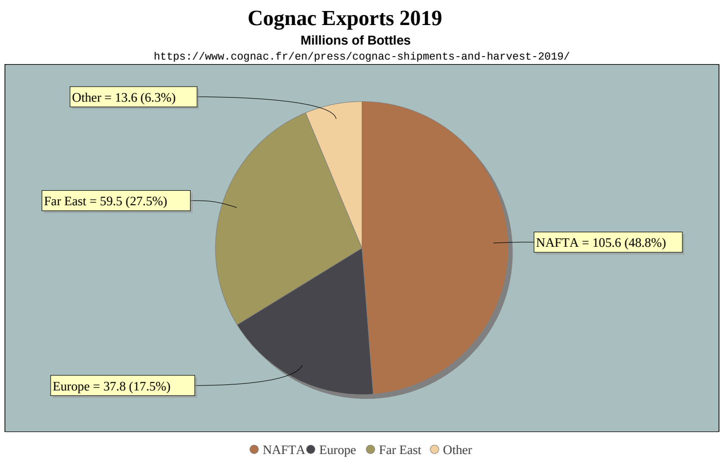


Figure 2.1: A simple pie chart (see `PieChartDemo1.java`*)*

Individual pie sections can be “exploded”, as shown in figure 2.2.

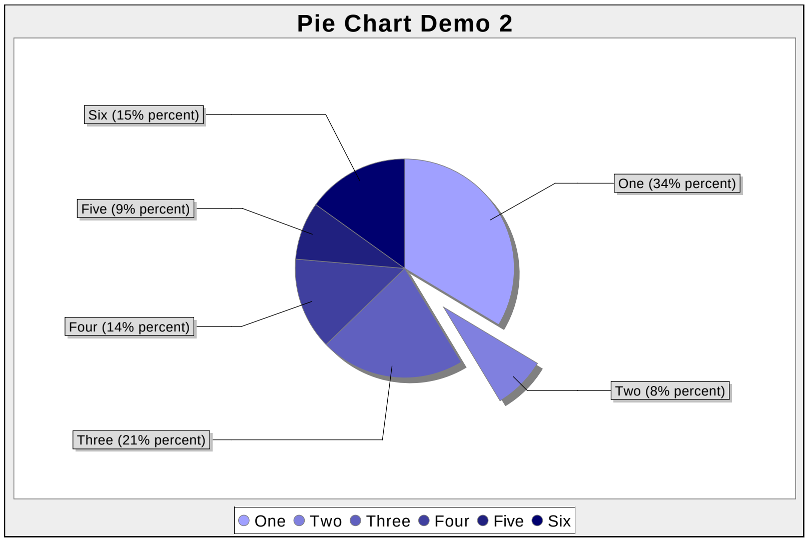


Figure 2.2: A pie chart with an “exploded” section (see `PieChartDemo2.java`*)*

2.3 Bar Charts

A range of bar charts can be created with JFreeChart, using any data that conforms to the `CategoryDataset` interface. Figure 2.3 shows a bar chart with a vertical orientation.

Another variation, the *waterfall chart*, is shown in figure 2.4.

Bar charts can also be generated from time series data—for example, see figure 2.5:

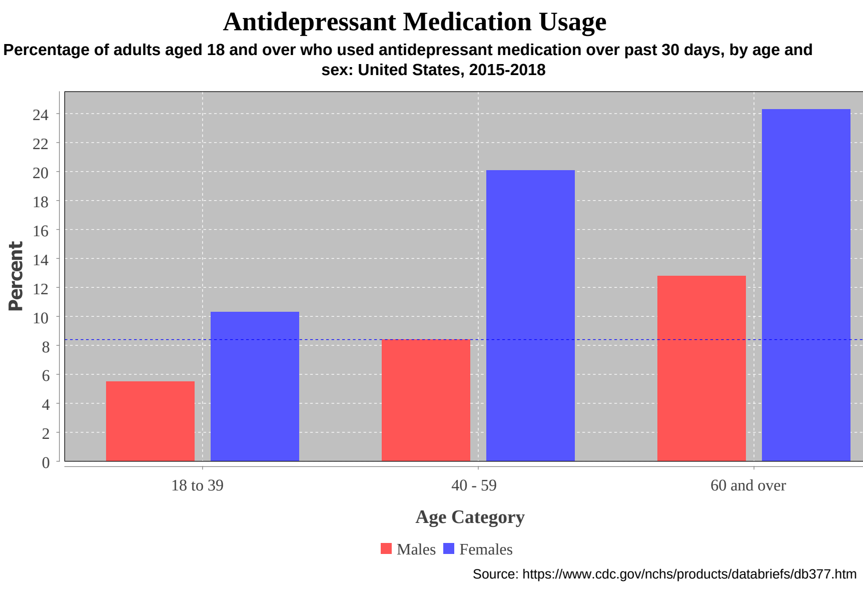


Figure 2.3: A vertical bar chart (see `BarChartDemo1.java`*)*

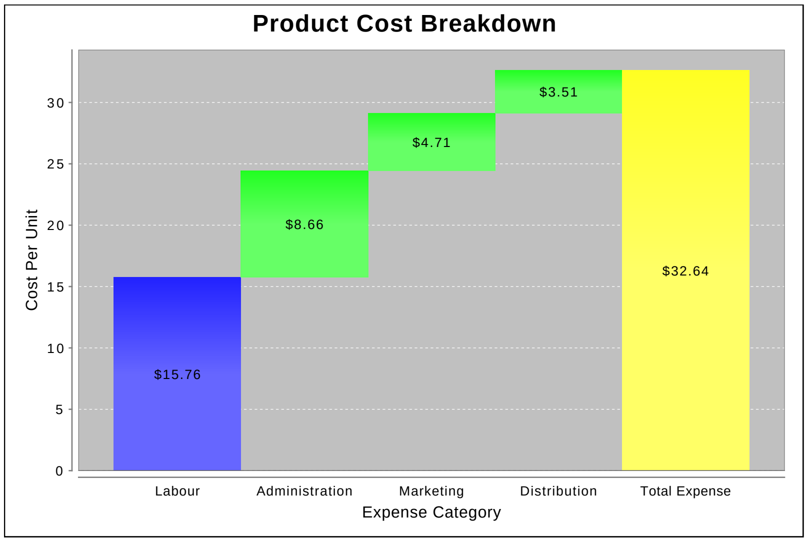


Figure 2.4: A waterfall chart (see `WaterfallChartDemo1.java`*)*

2.4 Line Chart

The *line chart* can be generated using the same `CategoryDataset` that is used for the bar charts—figure 2.6 shows an example.

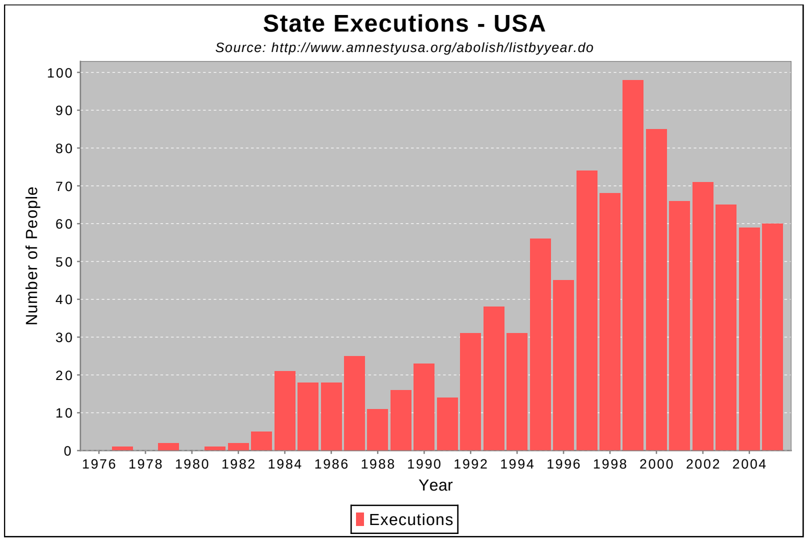


Figure 2.5: An XY bar chart (see `XYBarChartDemo1.java`*)*

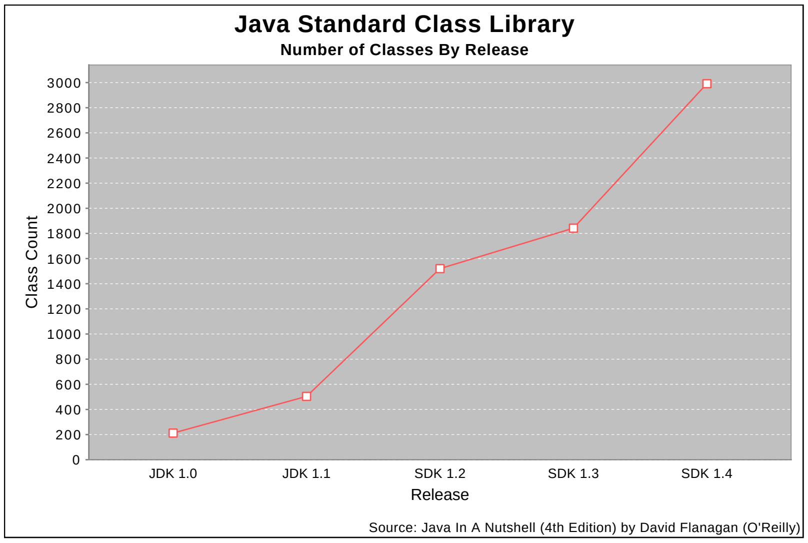


Figure 2.6: A line chart (see `LineChartDemo1.java`*)*

2.5 XY Plots

A third type of dataset, the `XYDataset`, is used to generate a range of chart types.

The standard *XY plot* has numerical x and y axes. By default, lines are drawn between each data point—see figure 2.7.

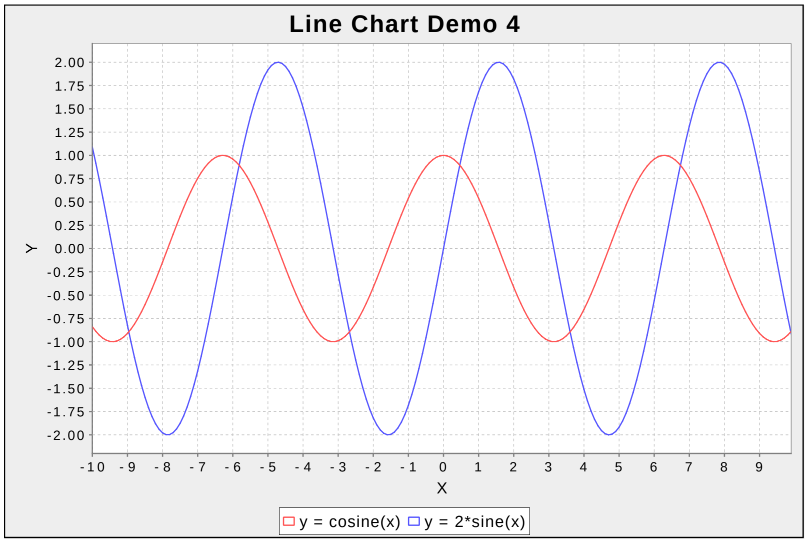


Figure 2.7: A line chart (see `LineChartDemo4.java`*)*

Scatter plots can be drawn by drawing a shape at each data point, rather than connecting the points with lines—an example is shown in figure 2.8.

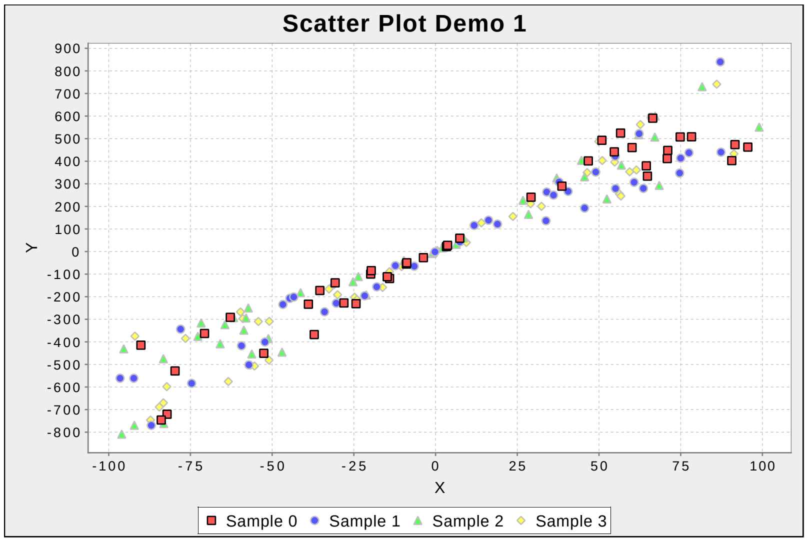


Figure 2.8: A scatter plot (see `ScatterPlotDemo1.java`*)*

2.6 Time Series Charts

JFreeChart supports *time series charts*, as shown in figure 2.9.

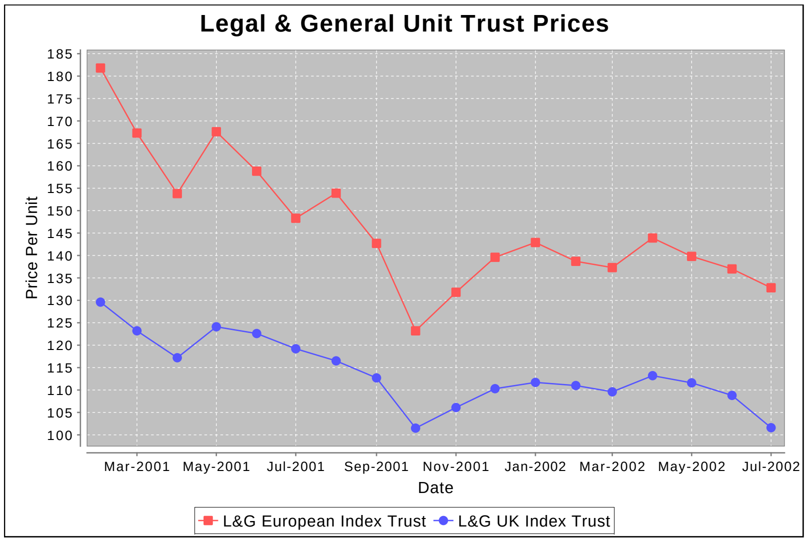


Figure 2.9: A time series chart (see `TimeSeriesDemo1.java`*)*

It is straightforward to add a moving average line to a time series chart—see figure 2.10 for an example.

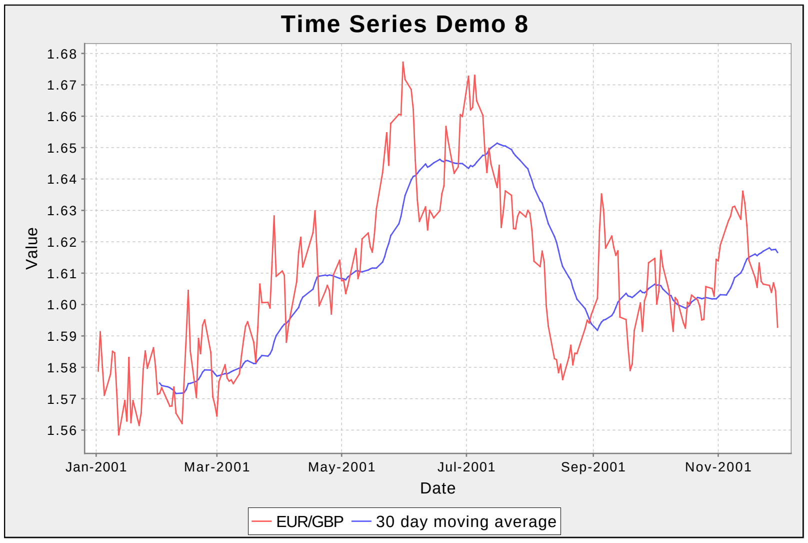


Figure 2.10: A time series chart with a moving average (see `TimeSeriesDemo8.java`*)*

Using an OHLCDataset (an extension of XYDataset) you can display *high-low-open-close* data, see figure 2.11 for an example.

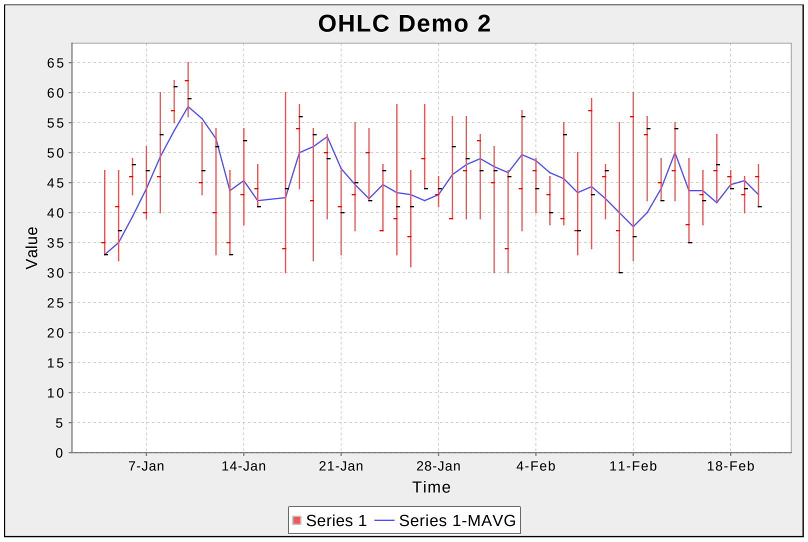


Figure 2.11: A high-low-open-close chart (see HighLowChartDemo2.java*)*

2.7 Histograms

Histograms can be generated using an IntervalXYDataset (another extension of XYDataset), see figure 2.12 for an example.

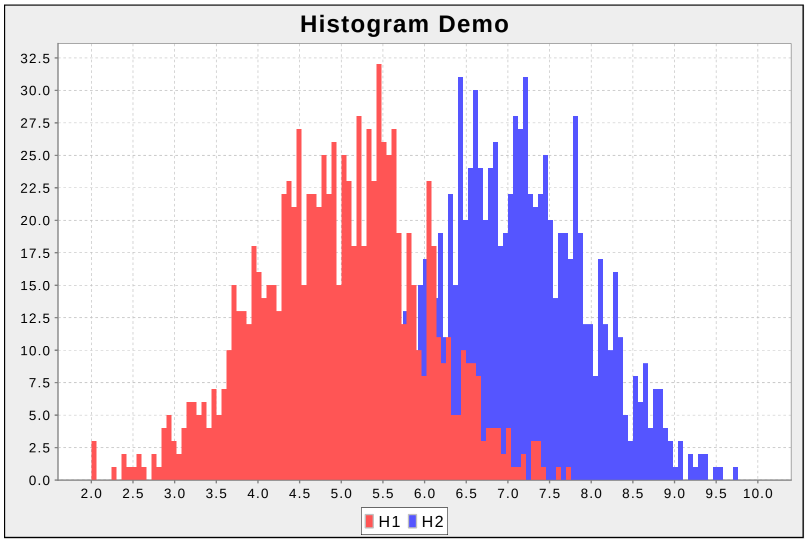


Figure 2.12: A histogram (see HistogramDemo1.java*)*

2.8 Area Charts

You can generate an *area chart* for data in a `CategoryDataset` or an `XYDataset`. Figure 2.13 shows an example.

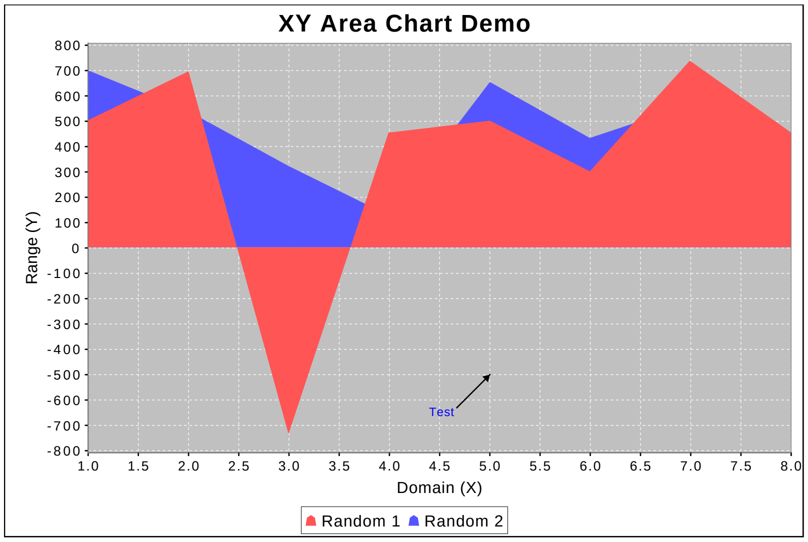


Figure 2.13: An area chart (see `XYAreaChartDemo1.java`*)*

JFreeChart also supports the creation of *stacked area charts* as shown in figure 2.14.

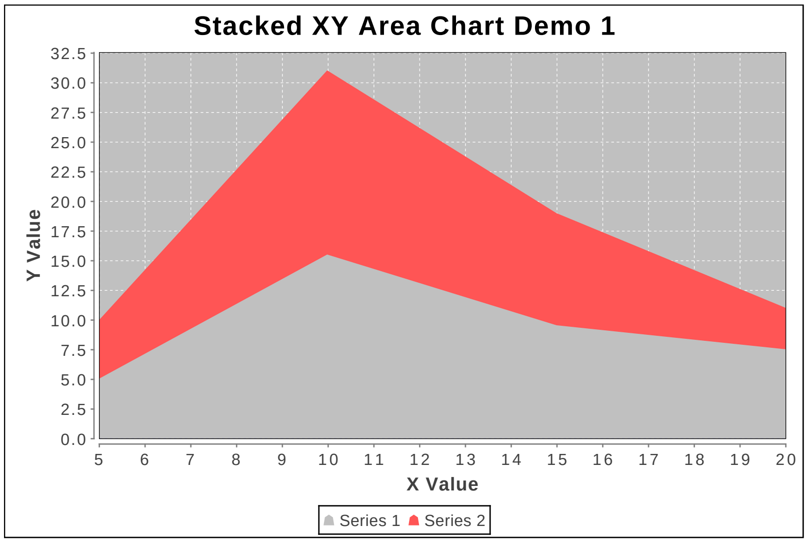


Figure 2.14: A stacked area chart (see `StackedXYAreaChartDemo1.java`*)*

2.9 Difference Chart

A *difference chart* highlights the difference between two series (see figure 2.15).

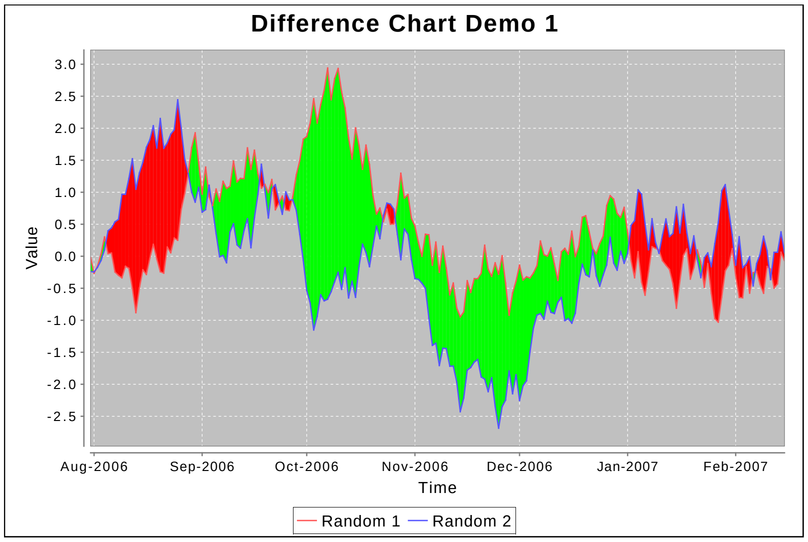


Figure 2.15: A difference chart (see `DifferenceChartDemo1.java`*)*

A second example, shown in figure 2.16 shows how a date axis can be used for the range values.

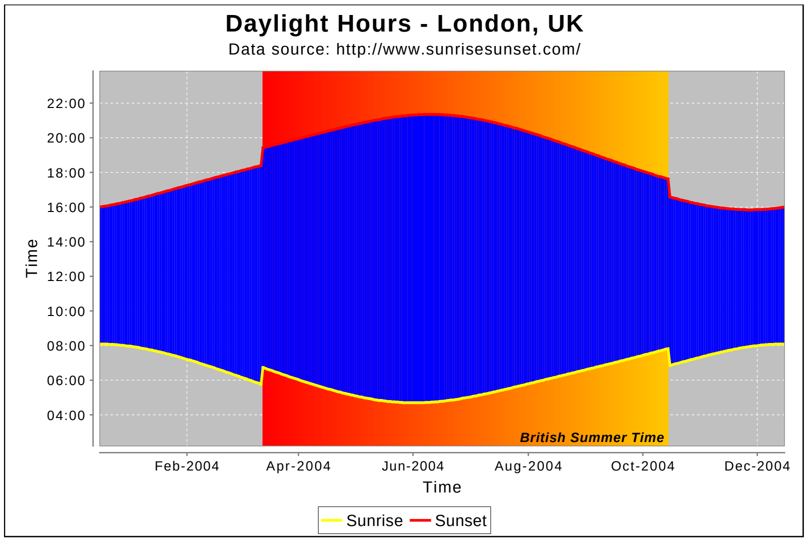


Figure 2.16: A difference chart with times on the range axis (see `DifferenceChartDemo2.java`*)*

2.10 Step Chart

A *step chart* displays numerical data as a sequence of “steps”—an example is shown in figure 2.17.

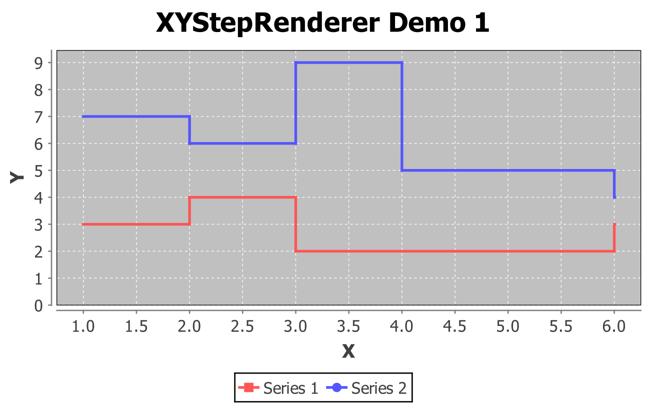


Figure 2.17: A step chart (see `XYStepRendererDemo1.java`*)*

Step charts are generated from data in an `XYDataset`.

2.11 Gantt Chart

Gantt charts can be generated using data from an `IntervalCategoryDataset`, as shown in figure 2.18.

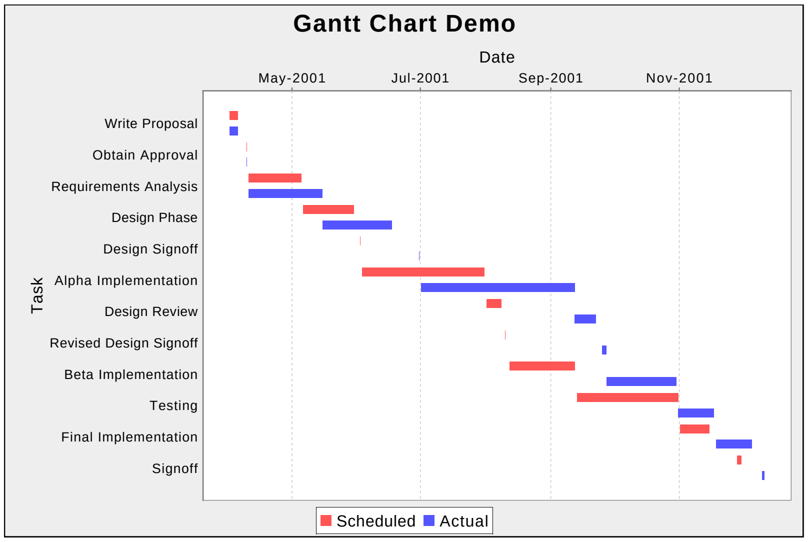


Figure 2.18: A Gantt chart (see `GanttChartDemo1.java`*)*

Another example, showing subtasks and progress indicators, is shown in figure 2.19.

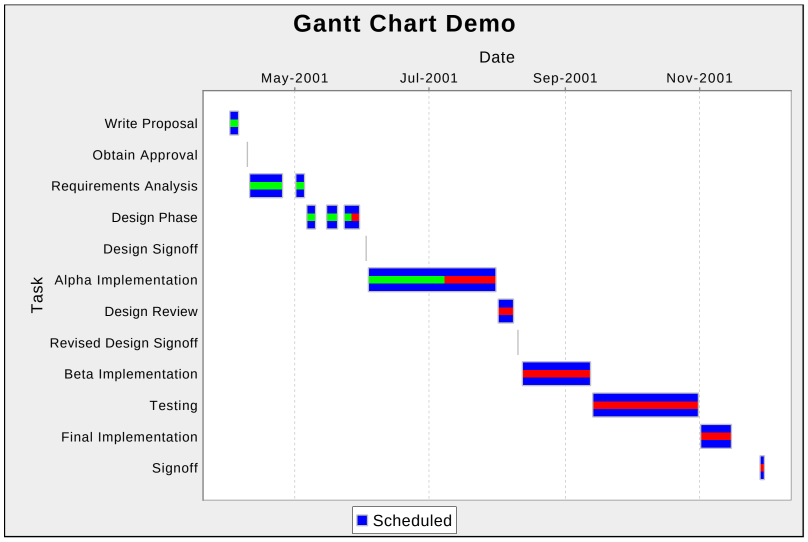


Figure 2.19: A Gantt chart with progress indicators (see `GanttChartDemo2.java`*)*

2.12 Multiple Axis Charts

JFreeChart has support for charts with multiple axes. Figure 2.20 shows a *price-volume chart* that demonstrates this feature.

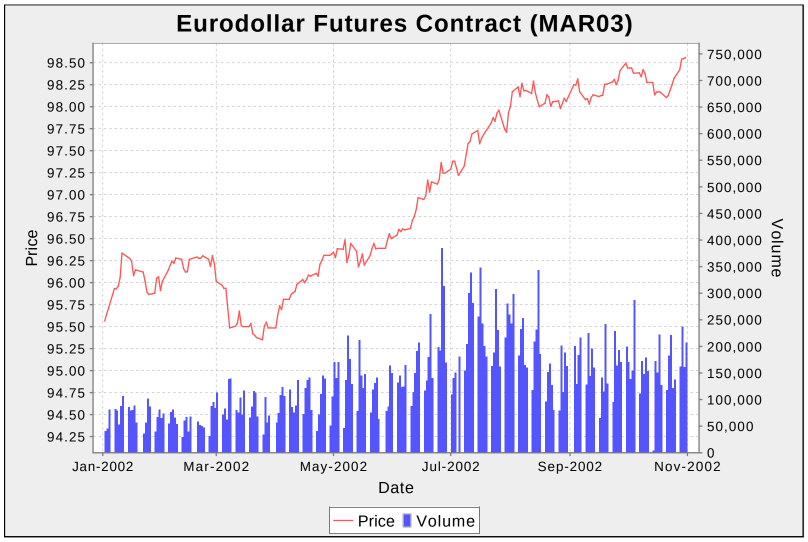


Figure 2.20: A price-volume chart (see `PriceVolumeDemo1.java`*)*

This feature is supported by the `CategoryPlot` and `XYPlot` classes. Figure 2.21 shows an example with four range axes.

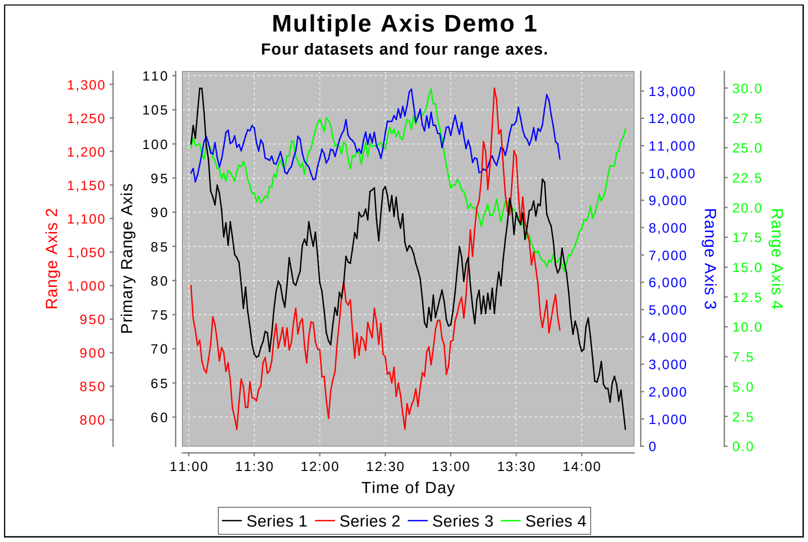


Figure 2.21: A chart with multiple axes (see `MultipleAxisDemo1.java`*)*

2.13 Combined and Overlaid Charts

JFreeChart supports combined and overlaid charts. Figure 2.22 shows a line chart overlaid on top of a bar chart.

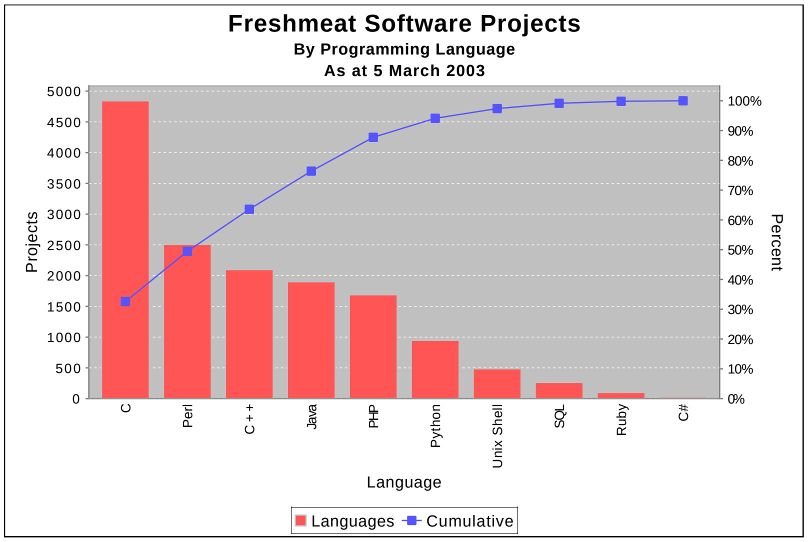


Figure 2.22: An overlaid chart (see `ParetoChartDemo1.java`*)*

It is possible to combine several charts that share a common domain axis, as shown in figure 2.23.

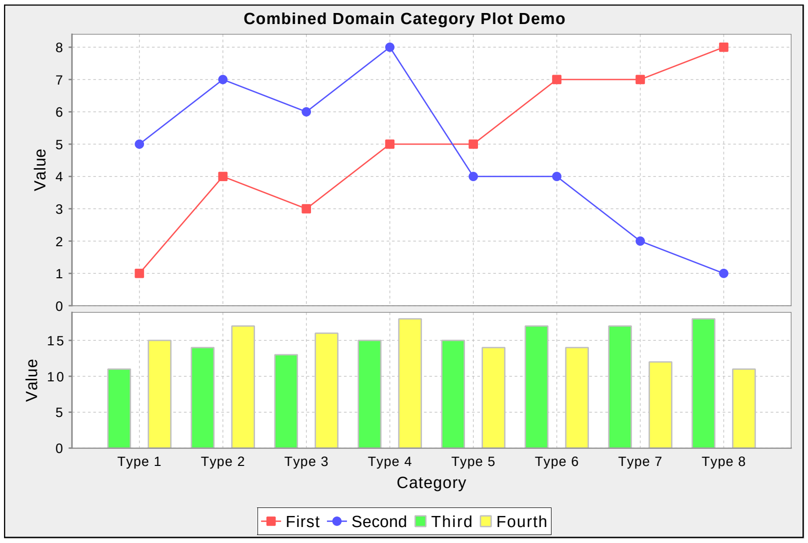


Figure 2.23: A chart with a combined domain (see `CombinedCategoryPlotDemo1.java`*)*

In a similar way, JFreeChart can combine several charts that share a common range axis, see figure 2.24.

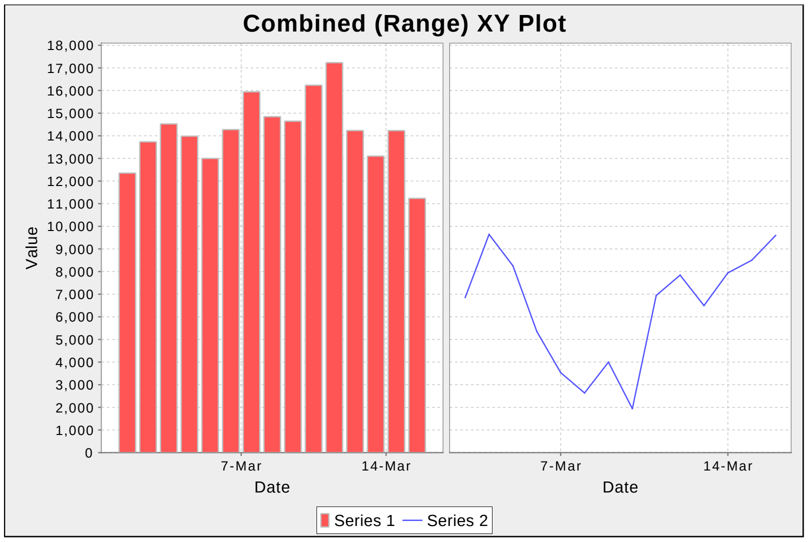


Figure 2.24: A chart with a combined range (see `CombinedXYPlotDemo2.java`*)*

Chapter 3

The JFreeChart Developer Guide

3.1 Overview

The *JFreeChart Developer Guide* provides extensive documentation for the JFreeChart Class Library. Written by David Gilbert, the principal author of JFreeChart, the guide contains tutorials and reference information that will help you to get the best out of JFreeChart. In addition, the complete source code for the JFreeChart demo application is available for download with the guide.

3.2 The Guide

The *JFreeChart Developer Guide* is not free—it is supplied to sponsors as a means of raising funds for the JFreeChart project. If you would like to sponsor the project, please visit the following URL:

```
https://github.com/sponsors/jfree
```

The document is made available via `HTTP` download in Acrobat PDF format.

Please note that we do NOT ship physical copies of the document.

Note that updates to the JFreeChart Developer Guide are made available free of charge for at least 1 year after purchase.

3.3 Demo Application Source Code

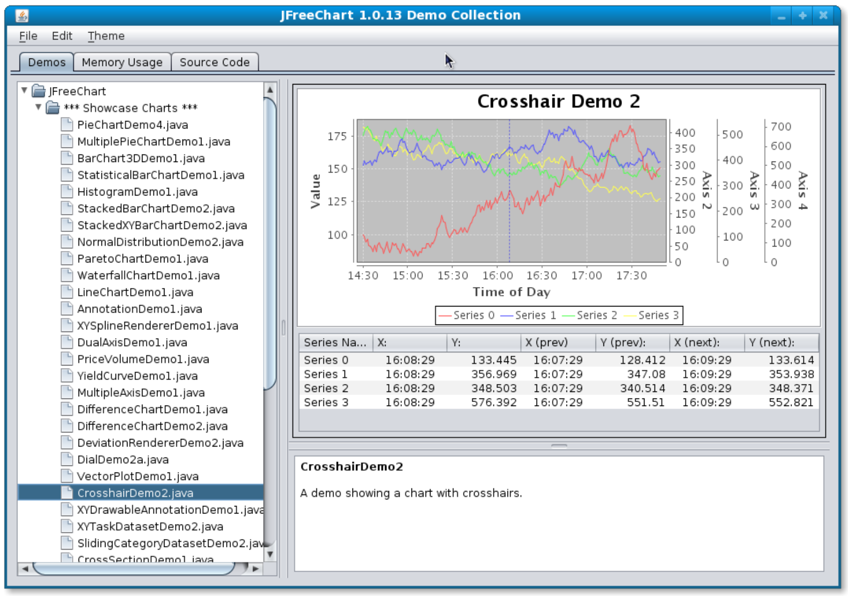
The source code for the demo application included in the JFreeChart distribution is available to download with the JFreeChart Developer Guide.

In addition, there is:

- a servlet demo, with charts embedded in an HTML page;
- several JDBC demos, where charts are generated using data from a relational database;
- demos showing how to capture chart mouse events;

The servlet and JDBC demos are described in the JFreeChart Developer Guide, including all the steps required for configuration.[1]

[1] Using Tomcat for the servlet demo and PostgreSQL for the JDBC demos.



Series Na...	X:	Y:	X (prev)	Y (prev):	X (next):	Y (next):
Series 0	16:08:29	133.445	16:07:29	128.412	16:09:29	133.614
Series 1	16:08:29	356.969	16:07:29	347.08	16:09:29	353.938
Series 2	16:08:29	348.503	16:07:29	340.514	16:09:29	348.371
Series 3	16:08:29	576.392	16:07:29	551.51	16:09:29	552.821

CrosshairDemo2

A demo showing a chart with crosshairs.

Figure 3.1: The JFreeChart Demo Collection

Appendix A

Maven and JFreeChart

A.1 Overview

JFreeChart is a Maven-driven project. Maven is used to manage all the tasks involved in building the JFreeChart jar file that your application depends upon. You can—and maybe already do—use Maven to manage your own projects. Maven is a solid choice and, for Java developers, knowledge of the tool is indispensible. This appendix provides useful information about Maven and JFreeChart, particularly for developers that have little or no experience with it.

A.2 What is Maven?

Maven is a tool for building Java projects and handling dependency management for those projects. Maven has capabilities beyond that, but for our purposes, this is enough. We are happy to stick to a minimal set of Maven's core features, and delve deeper only as required.

Projects that use Maven, JFreeChart included, have a project file (`pom.xml`) that describes the key characteristics of the project. This includes its dependencies and other important information that is used to:

- compile the source files;
- run unit tests;
- create a versioned jar file;
- generate JavaDoc documentation files.

...and a number of other useful things, such as publishing the project to the Maven Central Repository. You can do most of these things without Maven, but it takes care of many details that make your work as a developer easier. You can find out the latest information about Maven here:

```
https://maven.apache.org/
```

Maven has been successful in the Java space for many years and is well supported by the major Java integrated development environments.

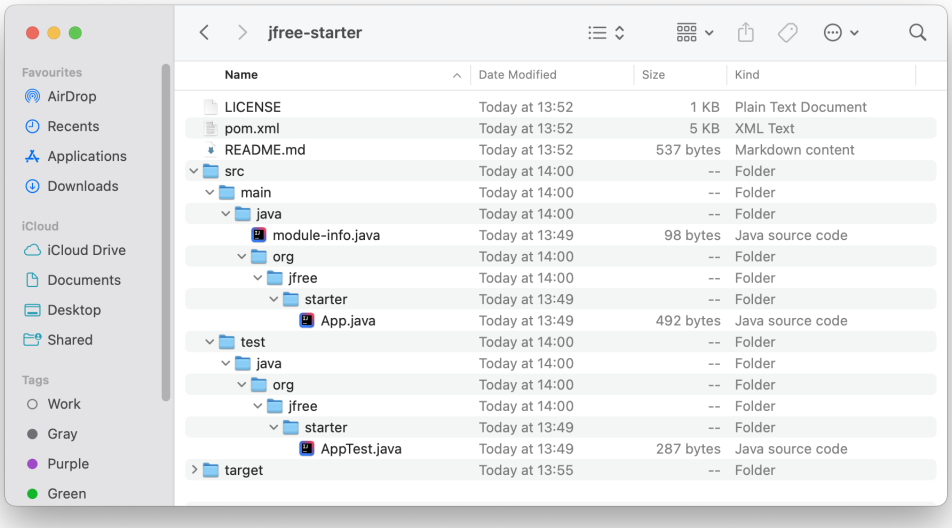
A.3 Starting a Java Project with Maven

To make it easier to get started with Maven, we present here a minimal Maven setup that you can use as a starting point for your own projects. The project will create a modular Java application and it will target Java 21 as this is the long term support ("LTS") edition of Java. You can, of course, use a later version of Java (the current release is Java 24) to build this project.

Maven follows a "convention over configuration" philosophy, and expects you to provide a standard layout for your project files. Typically you will create your project within a dedicated directory on your file system, and within this directory you should have:

- a project file describing the project (`pom.xml`) at the root level of the project directory;
- source files in `src/main/java`;
- test files in `src/test/java`.

This file structure is illustrated in figure A.1, with the addition of three files that are specific to our starter project: `module-info.java`, `App.java` and `AppTest.java`.



Name	Date Modified	Size	Kind
LICENSE	Today at 13:52	1 KB	Plain Text Document
pom.xml	Today at 13:52	5 KB	XML Text
README.md	Today at 13:52	537 bytes	Markdown content
src	Today at 14:00	--	Folder
main	Today at 14:00	--	Folder
java	Today at 14:00	--	Folder
module-info.java	Today at 13:49	98 bytes	Java source code
org	Today at 14:00	--	Folder
jfree	Today at 14:00	--	Folder
starter	Today at 13:49	--	Folder
App.java	Today at 13:49	492 bytes	Java source code
test	Today at 14:00	--	Folder
java	Today at 14:00	--	Folder
org	Today at 14:00	--	Folder
jfree	Today at 14:00	--	Folder
starter	Today at 13:49	--	Folder
AppTest.java	Today at 13:49	287 bytes	Java source code
target	Today at 13:55	--	Folder

Figure A.1: Maven File Structure

You can set this up manually by following the instructions below or, if you have an account at GitHub, you could take a shortcut and just clone the repository at:

```
https://github.com/jfree/jfree-starter
```

For those taking the manual path, create a working directory to store your project and then work through each of the following sections. If you cloned the repo at GitHub, then you should still read through the following sections for an explanation of the project content.

A.3.1 Maven Project File

Create a new text file with the name pom.xml. This is the Maven project file that will define the characteristics of your project. Use a text editor to insert the content below into the file, and save it in the working directory you created previously. If you are not familiar with Maven, this looks complicated—just ignore the complexity for now, the pieces will fall into place later.[1]

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>org.jfree</groupId>
    <artifactId>starter</artifactId>
    <version>1.0</version>
    <packaging>jar</packaging>

    <name>JFree Starter</name>
    <url>https://github.com/jfree/jfree-starter</url>

    <description>
      A minimal Maven-driven Java project targeting Java 21 or later.  You can use this project as a starting point.
    </description>

    <properties>
        <project.source.level>21</project.source.level>
        <project.target.level>21</project.target.level>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.jfree</groupId>
            <artifactId>jfreechart</artifactId>
            <version>1.5.6</version>
        </dependency>

        <!-- test dependencies -->
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.12.2</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-engine</artifactId>
            <version>5.12.2</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>

        <!-- specify the Maven plugins to use - supply config and control the versions. -->
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-clean-plugin</artifactId>
                <version>3.4.1</version>
            </plugin>
```

[1]It can take a long time, don't be discouraged by that.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-resources-plugin</artifactId>
    <version>3.3.1</version>
    <configuration>
        <encoding>${project.build.sourceEncoding}</encoding>
    </configuration>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.14.0</version>
    <configuration>
        <source>${project.source.level}</source>
        <target>${project.target.level}</target>
        <encoding>${project.build.sourceEncoding}</encoding>
        <compilerArgument>-Xlint:unchecked</compilerArgument>
        <showWarnings>false</showWarnings>
        <showDeprecation>true</showDeprecation>
    </configuration>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-dependency-plugin</artifactId>
    <version>3.7.0</version>
    <executions>
        <execution>
            <id>copy-dependencies</id>
            <phase>package</phase>
            <goals>
                <goal>copy-dependencies</goal>
            </goals>
        </execution>
    </executions>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>3.4.2</version>
    <configuration>
        <archive>
            <manifest>
                <mainClass>org.jfree.starter.App</mainClass>
            </manifest>
        </archive>
    </configuration>
</plugin>

<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-javadoc-plugin</artifactId>
    <version>3.11.2</version>
    <configuration>
        <linksource>true</linksource>
    </configuration>
    <executions>
        <execution>
            <id>attach-javadoc</id>
            <goals>
```

```
                            <goal>jar</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>3.5.3</version>
            </plugin>
            <plugin>
                <artifactId>maven-failsafe-plugin</artifactId>
                <version>3.5.3</version>
            </plugin>
        </plugins>
    </build>

</project>
```

Let's do a quick review of the important parts of this file. Near the top of the file, you will see the following elements that declare the type and identity of the main artifact that will be produced by your project:

```
<groupId>org.jfree</groupId>
<artifactId>starter</artifactId>
<version>1.0</version>
<packaging>jar</packaging>
```

The `<packaging>` element tells Maven that you want it to create a jar file from the project sources. This is very common for Java projects. There are other options, but we don't need to consider them here since our goal is to generate a jar file for our application.

The other three elements (`<groupId>`, `<artifactId>` and `<version>`) declare a unique identifier for the artifact (jar file) that will be produced by our project. The `<groupId>` provides a mechanism for grouping artifacts together—it should be unique and, for your own projects, you should change it from `org.jfree` to something else—this becomes important later if you want to make your project available publicly. The `<artifactId>` is the identifier for your artifact, you can change this to anything you want—it should be unique within the artifact group. The `<version>` element is relatively straightforward, you can enter any version number you wish to associate with your project.[2]

The `<properties>` section declares some properties that will be used by your project, for now just note that this is where we are specifying the target version for the Java compiler.

Further down the `pom.xml` file, you will see a `<dependencies>` section. Here we are declaring other artifacts that our project will depend upon. Maven can automatically fetch these dependencies. Our file adds a dependency on JFreeChart 1.5.6, as an example. It also adds dependencies for a couple of JUnit 5 artifacts, allowing our project to include and run JUnit tests. Notice that the JUnit dependencies specify the `<scope>` as `test`, this tells Maven that the dependencies are only required for testing and not needed for the application itself.

The `<plugins>` section is used to configure the Maven plugins that our project uses. You don't need to declare every plugin used by your project, but we've added most of them as it allows us to control the exact version of each plugin that is being used. Also, if you have some custom configuration to provide, this is where it is done. For example, for the Maven Jar plugin we

[2]For non-released versions, there is a convention in Maven to add the suffix `-SNAPSHOT` to the version number. Maven has special handling for snapshot builds, but we will not focus on that here.

add configuration that will set the class `org.jfree.starter.App` to be the main class in the jar file created for our project.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-jar-plugin</artifactId>
    <version>3.2.0</version>
    <configuration>
        <archive>
            <manifest>
                <mainClass>org.jfree.starter.App</mainClass>
            </manifest>
        </archive>
    </configuration>
</plugin>
```

A.3.2 Maven Directory Structure

Next, create the directory structure that Maven expects for the source and test files for your application. Create a directory structure that looks like the one shown earlier in figure A.1—skip the files `module-info.java`, `App.java` and `AppTest.java` as we'll add those afterwards.

A.3.3 Java Module Info

Our project is a modular Java application, so we need to add a `module-info.java` file at the root of the Java package hierarchy. Create the following file within the `src/main/java` directory:

```
/**
 * Module for the starter app.
 */
module JFreeStarter {
    requires org.jfree.jfreechart;
}
```

This very simple module file declares that our module is called `JFreeStarter` (you can change this name for your own project) and that it requires the `org.jfree.jfreechart` module. When you develop your own application, this module file will grow as you declare additional dependencies to other modules.

A.3.4 Java Application

Now add a Java source file declaring the main class for the application. In the `src/main/java/org/jfree/starter` directory create a file `App.java` as follows:

```
package org.jfree.starter;

/**
 * Starter application.
 */
public class App {

    /**
     * Creates a new instance of the most awesome app.
     */
```

```
        public App() {
        }

        void run() {
            System.out.println("Hello World!");
        }

        /**
         * Entry point for the app.
         *
         * @param args  these are ignored here.
         */
        public static void main(String[] args) {
            App app = new App();
            app.run();
            System.exit(0);
        }

    }
```

For your own project you can change the name of the application class and also the Java package in which it resides (for example, you could change from `org.jfree.starter` to `com.mycompany.myapp`).

A.3.5 JUnit Tests

This part is optional.

A.3.6 Building the Project

To build the project, type the following command from the root of the project (the working directory where the `pom.xml` file is saved):

```
$ mvn clean verify
```

Maven will use the information from the `pom.xml` to download required dependencies (for example, JFreeChart and JUnit, plus required Maven plugins) and build the project. After the build completes, you will notice a new directory (`target`) has been created in your project. This contains the artifacts generated by the Maven build.

A.3.7 Running the Java Application

To run the Java application you built in the previous section, type the following command:

```
$ java --module-path target/starter-1.0.jar:target/dependency/jfreechart-1.5.6.jar -m JFreeStarter
```

You now have all of the pieces in place to build a working Java application. Now you are free to start adding features to this minimal project. Good luck!

Appendix B

Integrated Development Environments

B.1 Introduction

There are a number of integrated development environments (IDEs) that developers use when working on Java programs. In this section, we describe how to configure some popular IDEs to use JFreeChart.[1] Specifically, we'll cover:

- IntelliJ IDEA (version 2025.1)
- NetBeans (version 26)
- Eclipse (version 2020-12)

B.2 IntelliJ IDEA

B.2.1 Overview

IntelliJ IDEA is a very popular IDE for Java developers. You can download it from:

```
https://www.jetbrains.com/idea/
```

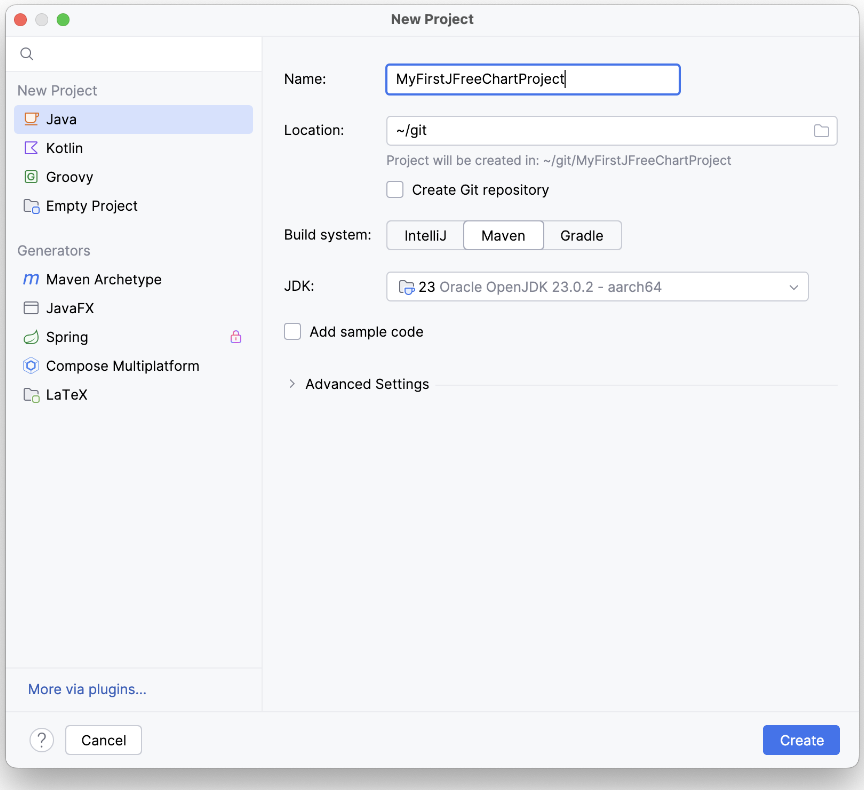
The instructions in this section have been prepared using version `2025.1`. If you are using a more recent version and something does not work as described, open an issue in the JFreeChart project at GitHub.

B.2.2 Creating an IntelliJ Project that uses JFreeChart

Follow these steps to start a new project in IntelliJ IDEA that uses JFreeChart:

1. Start IntelliJ IDEA and from the `File` menu select `New > Project...`. The dialog in figure B.1 will appear. Enter a project name, and select `Maven` for the build system. Click the `Create` button to proceed.

[1] Note that this section is concerned with *using* JFreeChart as a library. If you intend to *modify* the JFreeChart sources, you'll want to configure JFreeChart as a project within your IDE.



Name: MyFirstJFreeChartProject
Location: ~/git
Project will be created in: ~/git/MyFirstJFreeChartProject
Create Git repository
Build system: Maven
JDK: 23 Oracle OpenJDK 23.0.2 - aarch64
Add sample code

Figure B.1: New Project Dialog - Step 1.

2. Open the `pom.xml` file and add a dependencies section as follows:

```
<dependencies>
    <dependency>
        <groupId>org.jfree</groupId>
        <artifactId>jfreechart</artifactId>
        <version>1.5.6</version>
    </dependency>
</dependencies>
```

This adds the JFreeChart library as a dependency for the project. Maven will take care of fetching the specified version of JFreeChart when you build your application. Review the information about Maven in Appendix A if necessary.

3. Right-click on the `src/main/java` entry in the project tree and select `New -> Package`. Enter `org.jfree.starter` as the package name.

4. Right-click on the `org.jfree.starter` package that you just created, then select the "`New > Java Class`" menu item to create a new `App.java` source file in your project. Next, copy and paste the source code from section B.5 into this file.

5. Select the `Run Project` item from the `Run` menu, then watch as IntelliJ IDEA compiles and runs the application.

That's all there is to it!

B.3 NetBeans

B.3.1 Overview

NetBeans is a free IDE developed by the Apache Foundation. It was previously a product from Oracle (and before that, Sun Microsystems). You can download a free copy of NetBeans from:

`http://netbeans.apache.org/`

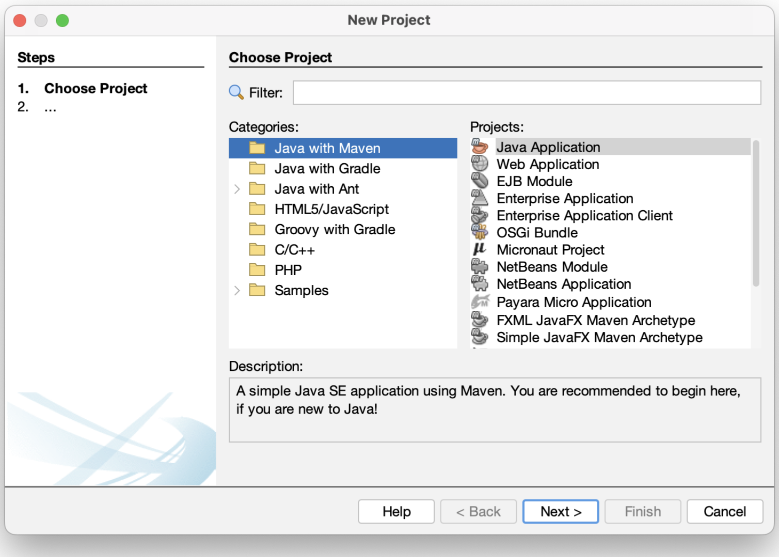
NetBeans has built in support for Maven, a build tool that is described in Appendix A. This makes it straightforward to include JFreeChart in your application, with NetBeans automatically handling features like code completion, Javadoc popups, stepping through the JFreeChart sources during debugging, and more.

The instructions in this section have been prepared using NetBeans version 26. If you are using a more recent version and something does not work as described, open an issue in the JFreeChart project at GitHub.

B.3.2 Creating a NetBeans Project that uses JFreeChart

Follow these steps to start a new project in NetBeans that uses JFreeChart:

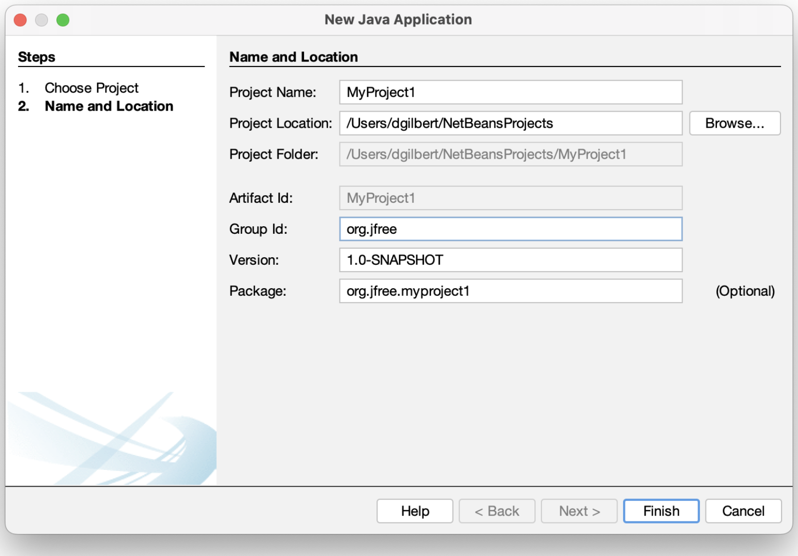
1. Start NetBeans and from the `File` menu select the `New Project...` item. The dialog in figure B.2 will appear. Select `Java Application` project from the `Java with Maven` category. Click the `Next` button to proceed.



A simple Java SE application using Maven. You are recommended to begin here, if you are new to Java!

Figure B.2: New Project Dialog - Step 1.

2. NetBeans will next request some attributes for your project, as shown in figure B.3. Enter the name and file-system location for your project. You can also modify the `<groupID>` and `<version>` that will be defined in the Maven `pom.xml` file that NetBeans will generate for you (if you are unsure, just take the default values–they can be changed easily later). Click the `Finish` button, then review the project structure that NetBeans has created for you—see figure B.4.



New Java Application

Steps
1. Choose Project
2. **Name and Location**

Name and Location

Project Name: MyProject1
Project Location: /Users/dgilbert/NetBeansProjects
Project Folder: /Users/dgilbert/NetBeansProjects/MyProject1
Artifact Id: MyProject1
Group Id: org.jfree
Version: 1.0-SNAPSHOT
Package: org.jfree.myproject1 (Optional)

Figure B.3: New Project Dialog - Application Details.

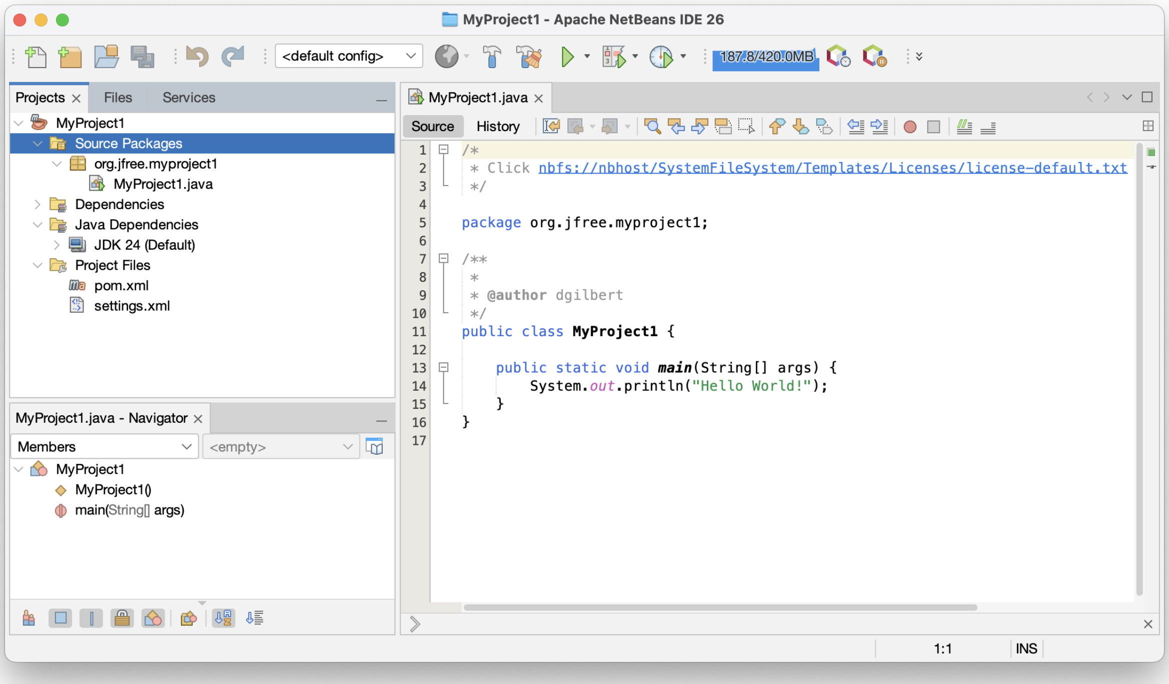
3. Open the `pom.xml` file and add a dependencies section as follows:

```
<dependencies>
    <dependency>
        <groupId>org.jfree</groupId>
        <artifactId>jfreechart</artifactId>
        <version>1.5.6</version>
    </dependency>
</dependencies>
```

This adds the JFreeChart library as a dependency for the project. Maven will take care of fetching the specified version of JFreeChart when you build your application. Review the information about Maven in Appendix A if necessary.

4. Right-click on the `Source Packages` item in the project tree, then select the `New > Java Package` menu item to create a new `org.jfree.starter` package in your project. See figure B.5.

5. Right-click on the package just created, then select the "`New > Java Class`" menu item to create a new `App.java` source file in your project. Next, copy and paste the source code from section B.5 into this file.



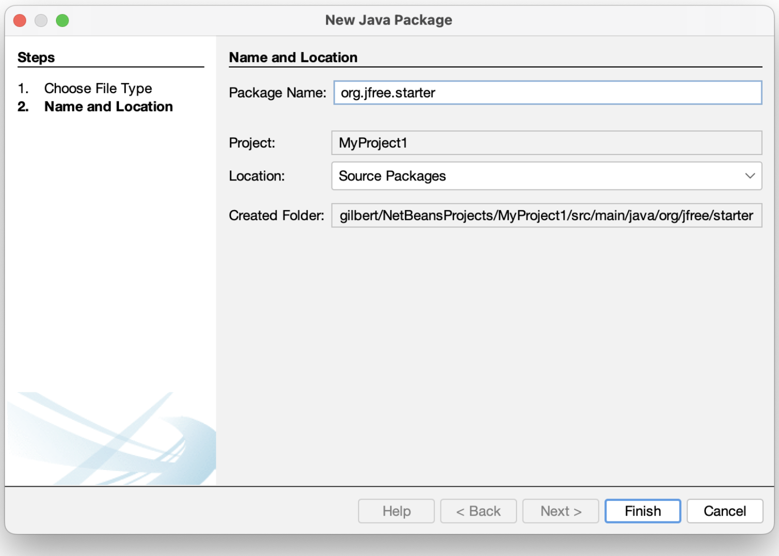
```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 */

package org.jfree.myproject1;

/**
 *
 * @author dgilbert
 */
public class MyProject1 {

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

Figure B.4: The project structure.

6. Edit the `pom.xml` file to set the `exec.mainClass` property to `org.jfree.starter.App`.

7. Select the `Run Project` item from the `Run` menu, then watch as NetBeans compiles and runs the application.

That's all there is to it!


New Java Package

Steps
1. Choose File Type
2. **Name and Location**

Name and Location

Package Name: org.jfree.starter

Project: MyProject1

Location: Source Packages

Created Folder: gilbert/NetBeansProjects/MyProject1/src/main/java/org/jfree/starter

Figure B.5: Create a new package.

B.4 Eclipse

B.4.1 Overview

Eclipse is a free IDE developed by the Eclipse Foundation (and, originally, IBM):

`http://www.eclipse.org/`

Eclipse has its own project format, but also features out-of-the-box support for Maven and we will use this for getting started.

B.4.2 Configuration Steps

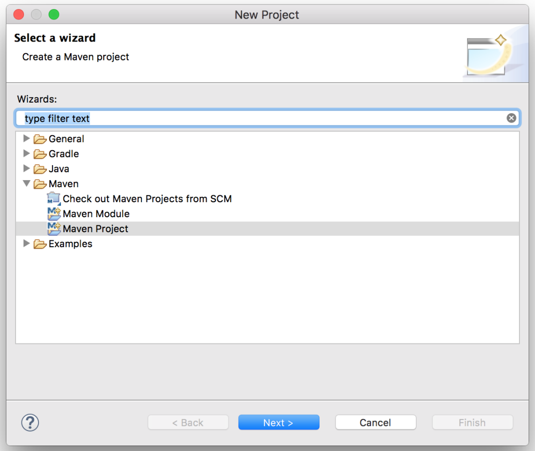
Follow these steps to create a new project in Eclipse that uses JFreeChart:

1. Start Eclipse and from the `File` menu select `"New > Project..."`. Choose Maven Project from the Maven folder—see figure B.6—then click the `Next` button.

2. In the next step, check the `"Create a simple project"` option—see figure B.7—then click the `Next` button.

3. In the next step, fill out the basic details for the project (`Group ID`, `Artifact ID`, `Name` and `Description`)—see figure B.8—then click the `Finish` button.

4. The project will now be created, you should see something similar to figure B.9. Open the `pom.xml` file and add a dependencies section as follows:



New Project

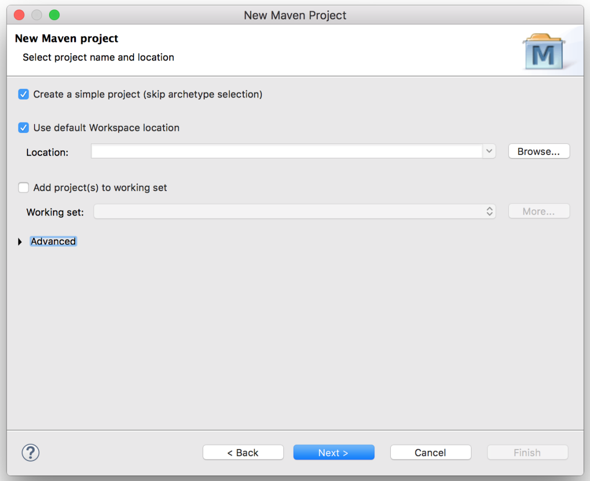
Select a wizard

Create a Maven project

Wizards:

type filter text

Figure B.6: Eclipse New Maven Project Dialog - Step 1.



New Maven Project

New Maven project

Select project name and location

- [✓] Create a simple project (skip archetype selection)
- [✓] Use default Workspace location

Location:

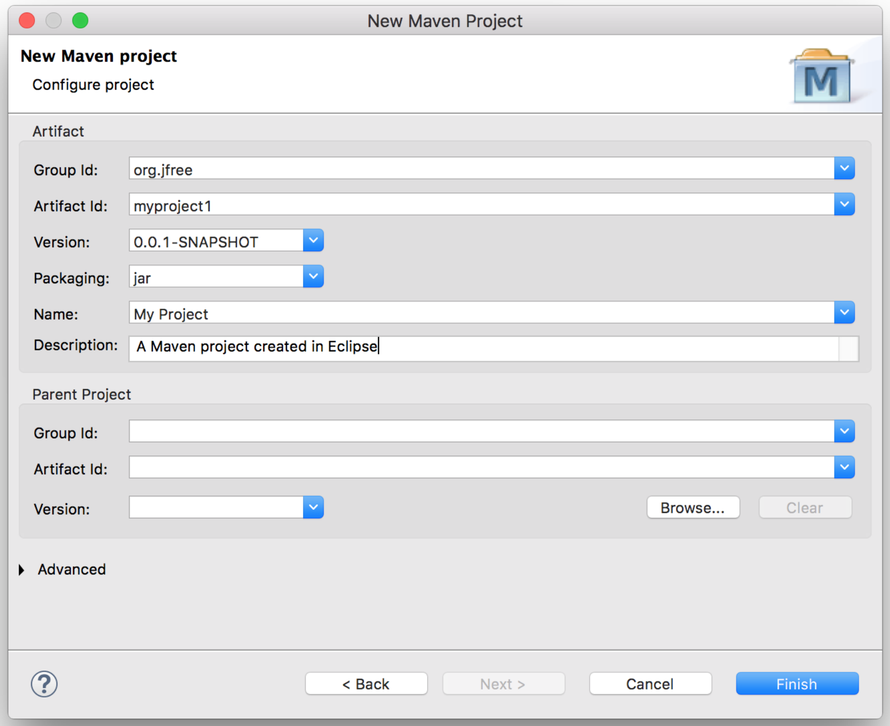
- [ ] Add project(s) to working set

Working set:

Figure B.7: Eclipse New Maven Project Dialog - Step 2.

```
<dependencies>
    <dependency>
        <groupId>org.jfree</groupId>
        <artifactId>jfreechart</artifactId>
        <version>1.5.6</version>
    </dependency>
</dependencies>
```

This adds the JFreeChart library as a dependency for the project. Maven will take



New Maven project
Configure project

Artifact

Group Id:	org.jfree
Artifact Id:	myproject1
Version:	0.0.1-SNAPSHOT
Packaging:	jar
Name:	My Project
Description:	A Maven project created in Eclipse

Parent Project

Group Id:	
Artifact Id:	
Version:	

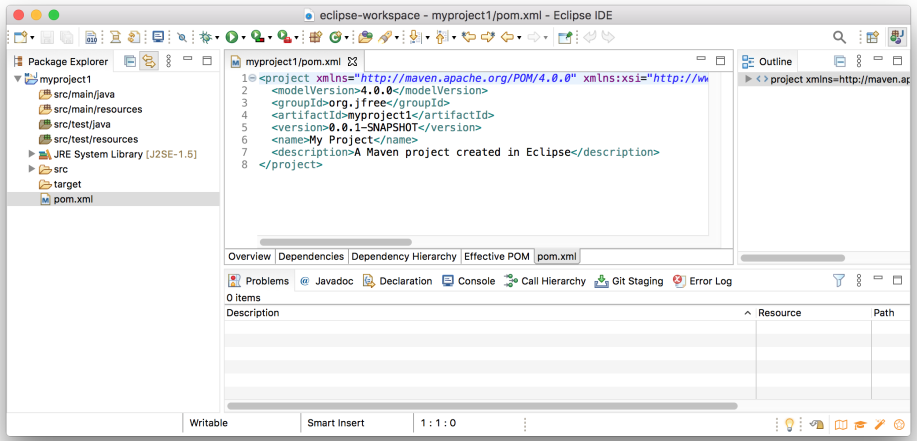
Figure B.8: Eclipse New Maven Project Dialog - Step 3.

care of fetching the specified version of JFreeChart when you build your application. Review the information about Maven in Appendix A if necessary.

5. Right-click on the package just created, then select the "`New > Java Class`" menu item to create a new `App.java` source file in your project. Next, copy and paste the source code from section B.5 into this file.

6. Select the `App.java` item from the `Project Explorer`, right click and select the `Run` menu item, then watch as Eclipse compiles and runs the application.

That's all there is to it!



```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://ww
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.jfree</groupId>
  <artifactId>myproject1</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>My Project</name>
  <description>A Maven project created in Eclipse</description>
</project>
```

Figure B.9: The project structure.

B.5 Sample Program Source Code

The following source code is referenced for each of the IDEs in this section.

```
package org.jfree.starter;

import java.awt.BasicStroke;
import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.Font;
import java.awt.geom.Ellipse2D;
import javax.swing.JFrame;
import static javax.swing.WindowConstants.EXIT_ON_CLOSE;
import org.jfree.chart.ChartFactory;
import org.jfree.chart.ChartPanel;
import org.jfree.chart.ChartUtils;
import org.jfree.chart.JFreeChart;
import org.jfree.chart.axis.NumberAxis;
import org.jfree.chart.plot.CategoryPlot;
import org.jfree.chart.renderer.category.LineAndShapeRenderer;
import org.jfree.chart.title.TextTitle;
import org.jfree.chart.ui.HorizontalAlignment;
import org.jfree.chart.ui.RectangleEdge;
import org.jfree.data.category.CategoryDataset;
import org.jfree.data.category.DefaultCategoryDataset;

/**
 * Simple line chart with JFreeChart and Java Swing.
 */
public class App extends JFrame {

    /**
     * Creates a new instance of the app.
     *
     * @param title  the frame title.
     */
    public App(String title) {
        setTitle(title);
        CategoryDataset dataset = createDataset();
        JFreeChart chart = createChart(dataset);
        ChartPanel panel = new ChartPanel(chart);
        panel.setPreferredSize(new Dimension(1024, 360));
        setLayout(new BorderLayout());
        getContentPane().add(panel, BorderLayout.CENTER);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
    }

    /**
     * Creates a sample dataset (hard-coded for the purpose of keeping the
     * demo self-contained - in practice you would normally read your data
     * from a file, database or other source).
     *
     * @return A sample dataset.
     */
    private CategoryDataset createDataset() {
        DefaultCategoryDataset dataset = new DefaultCategoryDataset();

        // data from Java in a Nutshell
        dataset.addValue(212, "Classes", "JDK 1.0");
        dataset.addValue(504, "Classes", "JDK 1.1");
        dataset.addValue(1520, "Classes", "JDK 1.2");
        dataset.addValue(1842, "Classes", "JDK 1.3");
        dataset.addValue(2991, "Classes", "JDK 1.4");
        dataset.addValue(3500, "Classes", "JDK 1.5");

      // from https://stackoverflow.com/questions/3112882/how-many-classes-are-there-in-java-standard-edition
        dataset.addValue(3793, "Classes", "JDK 1.6");
        dataset.addValue(4024, "Classes", "JDK 1.7");
        dataset.addValue(4240, "Classes", "JDK 8");
        dataset.addValue(6005, "Classes", "JDK 9");
        dataset.addValue(6002, "Classes", "JDK 10");
        dataset.addValue(4411, "Classes", "JDK 11");
        dataset.addValue(4433, "Classes", "JDK 12");
        dataset.addValue(4545, "Classes", "JDK 13");
        dataset.addValue(4569, "Classes", "JDK 14");
        return dataset;
    }

    /**
     * Creates a sample chart.
     *
     * @param dataset  a dataset.
     *
     * @return The chart.
     */
    private JFreeChart createChart(CategoryDataset dataset) {

        // create the chart...
        JFreeChart chart = ChartFactory.createLineChart(
            "Java Standard Class Library",   // chart title
            null,                            // domain axis label
            "Class Count",                   // range axis label
            dataset);
        chart.removeLegend();

        CategoryPlot plot = (CategoryPlot) chart.getPlot();
        plot.setRangePannable(true);
        plot.setRangeGridlinesVisible(false);

        // customise the range axis...
        NumberAxis rangeAxis = (NumberAxis) plot.getRangeAxis();
        rangeAxis.setStandardTickUnits(NumberAxis.createIntegerTickUnits());

        ChartUtils.applyCurrentTheme(chart);

        chart.addSubtitle(new TextTitle("Number of Classes By Release"));
        TextTitle source = new TextTitle(
                "Sources: https://stackoverflow.com/q/3112882 "
                + "and Java In A Nutshell (5th Edition) by David Flanagan (O'Reilly)");
        source.setFont(new Font("SansSerif", Font.PLAIN, 10));
        source.setPosition(RectangleEdge.BOTTOM);
        source.setHorizontalAlignment(HorizontalAlignment.RIGHT);
        chart.addSubtitle(source);

        // customise the renderer...
        LineAndShapeRenderer renderer = (LineAndShapeRenderer) plot.getRenderer();
        renderer.setDefaultShapesVisible(true);
        renderer.setDrawOutlines(true);
        renderer.setUseFillPaint(true);
        renderer.setDefaultFillPaint(Color.WHITE);
        renderer.setSeriesStroke(0, new BasicStroke(3.0f));
        renderer.setSeriesOutlineStroke(0, new BasicStroke(2.0f));
        renderer.setSeriesShape(0, new Ellipse2D.Double(-5.0, -5.0, 10.0, 10.0));
        return chart;
    }
    /**
     * Entry point for the app.
     *
     * @param args these are ignored here.
     */
    public static void main(String[] args) {
        App app = new App("JFreeChart Starter App");
        app.pack();
        app.setVisible(true);
    }
}
```

Appendix C

The GNU Lesser General Public Licence

C.1 Introduction

JFreeChart is licensed under the terms of the GNU Lesser General Public Licence (LGPL). The full text of this licence is reproduced in this appendix. You should read and understand this licence before using JFreeChart in your own projects.

If you are not familiar with the idea of *free software*, you can find out more at the Free Software Foundation's web site:

```
http://www.fsf.org
```

Please send e-mail to `dave@jfree.org` if you have any questions about the licensing of JFreeChart (but please read section C.3 first).

C.2 The Licence

The following licence has been used for the distribution of the JFreeChart class library:

receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

* a) The modified work must itself be a software library.

* b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

* c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

* d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

where the full notice is found.

```
<one line to give the library's name and a brief idea of what it does.>
Copyright (C) <year> <name of author>

This library is free software; you can redistribute it and/or modify it under the terms of the
GNU Lesser General Public License as published by the Free Software Foundation; either version
2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without
even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public License along with this library;
if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
02111-1307 USA
```

Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright interest in the library 'Frob' (a library for
tweaking knobs) written by James Random Hacker.

<signature of Ty Coon>, 1 April 1990
Ty Coon, President of Vice
```

That's all there is to it!

C.3 Frequently Asked Questions

C.3.1 Introduction

Some of the most frequently asked questions about JFreeChart concern the licence. I've published this FAQ to help developers understand my choice of licence for JFreeChart. If anything is unclear, or technically incorrect, please e-mail me (`dave@jfree.org`) and I will try to improve the text.

C.3.2 Questions and Answers

1. "Can I incorporate JFreeChart into a proprietary (closed-source) application?"

Yes, the GNU Lesser General Public Licence (LGPL) is specifically designed to allow this.

2. "Do I have to pay a licence fee to use JFreeChart?"

No, JFreeChart is free software. You are not required to pay a fee to use JFreeChart. All that we ask is that you comply with the terms of the licence, which (for most developers) is not very difficult.

If you want to sponsor the JFreeChart project, you can do so via GitHub:

```
https://github.com/sponsors/jfree
```

This is appreciated, but not required.

3. "If I use JFreeChart, do I have to release the source code for my application under the terms of the LGPL?"

No, you can choose whatever licence you wish for your software. But when you distribute your application, you must include the complete source code for JFreeChart—including any changes you make to it—under the terms of the LGPL. Your users end up with the same rights in relation to JFreeChart as you have been granted under the LGPL.

4. "My users will never look at the source code, and if they did, they wouldn't know what to do with it...why do I have to give it to them?"

The important point is that your users have access to the source code—whether or not they choose to use it is up to them. Bear in mind that non-technical users *can* make use of the source code by hiring someone else to work on it for them.

5. "What are the steps I must follow to release software that incorporates JFreeChart?"

The steps are listed in the licence (see section 6 especially). The most important things are:

- include a notice in your software that it uses the JFreeChart class library, and that the library is covered by the LGPL;
- include a copy of the LGPL so your users understand that JFreeChart is distributed WITHOUT WARRANTY, and the rights that they have under the licence;
- include the complete source code for the version of the library that you are distributing (or a written offer to supply it on demand);

6. "I want to display the JFreeChart copyright notice, what form should it take?"

Try this:

> *This software incorporates JFreeChart, (C)opyright 2000-2025 by David Gilbert and Contributors.*

7. "The LGPL is unnecessarily complicated!"

OK, that's not a question, but the point has been raised by a few developers.

Yes, the LGPL is complicated, but only out of necessity. The complexity is mostly related to the difficulty of defining (in precise legal terms) the relationship between a free software library and a proprietary application that uses the library.

A useful first step towards understanding the LGPL is to read the GNU General Public Licence (GPL). It is a much simpler licence, because it does not allow free software to be combined with non-free (or proprietary) software. The LGPL is a superset of the GPL (you are free to switch from the LGPL to the GPL at any time), but slightly more "relaxed" in that it allows you to combine free and non-free software.

A final note, some of the terminology in the LGPL is easier to understand if you keep in mind that the licence was originally developed with statically-linked C programs in mind. Ensuring that it is possible to relink a modified free library with a non-free application, adds significant complexity to the licence. For Java libraries, where code is dynamically linked, modifying and rebuilding a free library for use with a non-free application needn't be such a big issue, particularly if the free library resides in its own jar file.

8. "Who developed the licence?"

The licence was developed by the Free Software Foundation and has been adopted by many thousands of free software projects. You can find out more information at the Free Software Foundation website:

```
http://www.fsf.org
```

The Free Software Foundation performs important work, please consider supporting them financially.

9. "Have you considered releasing JFreeChart under a different licence, such as an "Apache-style" licence?"

Yes, a range of licences was considered for JFreeChart, but now that the choice has been made there are no plans to change the licence in the future.

A publication by Bruce Perens was especially helpful in comparing the available licences:

```
http://www.oreilly.com/catalog/opensources/book/perens.html
```

In the end, the LGPL was chosen because it is the closest fit in terms of my goals for JFreeChart. It is not a perfect licence, but there is nothing else that comes close (except the GPL) in terms of protecting the freedom of JFreeChart for everyone to use. Also, the LGPL is very widely used, and many developers are already familiar with its requirements.

Some other open source licences (for example the Apache Software Licence) allow open source software to be packaged and redistributed without source code. These licences offer more convenience to developers (especially in large companies) than the LGPL, but they allow a path from open source software to closed source software, which is not something I want to allow for JFreeChart.