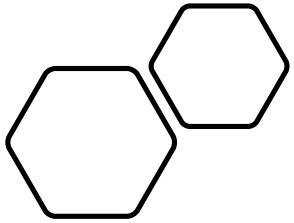
BridgeLabz

Employability Delivered

Eclipse IDE Intro

Narayan Mahadevan

Eclipse Get Started

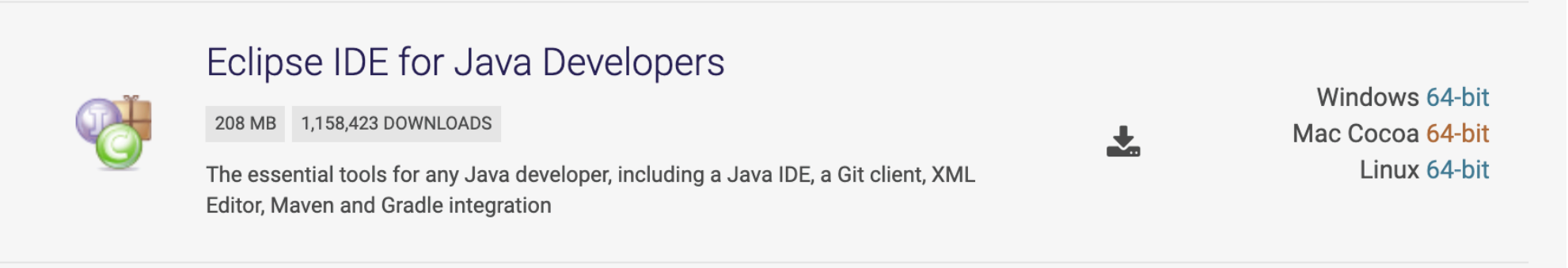


[Create your first Java HelloWorld Application]

Eclipse Installation

BridgeLabz
Employability Delivered

- Download the Eclipse IDE for Java Developers package from the following URL: https://eclipse.org/downloads/eclipse-packages/

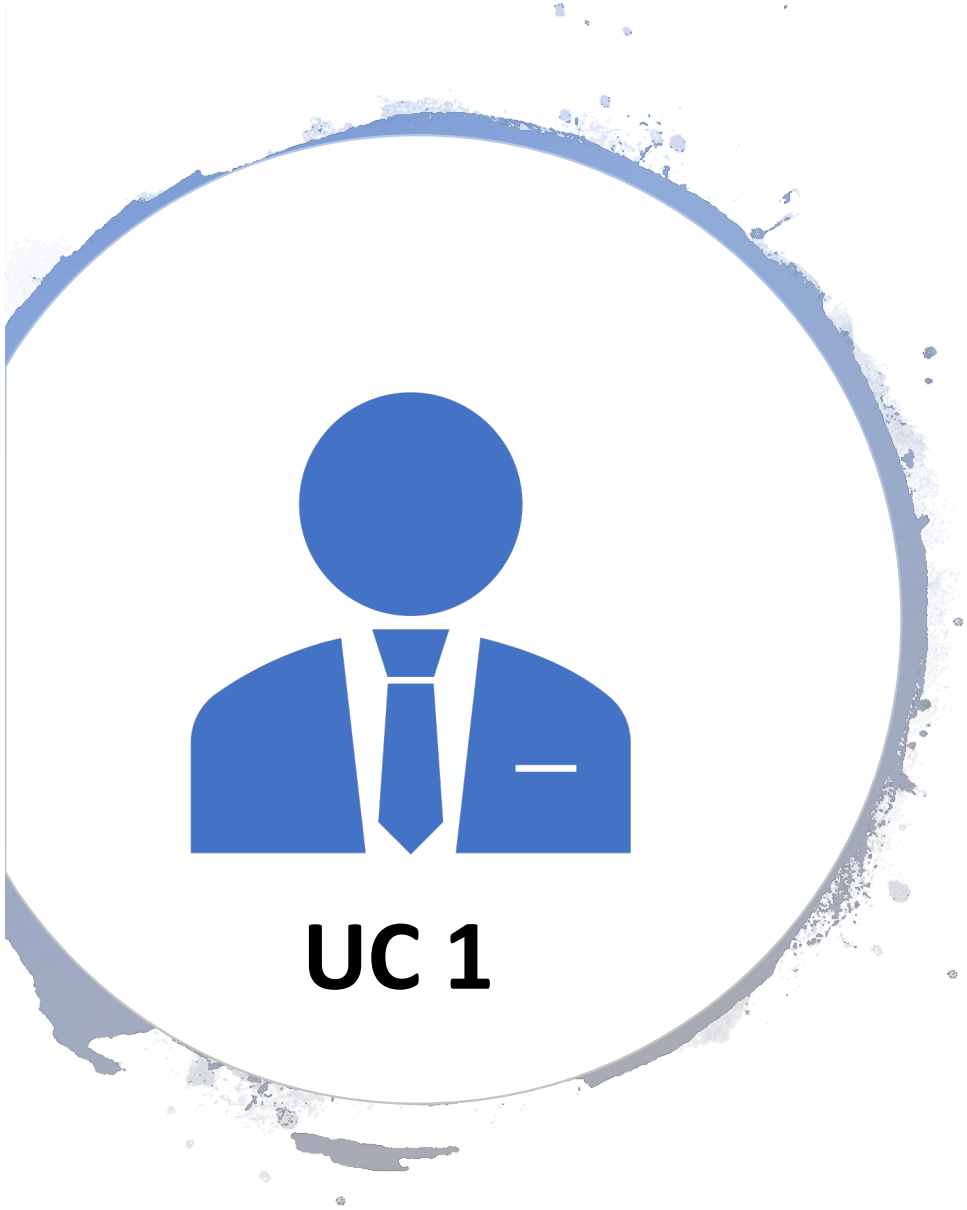


- The download links which are displayed depend on your operating system.

Eclipse Installation

BridgeLabz
Employability Delivered

- After you downloaded the file with the Eclipse distribution, unpack it to a local directory. No additional installation procedure is required, assuming that you Java installed on your machine.
- To start Eclipse, double-click the eclipse.exe (Microsoft Windows) or eclipse (Linux / Mac) file in the directory where you unpacked Eclipse. The Eclipse IDE requires at least **Java 8** to run. If Eclipse does not start, check your Java version.
- The Eclipse system prompts you for a **workspace**. The workspace is the location in your file system where Eclipse stores its preferences and other resources.



Ability to show Hello World Message when running Java Application in Eclipse IDE

Create HelloWorld Java Project

1

Step 1: Create Project – Create Click on Menu File -> New -> Java Project

2

Step 2: Create HelloWorld Java Class – Click Ctrl N to create your HelloWorld Java Class.

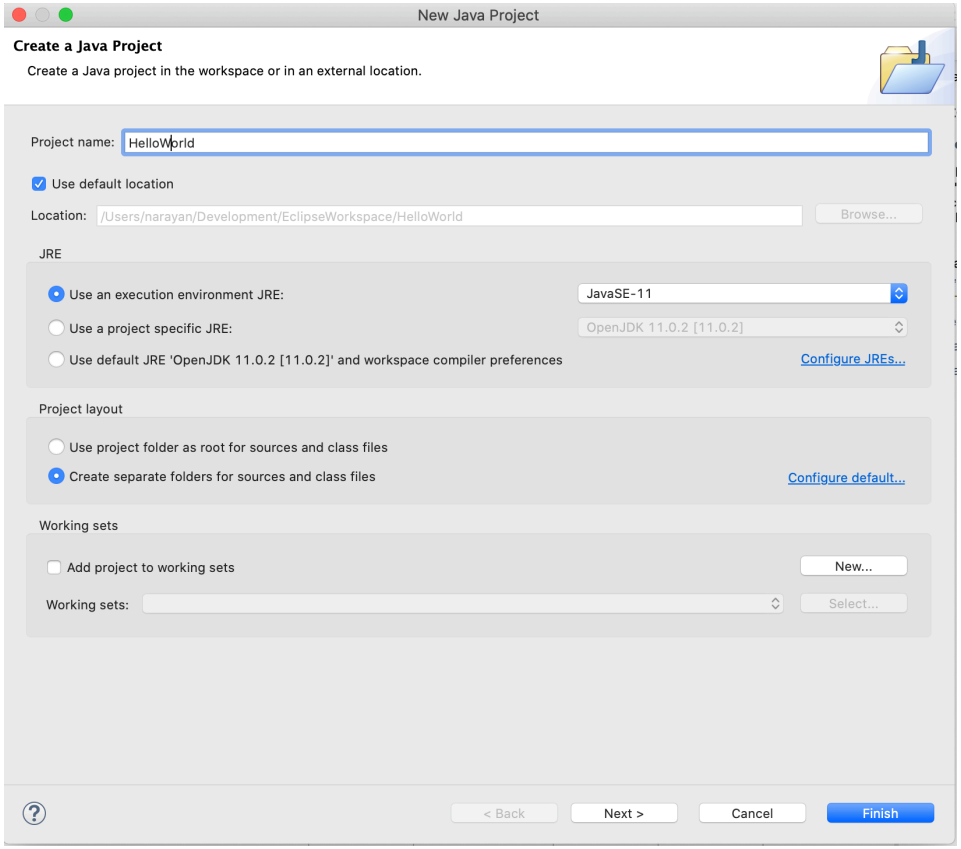
3

Note: Default Package is empty. You can specify something like javasamples

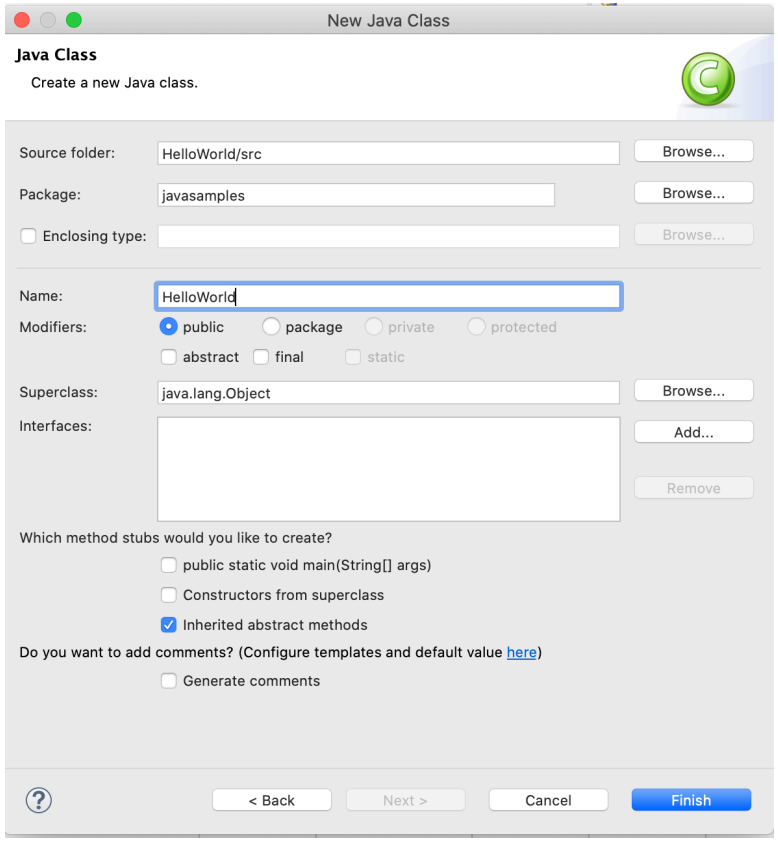
Create New Java Project

BridgeLabz
Employability Delivered

Step 1: Go To File -> New -> Java Project



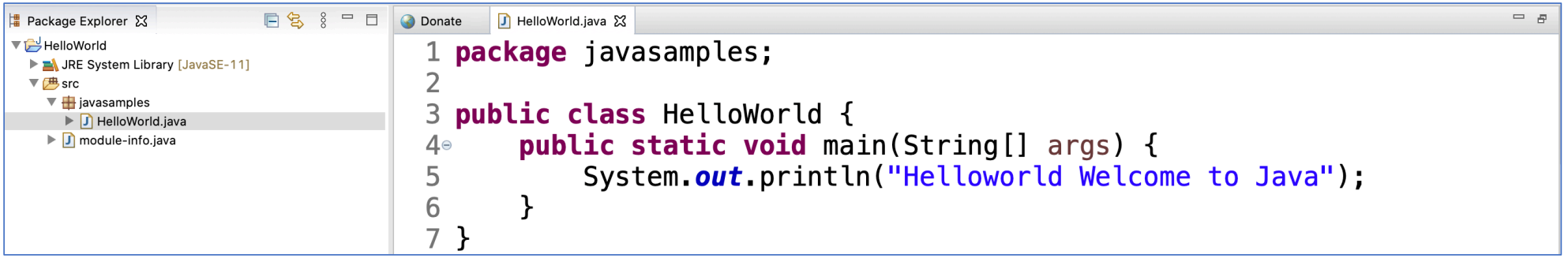
Step 2: Ctrl N to create Java Class



Shortcut Tips and Hello

BridgeLabz
Employability Delivered

- **Tip 1: Create** `public static void main(String args[])`
  - Type `main` and Ctrl - Space
- **Tip 2: For** `System.out.println()`
  - Type sysout Ctrl + Space
- **Tip3: Code Expansion use** Ctrl + Space
- **To Run the Program**

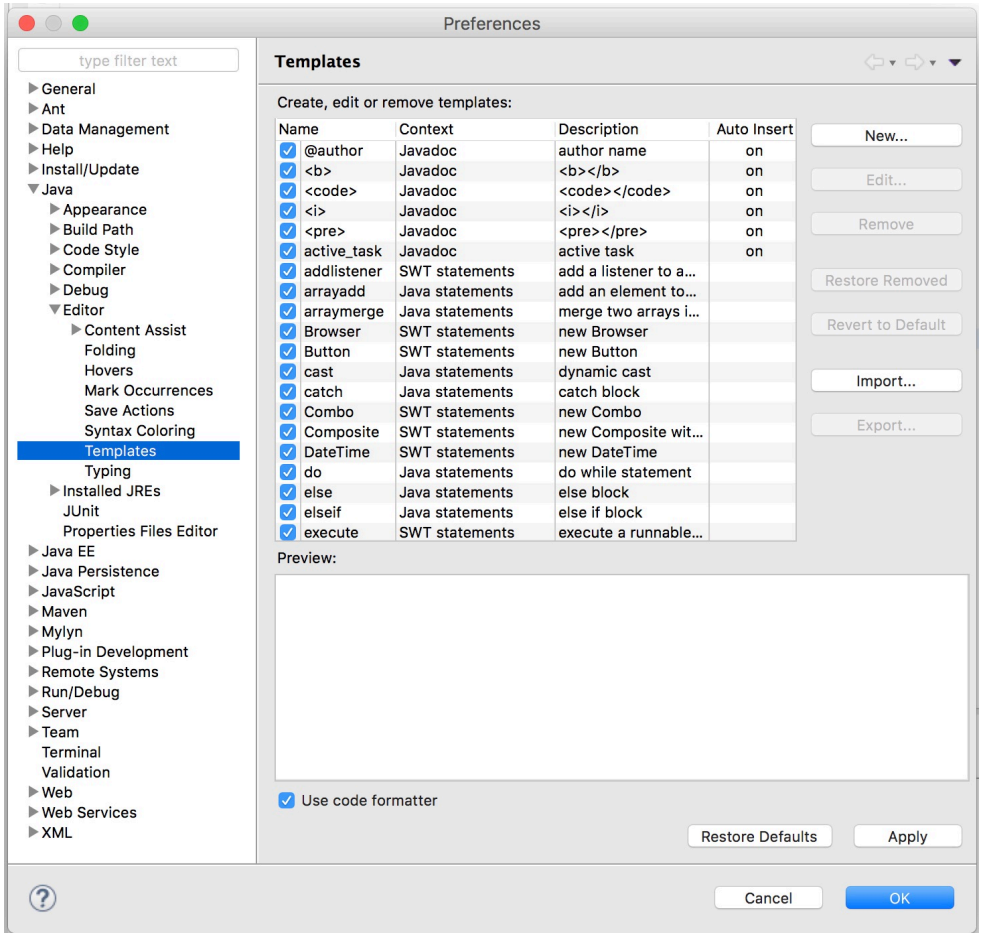


Ability to add psvm short cut in Eclipse for print static void main function

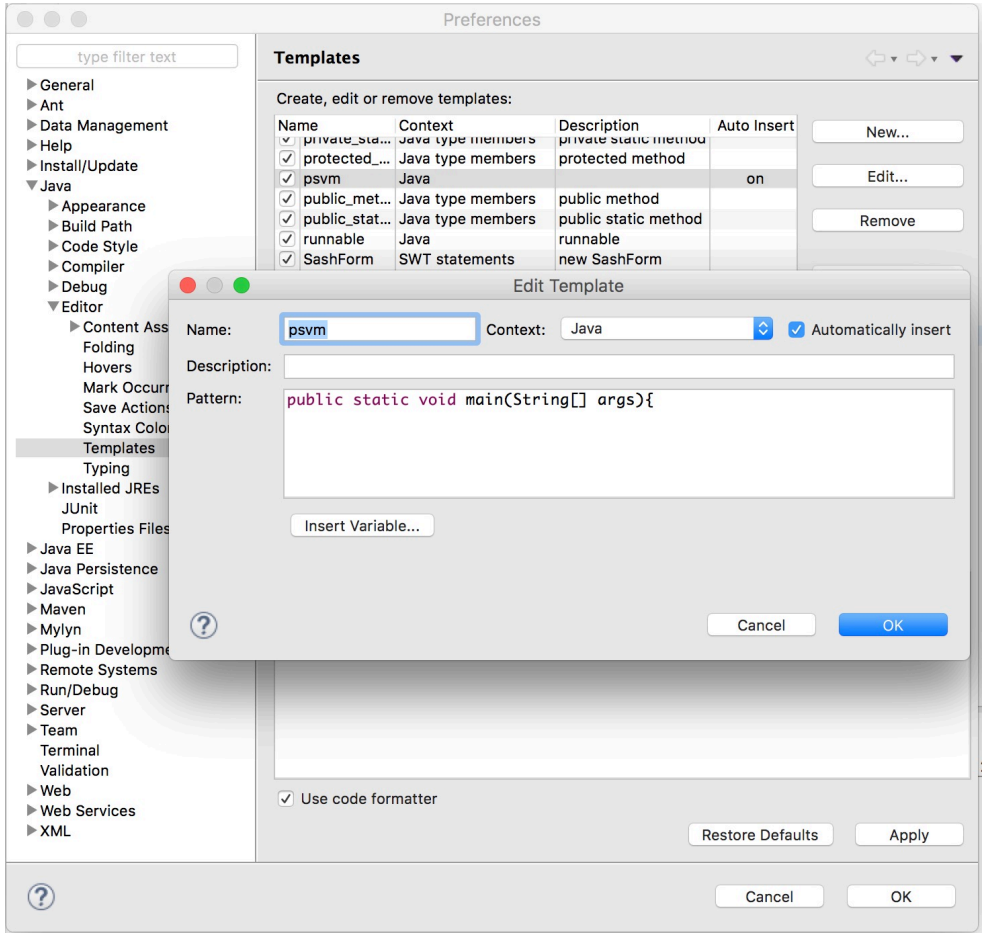
Adding Custom Shortcut

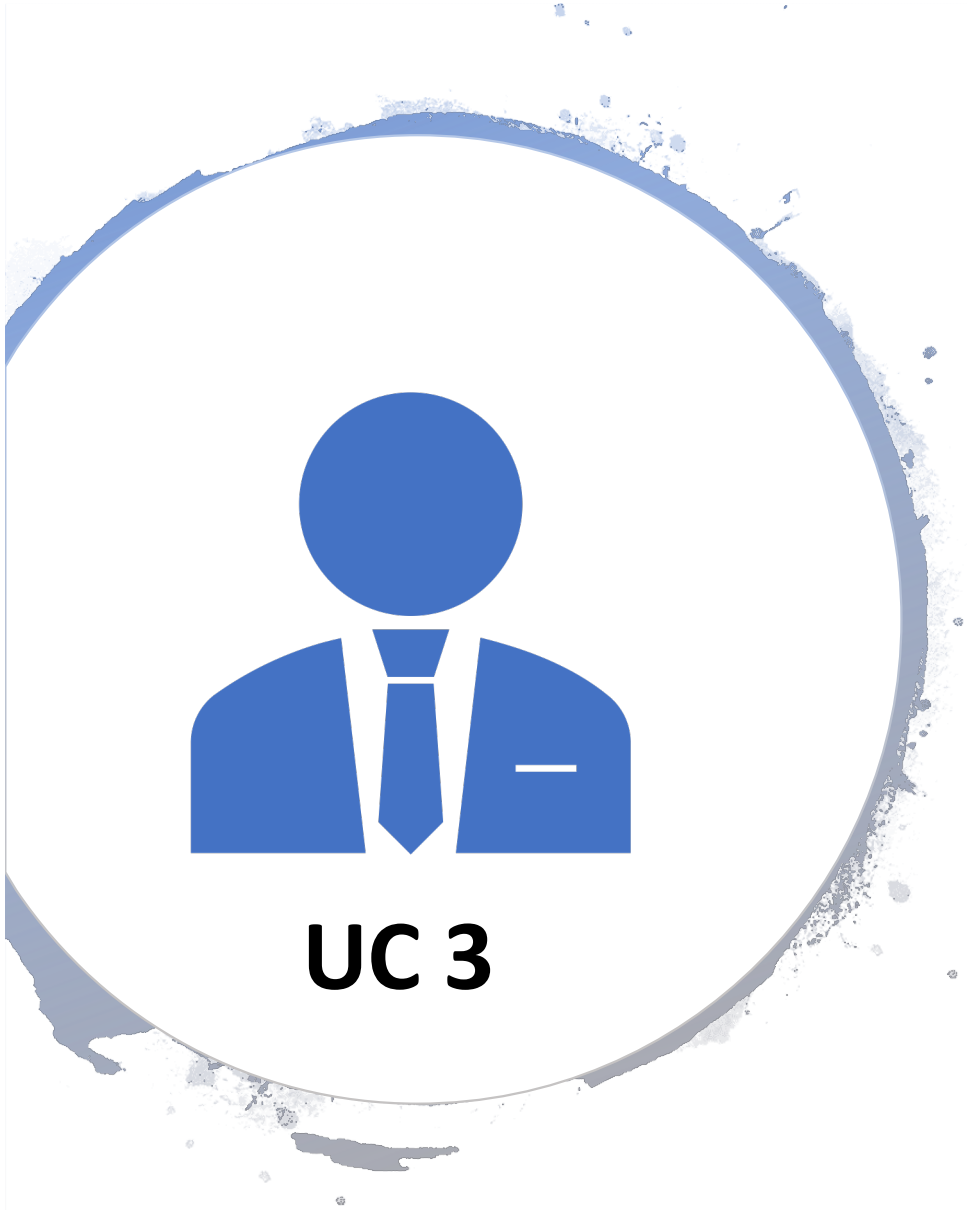
BridgeLabz
Employability Delivered

Step1: Go To Preferences>Java>Editor>Templates>New...



Step2: Add shortcut for public static void main





Ability to show Hello World Message when running Java Maven Application in Eclipse IDE

Maven Projects and Wizard Tool

- Maven is a build automation tool used primarily for Java projects. Maven can also be used to build and manage projects written in C#, Ruby, Scala, and other languages.
- Eclipse Maven Tooling manages the project dependencies and updates the classpath of the project dependencies in the Eclipse IDE. It ensures that the Maven experience in Eclipse is as smooth as possible. The tooling also provides different kind of wizards import and to create new Maven based projects.
- It also provides an editor for the ***pom.xml*** Maven configuration file via a structured interface. You can select the tab labeled *pom.xml* to edit the XML data directly.

Create HelloWorld Maven Project

1

Step 1: Create Maven Project – Create Click on Menu File -> New -> Project -> Maven Project

2

Step 2: Select Maven Artifact – Filter with maven-archetype-quickstart to select org.apache.maven.archetype

3

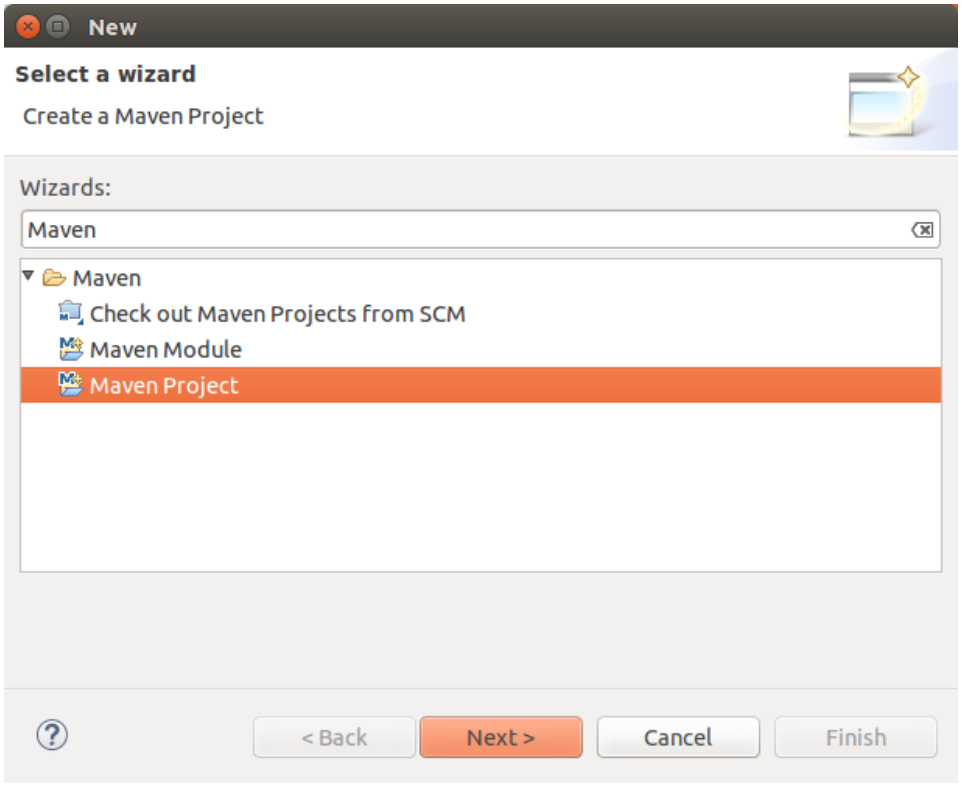
Step 3: Create Maven Project – Enter

1: Group Id: Package Prefix
2: Artifact Id: Project Name
3: Package

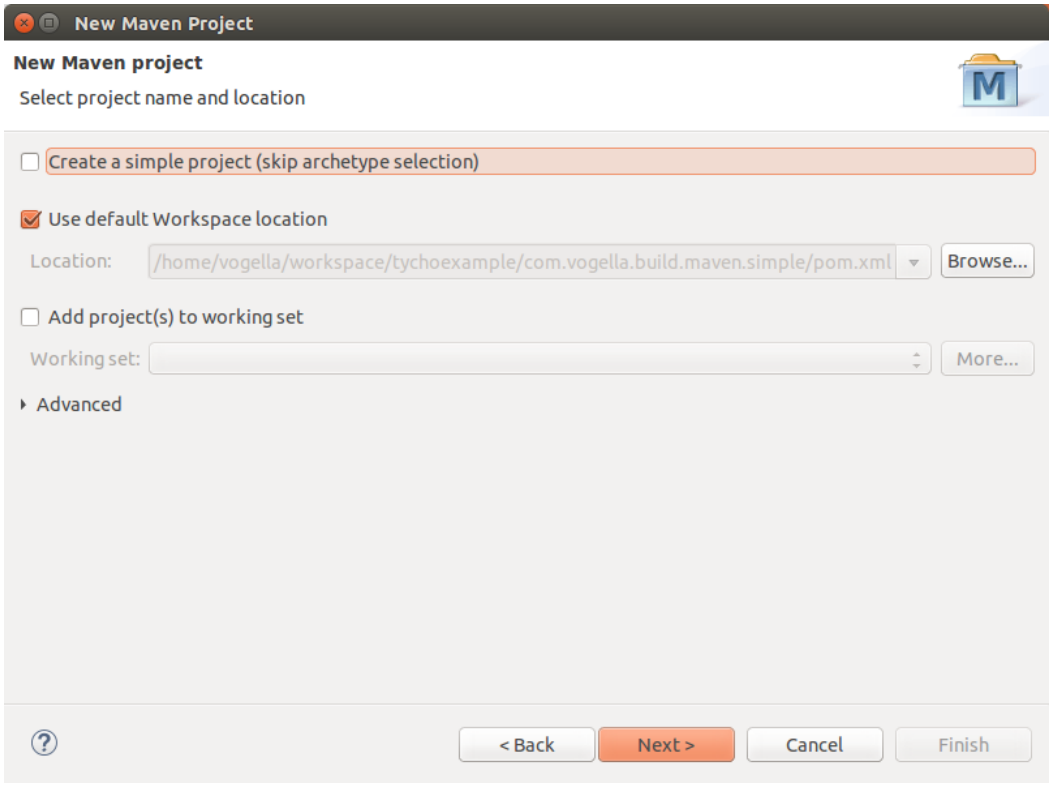
Create New Java Maven Project

BridgeLabz
Employability Delivered

Step 1: Go To File -> New -> Project -> Maven Project



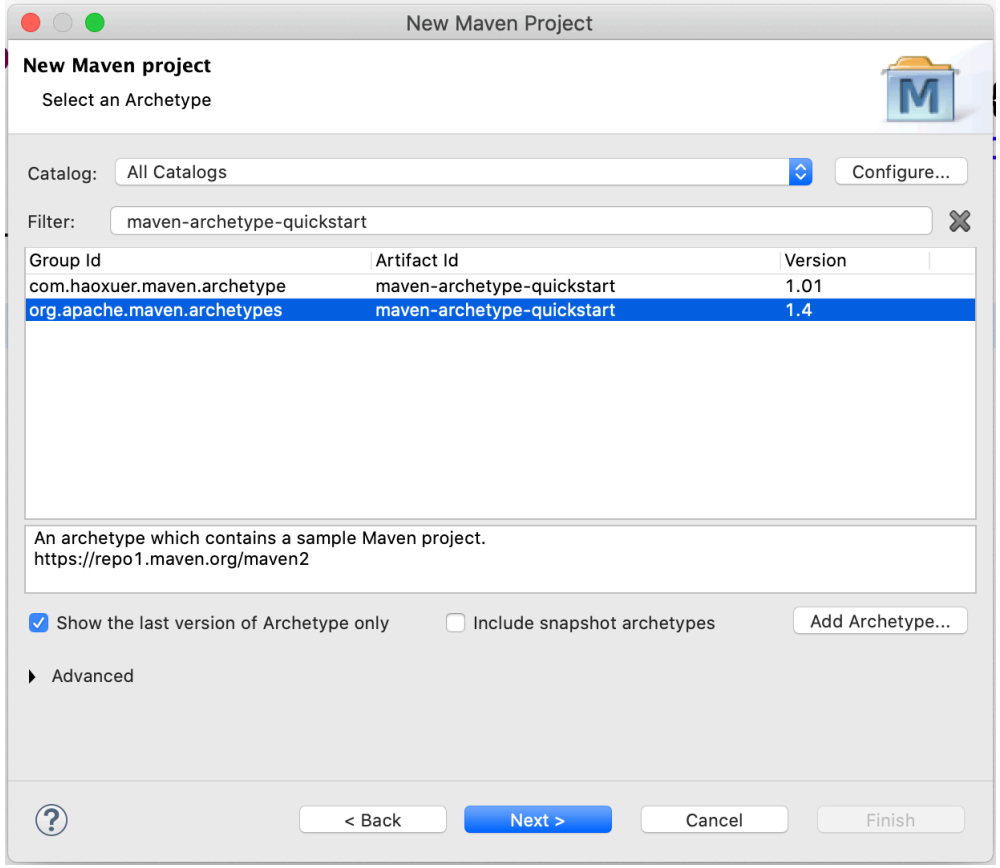
Step 2: Use Default Workspace Location



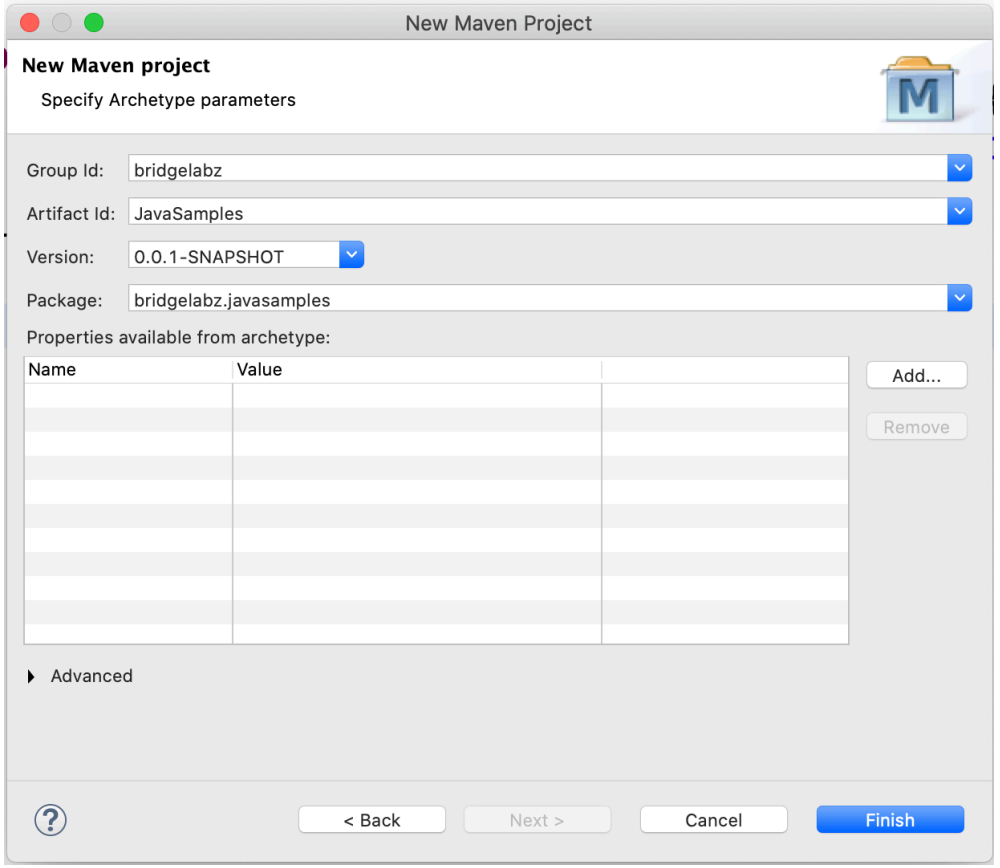
Create New Java Maven Project

BridgeLabz
Employability Delivered

Step 3: Filter using maven-archetype-quickstart

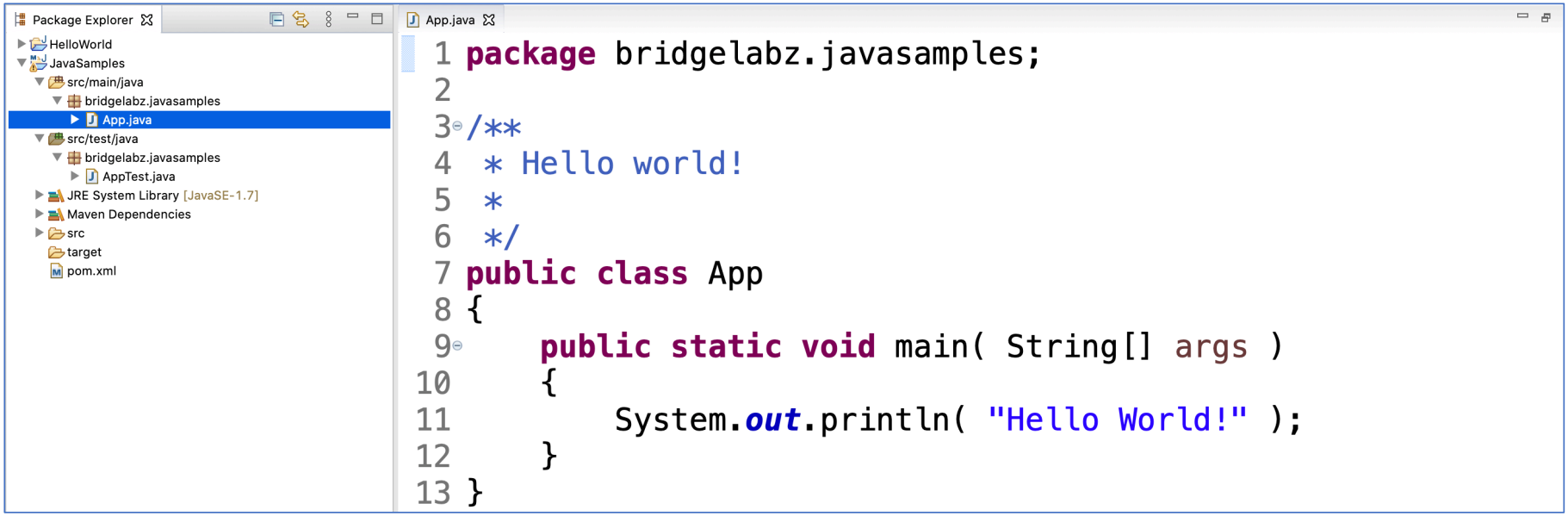


Step 4: Specify Group Id, Artifact Id



Create New Java Maven Project

Step 5: App.java and AppTest.java are Auto Created. Run App.java to print Hello World!

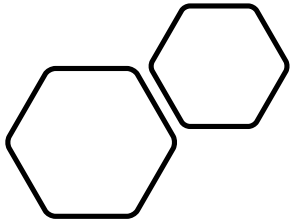


References

BridgeLabz
Employability Delivered

- https://www.vogella.com/tutorials/EclipseMaven/article.html
- https://www.tutorialspoint.com/How-to-write-generate-and-use-Javadoc-in-Eclipse
- https://www.vogella.com/tutorials/EclipseShortcuts/article.html
- https://examples.javacodegeeks.com/enterprise-java/log4j/log4j-2-getting-started-example/

Log4j2 Get Started



Extend HelloWorld To Log Messages using Log4j

Why Log4j 2???

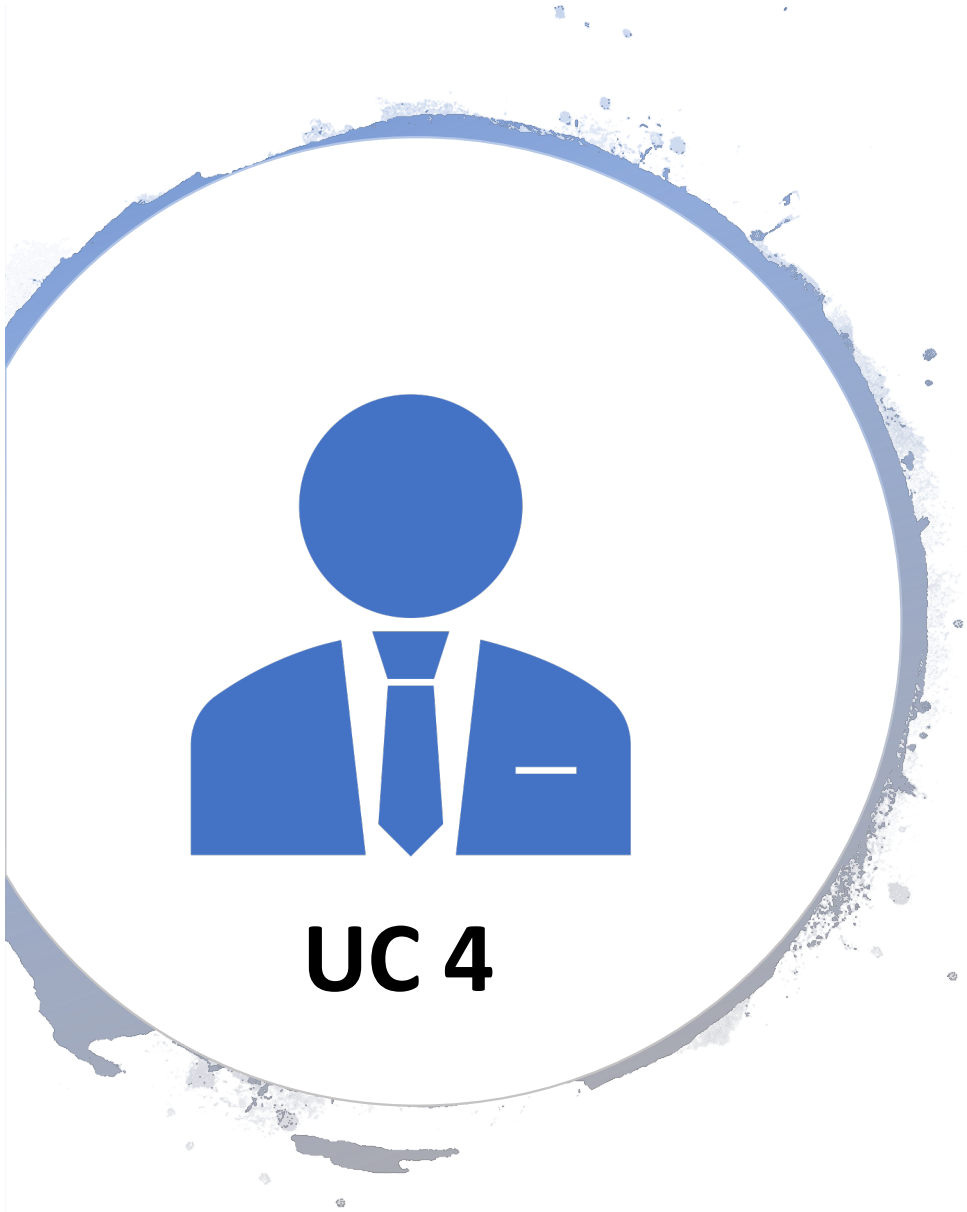
BridgeLabz
Employability Delivered

- Printing or Logging messages to the console is an integral part of debugging of a Java program especially in a **Live Production Environment**.
- Especially if developers are working on a **Server side application**, where they cannot see what's going on inside the server, then their only visibility tool is a log file.
- Though, Java has handy `System.out.println()` methods to print something on console, which can also be routed to log file but not sufficient for a **real-world Java application**.

What is Log4j2???

BridgeLabz
Employability Delivered

- Log4j2 is the updated version of the popular and influential Log4j library, which is a simple, flexible, and fast Java-based **logging framework**. It is **thread-safe** and supports internationalization.
- We mainly have 3 components to work with Log4j:
  - **Logger:** It is used to log the messages.
  - **Appender:** It is used to publish the logging information to the destination like the file, database, console etc.
  - **Layout:** It is used to format logging information in different styles.
- Especially if developers are working on a **Server side application**, where they cannot see what's going on inside the server, then their only visibility tool is a log file.



Ability to show Hello World Message using Log4j2 Logging Library when running Java Maven Application in Eclipse IDE

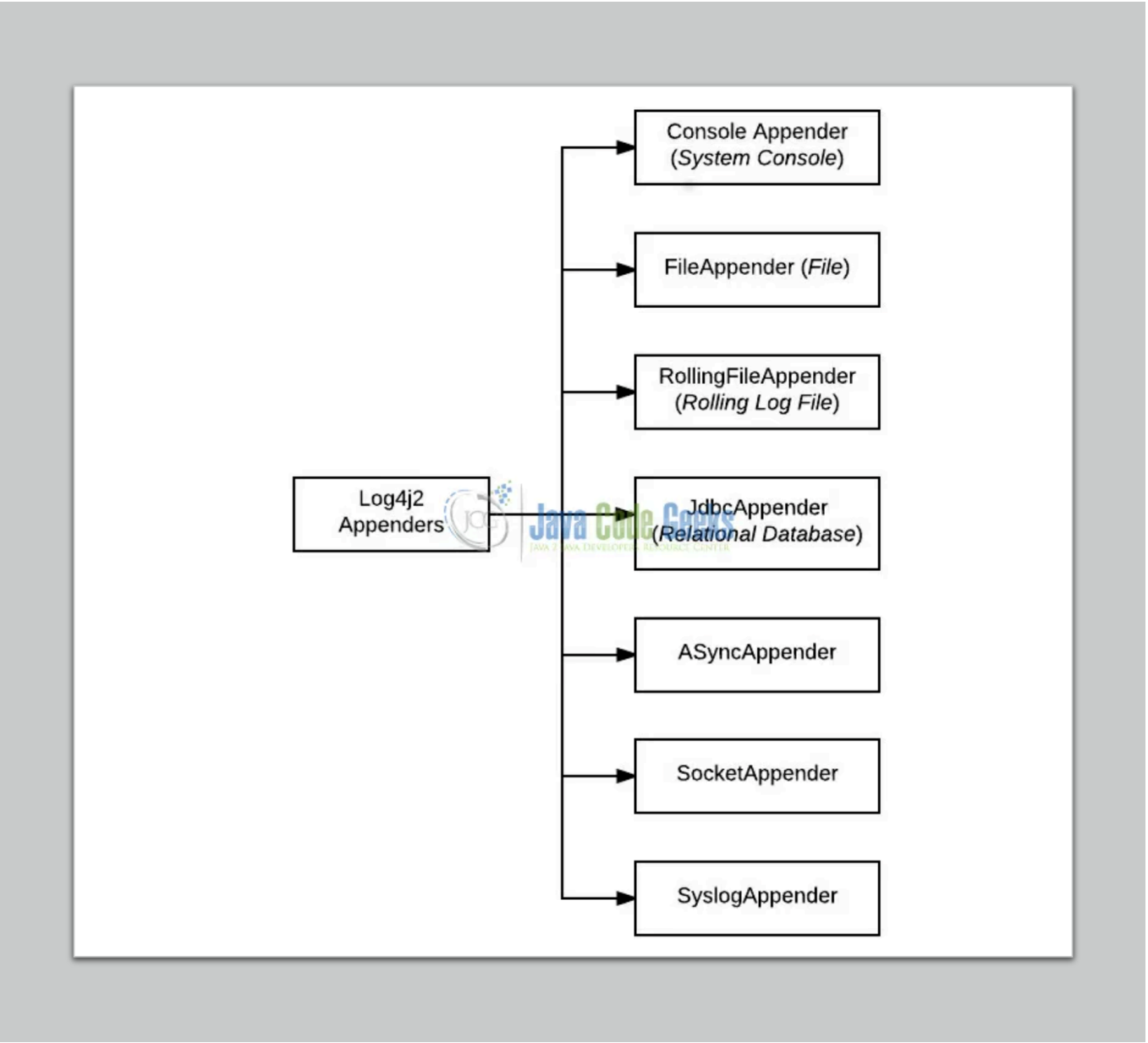
Log4j2 Logger Class

`Logger` class provides the methods for the logging process. We can use the `LogManager.getLogger()` method to get the Logger object. Here are 5 different logging methods which can be used in an application.

	Description	Method Syntax
debug(Object message)	It is used to print the message with the level `org.apache.logging.log4j.Level.DEBUG`.	public void debug(Object message)
error(Object message)	It is used to print the message with the level `org.apache.logging.log4j.Level.ERROR`.	public void error(Object message)
info(Object message)	It is used to print the message with the level `org.apache.logging.log4j.Level.INFO`.	public void info(Object message)
fatal(Object message)	It is used to print the message with the level `org.apache.logging.log4j.Level.FATAL`.	public void fatal(Object message)
warn(Object message)	It is used to print the message with the level `org.apache.logging.log4j.Level.WARN`.	public void warn(Object message)
trace(Object message)	It is used to print the message with the level `org.apache.logging.log4j.Level.TRACE`.	public void trace(Object message)

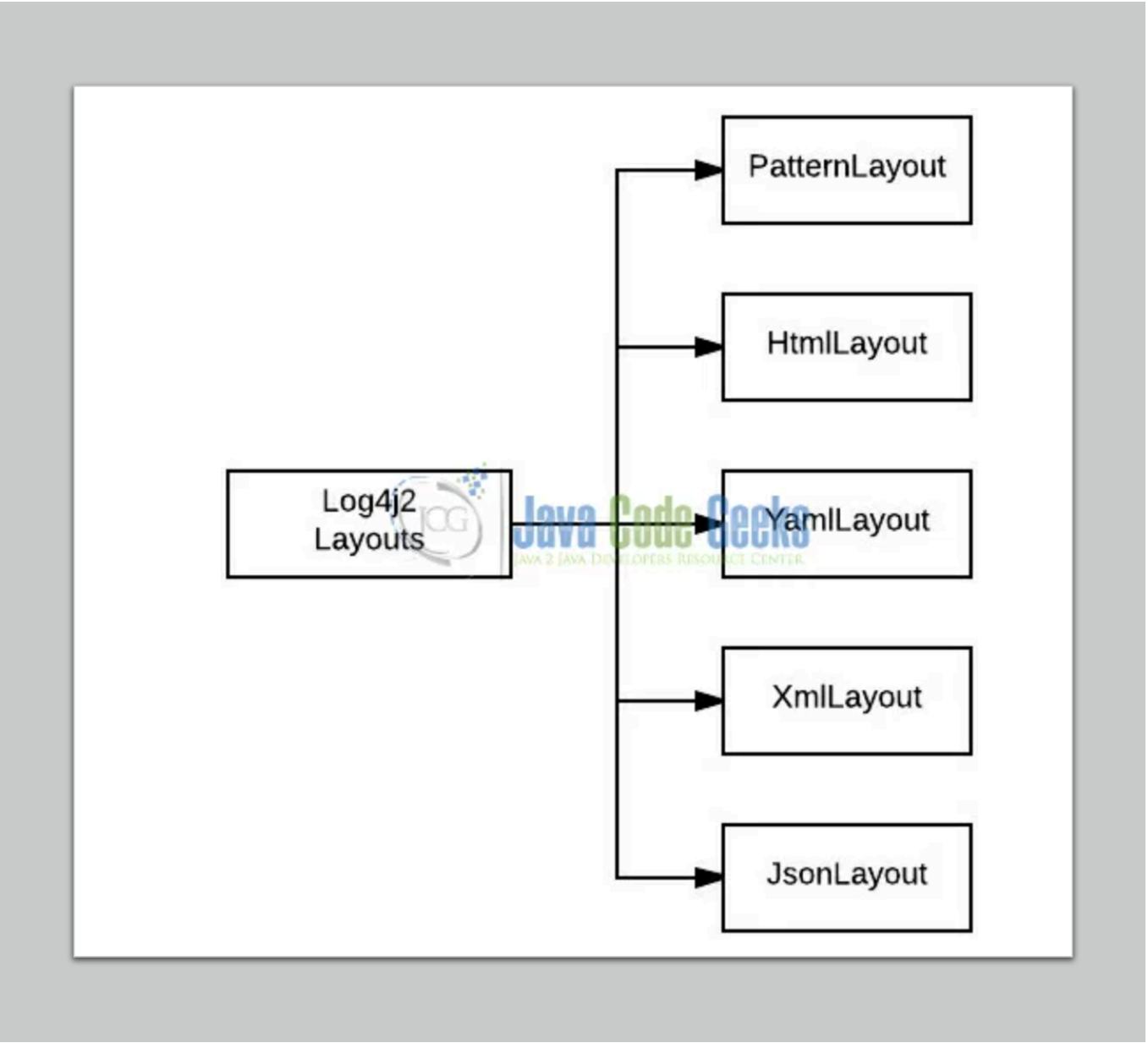
Log4j2 Appender Interface

`Appender` is an interface which is primarily responsible for printing the logging messages to the different destinations such as console, files, sockets, database etc. In `Log4j2` we have different types of `Appender` implementation classes.



Log4j Layout Class

`Layout` component specifies the format in which the log statements are written into the destination repository by the `Appender`. In `Log4j2` we have different types of `Layout` implementation classes.



Use Log4J2 with HelloWorld Maven Project

1

Step 1: `pom.xml` – Add Log4J2 Core and API dependency to `pom.xml`. Search google mvnrepository

2

Step 2: Add `log4j2.xml` – Add `log4j2.xml` to `src/main/resources` directory

3

Step 3: Notice in `log4j2.xml` – the following

1: Base Path – Logging Dir.
2: Appenders – File & Console
3: Loggers – Level & appender ref

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Configuration xmlns="http://logging.apache.org/log4j/2.0/config">
    <Properties>
        <Property name="basePath">logs</Property>
    </Properties>
    <Appenders>
        <!-- File Appender -->
        <File name="FILE" fileName="${basePath}/logfile.log" append="true">
            <PatternLayout pattern="%-5p | %d{yyyy-MM-dd HH:mm:ss} | [%t] %C{2} (%F:%L) - %m%n" />
        </File>
        <!-- Console Appender -->
        <Console name="STDOUT" target="SYSTEM_OUT">
            <PatternLayout pattern="%-5p | %d{yyyy-MM-dd HH:mm:ss} | [%t] %C{2} (%F:%L) - %m%n" />
        </Console>
    </Appenders>
    <Loggers>
        <Logger name="com.jcg" level="debug" />
        <Root level="info">
            <AppenderRef ref="STDOUT" />
            <AppenderRef ref="FILE" />
        </Root>
    </Loggers>
</Configuration>
```

log4j2.xml

```java
package bridgelabz.javasamples;

import org.apache.logging.log4j.LogManager;
import org.apache.logging.log4j.Logger;

/**
 * Hello world!
 *
 */
public class App
{
    private static final Logger LOG = LogManager.getLogger(App.class)
    public static void main( String[] args )
    {
        String message = "Hello, World";
        LOG.debug(message + " Will Be Printed On Debug");
        LOG.info(message + " Will Be Printed On Info");
        LOG.warn(message + " Will Be Printed On Warn");
        LOG.error(message + " Will Be Printed On Error");
        LOG.fatal(message + " Will Be Printed On Fatal");
        LOG.info("Appending string: {}.", message);
        System.out.println(message);
    }
}
```

Hello World App.java

BridgeLabz

Employability Delivered

Thankyou