June 9, 2017

Adam Clemons

java and maven primer companion document

A merry romp through java, maven, and jpa persistence

Contents

[Java Setup 5](#_Toc488148189)

[Installation 5](#_Toc488148190)

[Commands 5](#_Toc488148191)

[Version 5](#_Toc488148192)

[javac 5](#_Toc488148193)

[Eclipse Setup 6](#_Toc488148194)

[Installer 6](#_Toc488148195)

[Zip Archive 6](#_Toc488148196)

[NetBeans Setup 6](#_Toc488148197)

[Java Primitive Types and Operators 7](#_Toc488148198)

[Java Abstract and Interface Types 8](#_Toc488148199)

[Interface 8](#_Toc488148200)

[Java Implements and Extends 9](#_Toc488148201)

[@Override 9](#_Toc488148202)

[Java Entities – JPA and Hibernate 9](#_Toc488148203)

[@Entity 9](#_Toc488148204)

[Abstraction 9](#_Toc488148205)

[Generic Types 9](#_Toc488148206)

[Reflection 9](#_Toc488148207)

[extends vs. implements 10](#_Toc488148208)

[Maven 10](#_Toc488148209)

[Installation 10](#_Toc488148210)

[Adding A Private Nexus Repository to settings.xml 10](#_Toc488148211)

[Commands 10](#_Toc488148212)

[mvn install 10](#_Toc488148213)

[mvn clean 10](#_Toc488148214)

[-e and -U 10](#_Toc488148215)

[dependency:tree 11](#_Toc488148216)

[Using Maven with NetBeans 11](#_Toc488148217)

[Using Maven with Eclipse 12](#_Toc488148218)

[Getting the Demo Code and opening in your IDE 15](#_Toc488148219)

[Eclipse 15](#_Toc488148220)

[NetBeans 15](#_Toc488148221)

[Generic Text Editor (advanced) 15](#_Toc488148222)

[Running the Demo Code 15](#_Toc488148223)

[Eclipse 15](#_Toc488148224)

[NetBeans 15](#_Toc488148225)

[Generic Text Editor 15](#_Toc488148226)

Java Quick Start Overview

# Java Setup

## Installation

A guide for installing the latest Java JDK can be found here - <http://docs.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html>

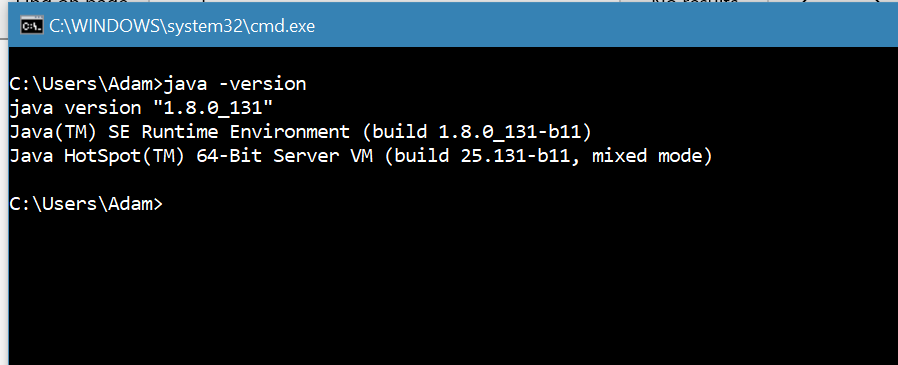
These instructions are written for Java SE 7, but they also apply to Java SE 8, just download the appropriate version

## Commands

Most Commands require you to have Java set on your System Path. Instructions are provided in the link from the previous section.

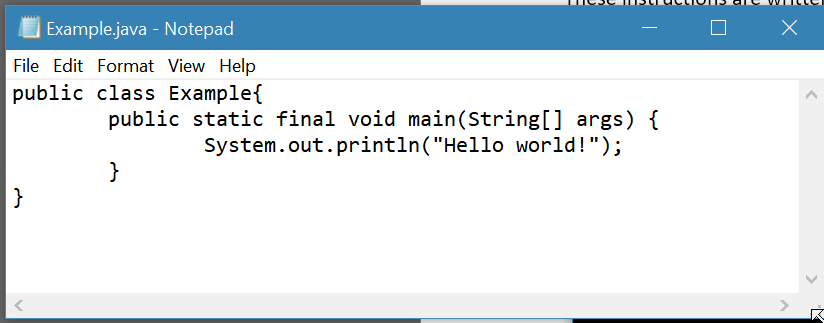
### Version

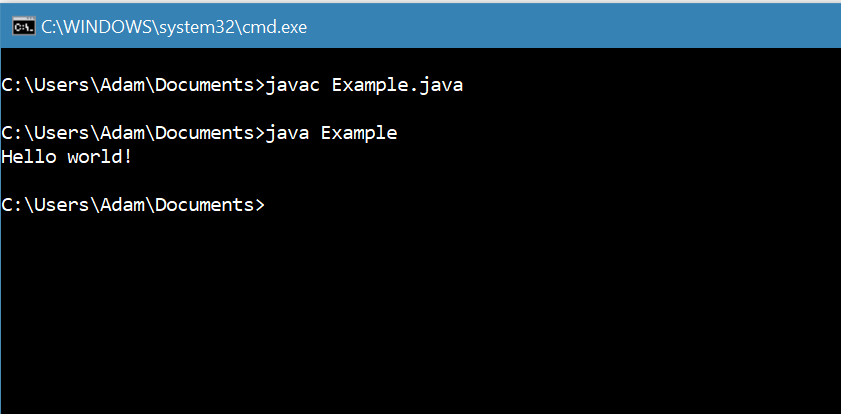
If Java is on your System Path, then you should be able to check the version via command line using the following command.



### javac

The javac command is used to compile .java files to .class files. This is not often used, as most projects use a build system like Maven. javac can be used for simple demos like the one shown below.





# Eclipse Setup

You can install Eclipse one of two ways. One will download, extract and create shortcuts for you, the other will just give you a .zip archive to extract. Here are links to.

### Installer

<https://www.eclipse.org/downloads/download.php?file=/oomph/epp/neon/R3/eclipse-inst-win64.exe>

Be sure to select the Java EE version, as it includes many plugins that help with Maven and web application development.

## Zip Archive

<http://www.eclipse.org/downloads/packages/eclipse-ide-java-ee-developers/neon3>

# NetBeans Setup

NetBeans can be downloaded with the Java JDK as a bundle, or it can be downloaded and installed separately. You will need Admin rights on your PC to install.

<https://netbeans.org/downloads/>

The Java EE version is best for Web Application Development.

# Java Primitive Types and Operators

**Primitive Types**

|  |  |  |  |
| --- | --- | --- | --- |
| **TYPE** | **DEFAULT VALUE** | **SIZE** | **Description** |
| Boolean | False | 1 bit | True/False values |
| Char | \u0000 | 16 bits | Single Unicode character |
| Int | 0 | 32 bits | Signed 32-bit integer, ~-2.14M and ~2.14M |
| Long | 0 | 64 bits | Singed 64-bit integer, This is usually used for non UUID primary keys. |
| Float | 0.0 | 32 bits | 32 bit decimal number |
| Double | 0.0 | 64 bits | Usually if greater than 32-bit precision is needed, the BigDecimal class is used. BigDecimal is not a primitive. |

Java also has a number of built-in operators. These come in several types.

For a complete list of Unicode Chars you can use, see this complete list - <http://www.ssec.wisc.edu/~tomw/java/unicode.html>

**Primitive Operators**

This is not an exhaustive list. For complete list of operators, including the bitwise operators excluded from this document, see <https://www.tutorialspoint.com/java/java_basic_operators.htm>

|  |  |  |
| --- | --- | --- |
| **Unary Operators** | **Example expression** | **Yields** |
| ++ | 1 ++ | 2 |
| -- | 1 -- | 0 |

|  |  |  |
| --- | --- | --- |
| **Arithmetic Operators** | **Example** | **Yields** |
| \* | 5 \* 4 | 20 |
| / | 4 / 2 | 2 |
| % | 27 % 5 | 2 |



|  |  |  |
| --- | --- | --- |
| **Relational Operators** | **Example** | **Yields** |
| < | 2 < 3 | True |
| > | 2 > 3 | False |
| <= | 3 <= 4 | True |
| >= | 3 >= 4 | True |
| instanceof | 1 instanceof String | False |



|  |  |  |
| --- | --- | --- |
| **Logical Operators** | **Example** | **Yields** |
| && | True && False | False |
| || | True || False | True |

|  |  |  |
| --- | --- | --- |
| **Ternary Operators** | **Example** | **Yields** |
| ? | var = (True) ? 1 : 0 | Var = 1 |
| : | var = (False) ? 1 : 0 | Var = 0 |

|  |  |  |
| --- | --- | --- |
| **Assignment Operators** | **Example** | **Yields** |
| = |  |  |
| += |  |  |
| -= |  |  |
| \*= |  |  |
| /= |  |  |
| %= |  |  |

# Java Abstract and Interface Types

Java is an Object Oriented Language[[1]](#footnote-1), meaning that all things are Objects and Object Attributes can be inherited from other Objects.

## Interface

An Interface can be thought of as a contract. All Classes that implement an interface must implement all the methods within that interface[[2]](#footnote-2). Here’s an Example Interface.

## Java Implements and Extends

## @Override

# Java Entities – JPA and Hibernate

## @Entity

SBDemo entities package has a few examples.

# Abstraction

## Generic Types

Java allows the use of Generic Type variables, so you can make dynamically typed container classes, You can also use this to create Factories for service implementations, or even for entire UI components.

You can see an example of generic types in the SBDemo project with the AbstractListView class.

## Reflection

Reflection is a magical tool. With reflection, you can dynamically build a UI based on the fields available in a Bean definition, or you can have one method that gets and sets multiple fields based on the type of the parameters.

You can see examples of these in the SBDemoSession and AbstractListView classes.

## extends vs. implements

# Maven

## Installation

The Maven installation documentation is provided on the maven apache website here - <https://maven.apache.org/guides/getting-started/windows-prerequisites.html>

## Adding A Private Nexus Repository to settings.xml

In the ${maven.home}\conf directory, there is a file named ***settings.xml***. If your organization uses a private Nexus Repository, you may need to add a Repository to your configuration to resolve internally developed dependencies. Someone on your Organization’s dev team should be able to provide the configuration values to you.

If you notice that dependency artifacts are not resolving, misconfiguration of the settings.xml is a likely culprit.

## Commands

For a complete list of Maven Commands, you can check the Maven Documentation here –

<https://maven.apache.org/ref/3.1.0/maven-embedder/cli.html>

For the purposes of this document, we will focus on a subset of the Command line options to be more practical for users new to Maven.

All Maven commands should be executed through the IDE, or in the same directory as your POM.xml project file.

### mvn install

This command will build your project artifact and add the artifact to your local Maven repository. The local maven repository is a cache of your build dependencies and your compiled project artifacts.

### mvn clean

clean will remove all compiled or generated files from the ${project.root} directory. Sometimes this is needed to clear out older generated sources (such as generated object Classes from an XSD or WSDL) to make sure you are building with the latest sources. This command can be chained with the install command to perform both actions in one run.

### -e and -U

In some instances, a maven build may fail locally and not provide enough logging to determine the error. To add extra logging to your Maven builds, use the -e command line switch.

You can also force an update of all your dependencies from the Maven Central (or your organizations’ Nexus) Repository by using the -U switch. This will go to the central or nexus repository to get the latest dependencies, even if your local versions were just pulled from the repository.

This is helpful when a new dependency versions is available in the Central or Nexus repository, but your local Maven repository has not marked that dependency to be re-fetched. This is generally only an issue with SNAPSHOT builds within your own organization.

Example - mvn clean -e -U install

### dependency:tree

Maven also has the ability to output the full dependency tree of a project to allow you to easily detect conflicts, duplicates and even inherited dependencies in the console. To run this, you can use the following

mvn dependency:tree

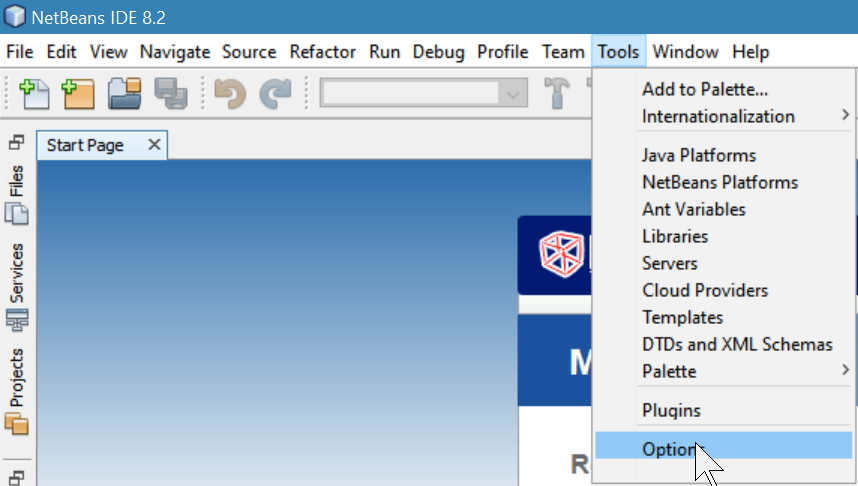
or, for even greater detail

mvn dependency:tree -Dverbose

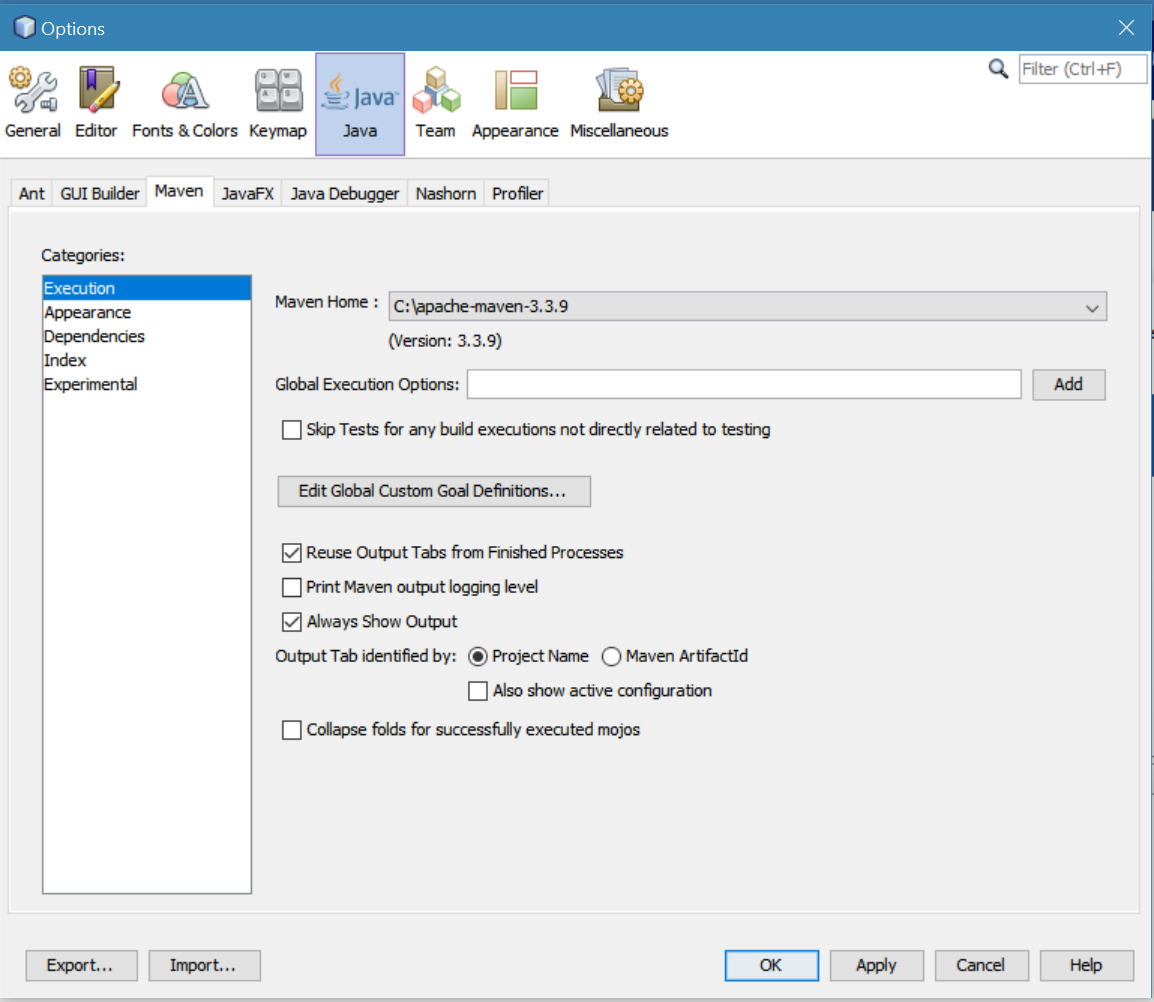
## Using Maven with NetBeans

NetBeans for Java EE comes with a version of Maven bundled with it. However, in some situations, it is best to use a system-wide maven installation for sharing your Maven settings between multiple IDEs, or to allow for easier configuration of Maven.

If you are using a system-wide Maven installation as described above, you can configure NetBeans to use this instead by going to ***Tools*** > ***Options*** as shown below.



Next, go to the ***Java*** > ***Maven*** section in the Options Menu.

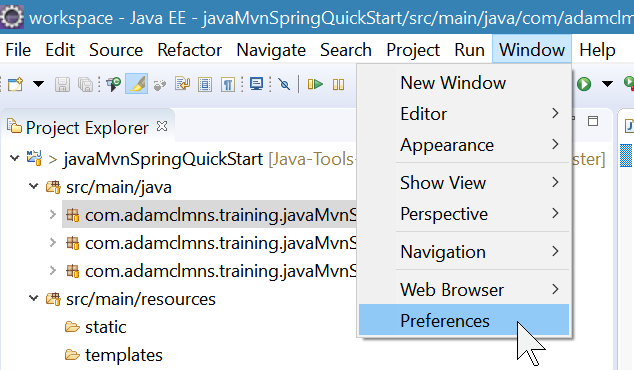


In the dropdown labeled ***Maven Home:*** you can browse to your local Maven Installation. The screenshot above shows NetBeans is already configured to use the Maven installation at *C:\apache-maven-3.3.9\*

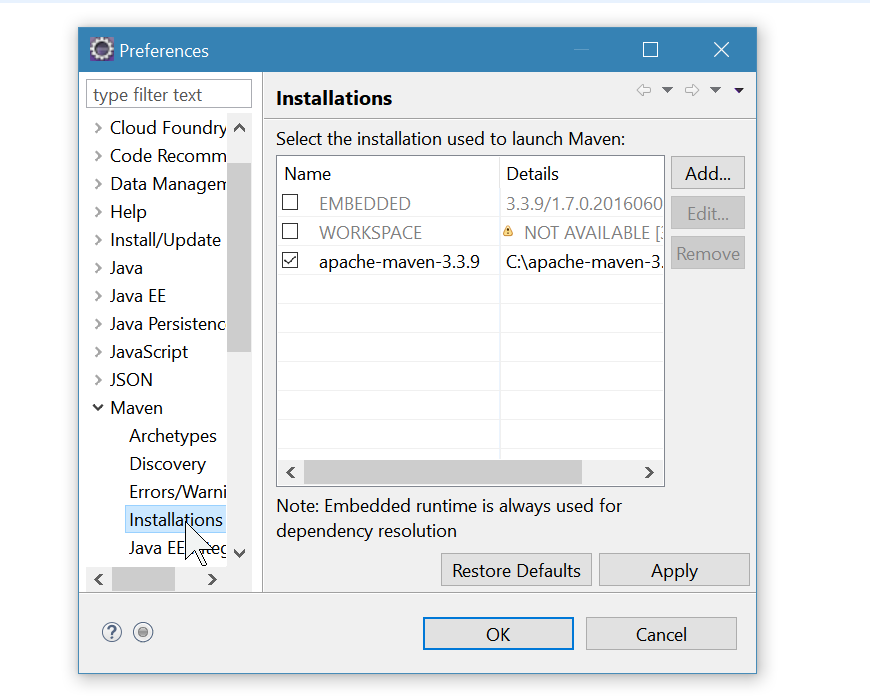
## Using Maven with Eclipse

Eclipse, just like NetBeans also includes a built-in Maven version. This will work for general development, but if you plan to bounce between code editors, or if you want to use the same Maven from the command line that your IDE is using, you can set it to use the system installed version.

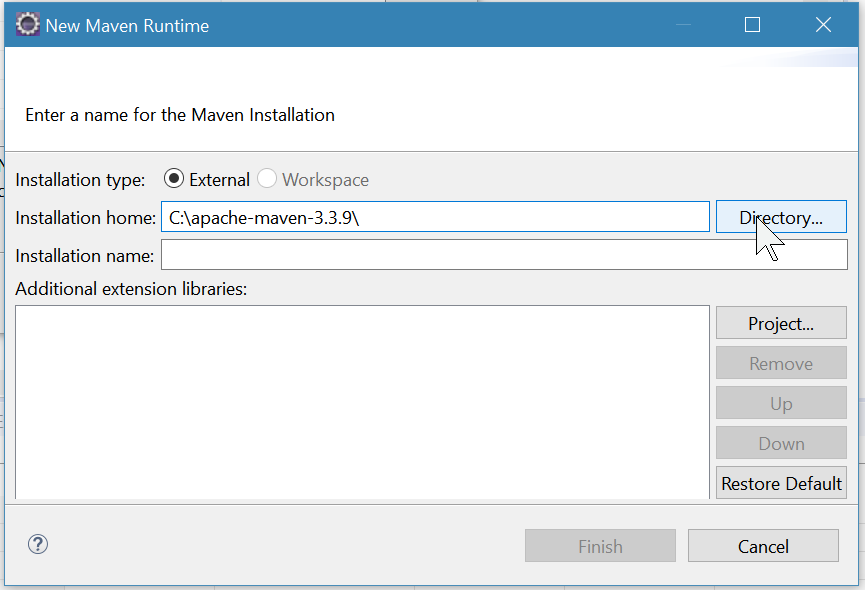
To configure Eclipse to use your system maven, go to ***Window*** > ***Preferences***



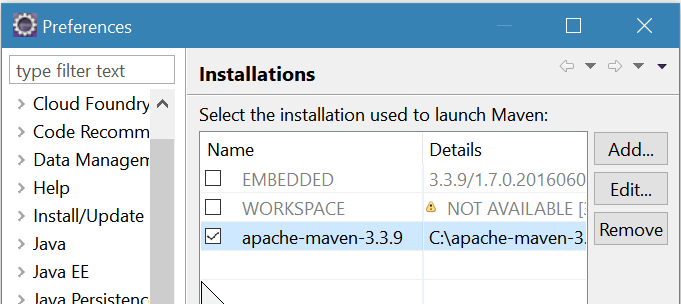
Next, navigate to the ***Maven*** > ***Installations*** section of the Preferences Window.



To add a local Maven installation, you can click ***Add…*** and browse to the directory of your Maven Installation.



Once you click “Finish” you can activate your Maven Installation by checking the box next to it in the Preferences Window.



Once you’ve checked your Maven installation in the Options, click ***Apply*** and then ***OK***

# Getting the Demo Code and opening in your IDE

## Eclipse

## NetBeans

## Generic Text Editor (advanced)

# Running the Demo Code

## Eclipse

## NetBeans

## Generic Text Editor

1. There is some debate within the developer community as to the ‘purity’ of Java’s Object Orientation, mostly because of the primitive types not inheriting from *Object*. [↑](#footnote-ref-1)
2. An Abstract Class Implementing an Interface does not have to implement all the methods, since an Abstract Class is not considered to be “Concrete” [↑](#footnote-ref-2)