

# **JAVA WITH ECLIPSE: SETUP & GETTING STARTED**

# Topics in This Section

- **Measuring Java popularity**
- **Installing Java**
- **Installing and configuring Eclipse**
- **Importing sample projects**
- **Executing desktop programs from Eclipse**
- **Executing Java programs manually**
- **Using Beanshell**
- **Using Eclipse shortcuts**

# Java SE vs. Java EE

- **Java SE (Standard Edition)**
  - “Core” Java
  - Java version used in this course
- **Java EE (Enterprise Edition)**
  - Same core language, but adds in many libraries for Web apps and other enterprise tasks
    - For tutorials on building Web apps in Java, please see JSF and PrimeFaces tutorials at [coreservlets.com](http://coreservlets.com)
  - Many or most real-life deployments start with Java SE and then get a server that is bundled with the needed Java EE libraries
    - So, most developers download Java SE, not Java EE, even if they will be doing Web or enterprise applications

# Java SE Versions

- **Latest Java SE version**

- Latest is Java 8; Java 8 should be used for almost all new projects.
  - Java 8 was final in March 2014. This tutorial covers general Java programming integrated with Java 8, but for just the Java-8-specific topics, see <http://www.coreservlets.com/java-8-tutorial/>.
  - Java 9 release scheduled for March 2017. Onsite Java 9 training from [coreservlets.com](http://www.coreservlets.com) coming soon. Email [hall@coreservlets.com](mailto:hall@coreservlets.com) if interested.

- **Java SE naming conventions**

- Naming conventions are confusing
  - Java 8 == JDK 1.8
  - Java 7 == JDK 1.7
  - Java 6 == JDK 1.6
  - Java 5 == JDK 1.5
  - Java 2, version 1.4 == JDK 1.4

# Features of Recent Java Versions

- **Java 5**
  - Major update. Generics, varargs, printf, @Override, new “for” loop.
- **Java 6**
  - Minor update. Updates to collections, Swing, etc.
- **Java 7**
  - Medium update. Fork/join framework, diamond operator, Strings in switch statements, try-with-resources, updates to Swing (especially new look and feel).
- **Java 8**
  - Major update. Lambdas for functional programming. Streams for bulk operations. Final version March 2014.
    - See tutorial at <http://www.coreservlets.com/java-8-tutorial/>

# Which Java SE Version Should You Use?

- **Server-side applications**

- Use the latest Java version that your app server supports
  - JDK 1.5 – 1.8, depending on how old your server is
- If you can choose, use JDK 1.8 (but 1.7 still moderately common as of mid-2016)

- **Desktop apps**

- For best power and speed, use Java 8 (aka JDK 1.8)
  - Even old projects can probably run on Java 8 unchanged

- **Android phone apps**

- Through Marshmallow, only Java 6 supported. Android Nougat to support Java 8.

- **Browser apps (Applets or Java WebStart)**

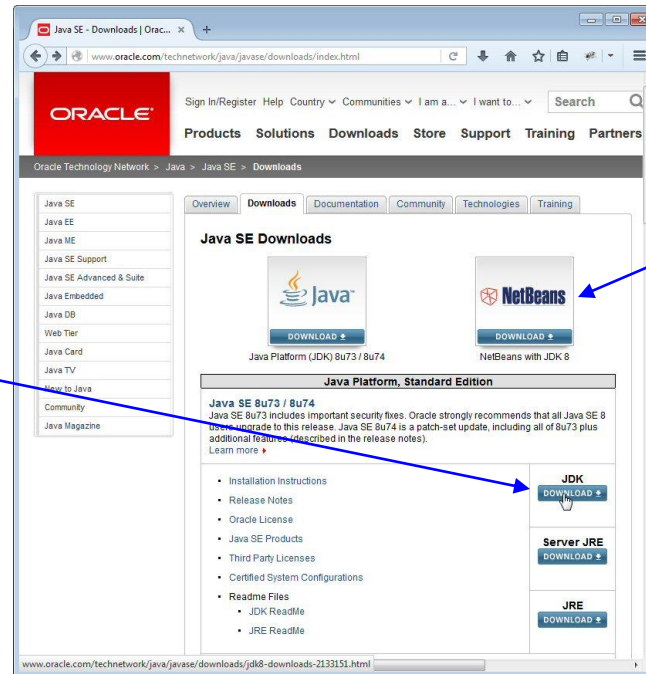
- In recent browsers, Java must be explicitly enabled
  - For intranet apps, use Java 8. Rarely used for internet apps.

# Installing Java SE (Standard Edition)

- **Install Java**

<http://www.oracle.com/technetwork/java/javase/downloads/>

Use this version. The “JDK – Java Development Kit” includes compiler for .java files, whereas the “JRE – Java Runtime Environment” is only for executing prebuilt .class files.



This tutorial uses Eclipse, but if you prefer the NetBeans environment, it is very easy to adapt the instructions to that development environment. So, if you prefer NetBeans or your organization has standardized on it, use this download instead of (not in addition to) the one below.

- **Bookmark the Java API (“JavaDocs”)**

- <http://docs.oracle.com/javase/8/docs/api/>
- <http://docs.oracle.com/javase/7/docs/api/> (if you need old version)
  - This is the most important Java reference for developers
  - Eclipse integrates this API, but a separate link is still good

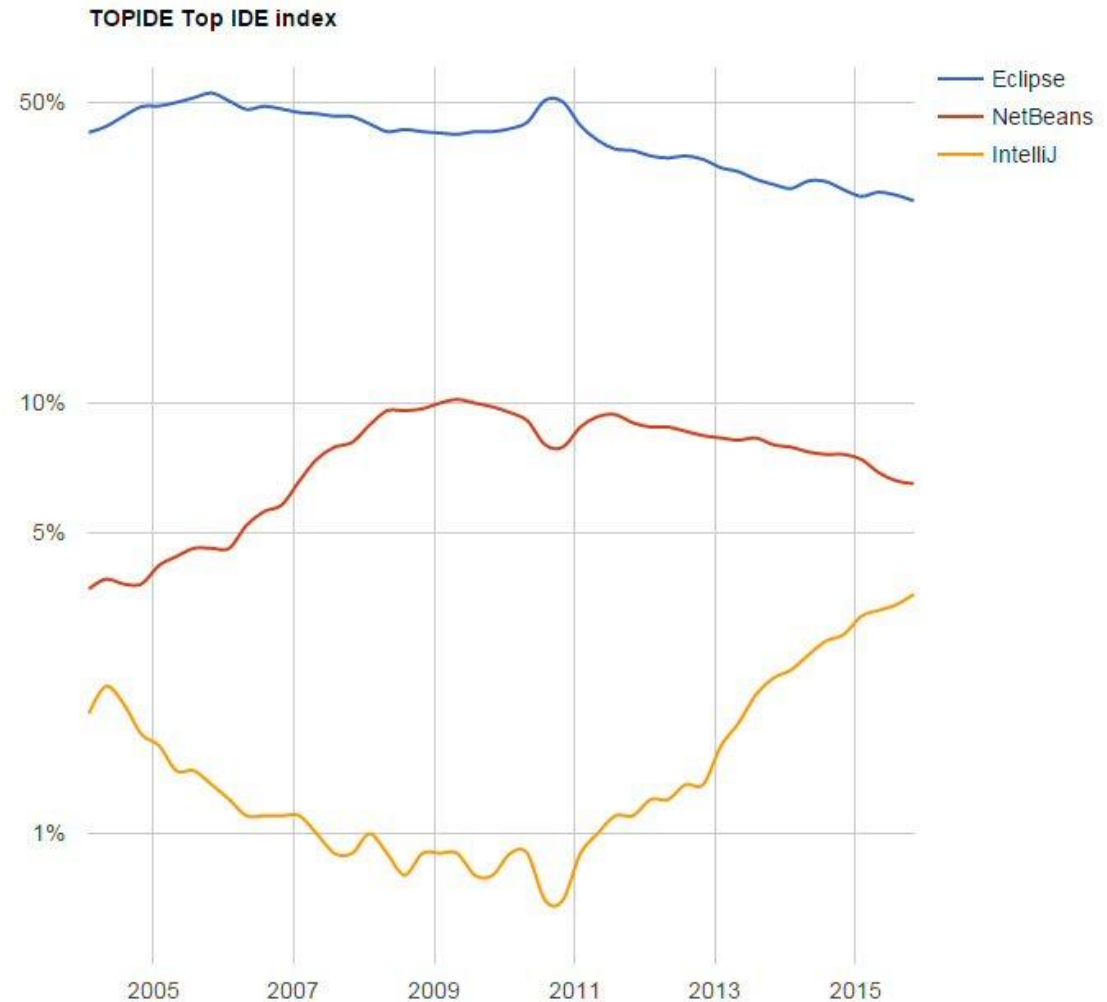
# Installing Eclipse

- **Overview**

- Eclipse is a free open source IDE. Support for Java, Android, HTML, CSS, JavaScript, C++, PHP, JSF, servlets, JSON, and more.
  - <http://eclipse.org/downloads/>
  - Choose “Eclipse IDE for Java EE Developers”

- **Features**

- Checks your syntax as you type
- Automatically compiles every time you save file
- Many tools: refactoring, debugging, server integration, templates for common tasks, etc.
  - Low learning curve:  
*beginners can use Eclipse without knowing these tools*



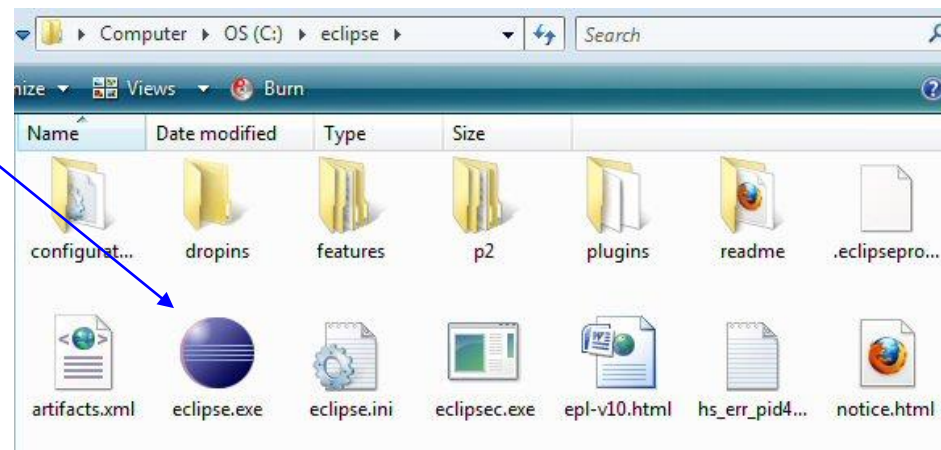
NetBeans and IntelliJ are two other popular IDEs (Integrated Development Environments) for Java. There is no agreement on which is better, but there is clear empirical evidence on which is more widely used.

Graph from <http://pypl.github.io/IDE.html>. Note the log scale.



# Running Eclipse

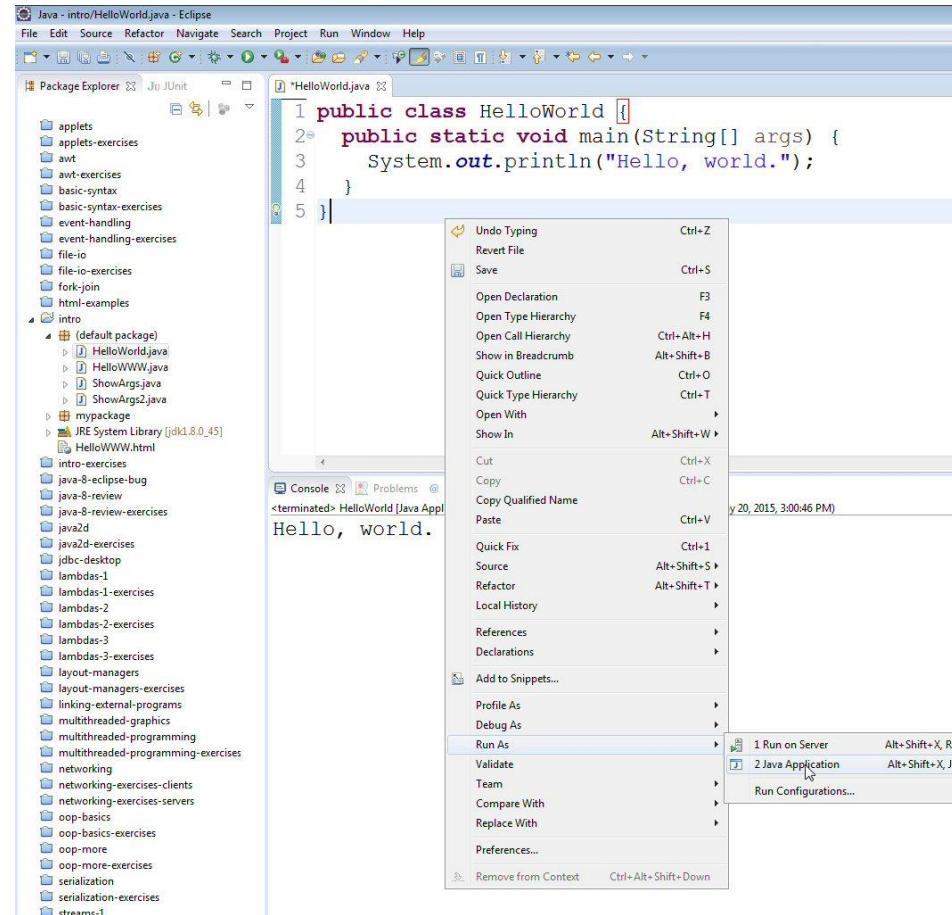
- **Use installer (Mars and later) or just unzip downloaded file**
  - Call the folder you unzip into “installDir”
- **Double click eclipse.exe (Mac/Linux similar)**
  - From *installDir/bin*
    - Pic is for Windows, but Mac and Linux are similar
- **Click on “Workbench” icon**
  - Next time you bring up Eclipse, it will come up in workbench automatically
- **Shortcut**
  - Many developers put Eclipse link on their desktop
    - R-click eclipse.exe, Copy, then go to desktop, R-click, and Paste Shortcut (not just Paste!)



# Eclipse: Running Programs

- **Executing program from existing project**

- Open existing project
- Double click Java file to bring it up in editor
- R-click anywhere in code
- Select Run As → Java Application
- Output goes in Console at bottom

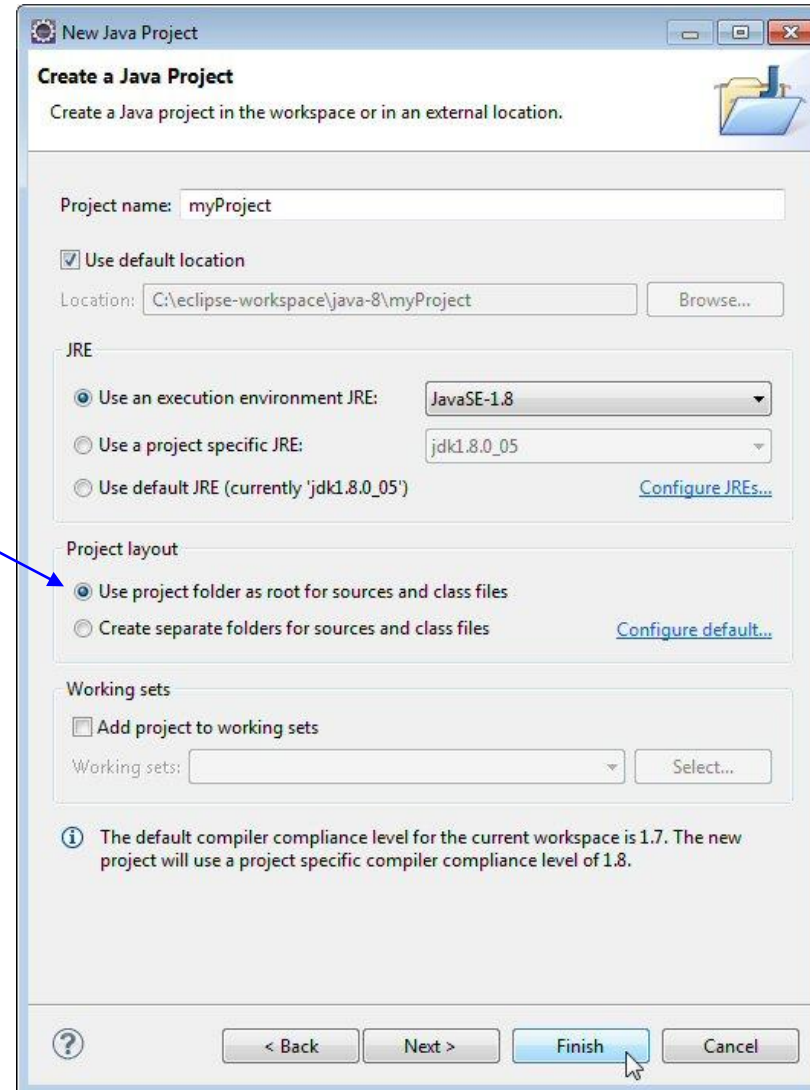


- Note: Class must have a “main” method – this is explained in the upcoming basic syntax section

# Eclipse: Making Projects

- **Main steps**

- File → New → Project → Java → Java Project
  - Pick any name
- If you plan to run from command line
  - Choose sources/classes in same project folder



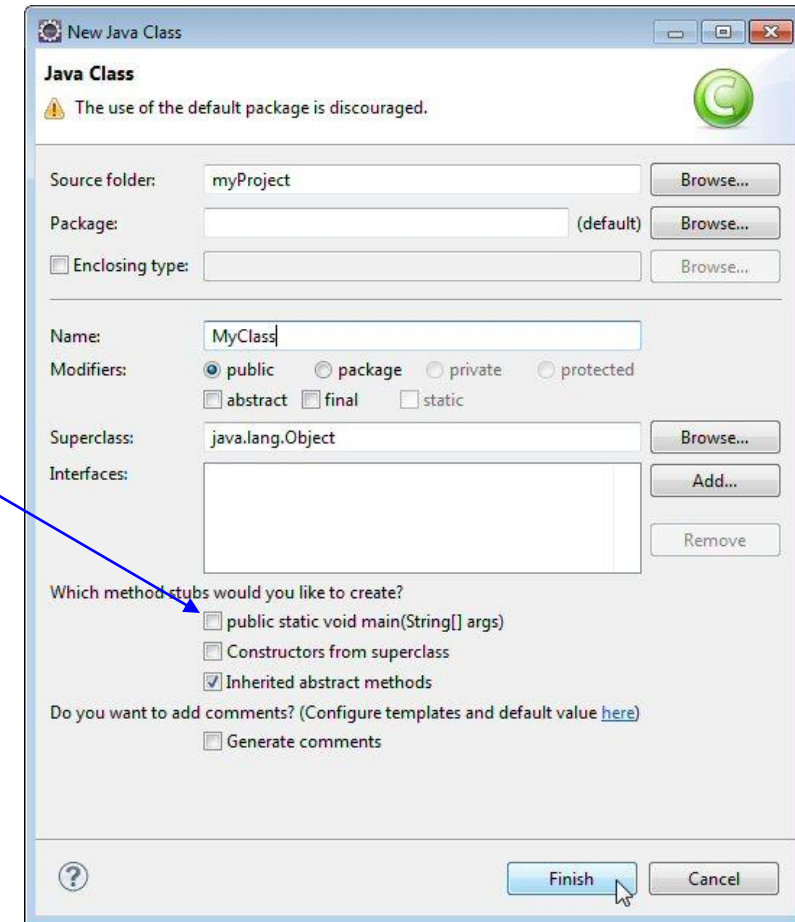
# Eclipse: Creating Classes

- **Main steps**

- R-click on project → New → Class
- Choose a capitalized class name (e.g., Class1 or MyFirstClass)
  - You can have Eclipse make “main” when class is created, but easier to use shortcut to insert it later
  - Eventually you will make package (subdirectory) first, then put class there  
Packages explained in upcoming section

- **Alternative**

- Can also copy/rename existing class



# Creating and Running Program

- **Create the .java file**
  - Write and save a file (say **Test.java**) that defines public class **Test**
    - Other than “real” Java IDEs (e.g., Eclipse, NetBeans, IntelliJ IDEA), there are a number of text editors (e.g., TextPad, UltraEdit, vi, emacs) with good Java support.
  - File and class names are case sensitive
- **Compile the the .java file**
  - Compile Test.java
    - > javac Test.java**
    - This step creates a file called Test.class
- **Run the .class file**
  - > java Test**
  - This step assumes your class has “main” method

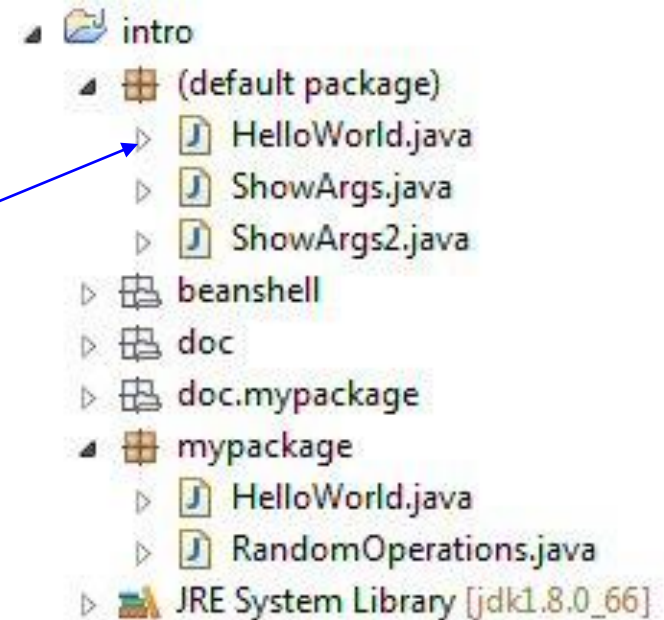
# Installing Sample Projects

- **Code from all tutorials is available online**
  - <http://courses.coreservlets.com/>
    - Click on Java tutorial on top left of page
- **Import project into Eclipse**
  - Click on appropriate tutorial section
  - Download ZIP file
    - The one for this section is called “intro”
  - Start Eclipse and go to Workbench
  - File → Import → General → Existing Projects into Workspace → Select archive file (not “Select root directory”).
    - Then browse to ZIP file you downloaded, OK, Finish

# Basic Hello World Application

- **File HelloWorld.java:**

```
public class HelloWorld {  
    public static void main(String[] args) {  
        System.out.println("Hello, world.");  
    }  
}
```



- **Notes**

- “Application” is lingo for a stand-alone Java program, i.e., an application is a Java class that contains “main”
  - Most Java classes do not contain “main”, but only those that contain “main” can be *directly* executed



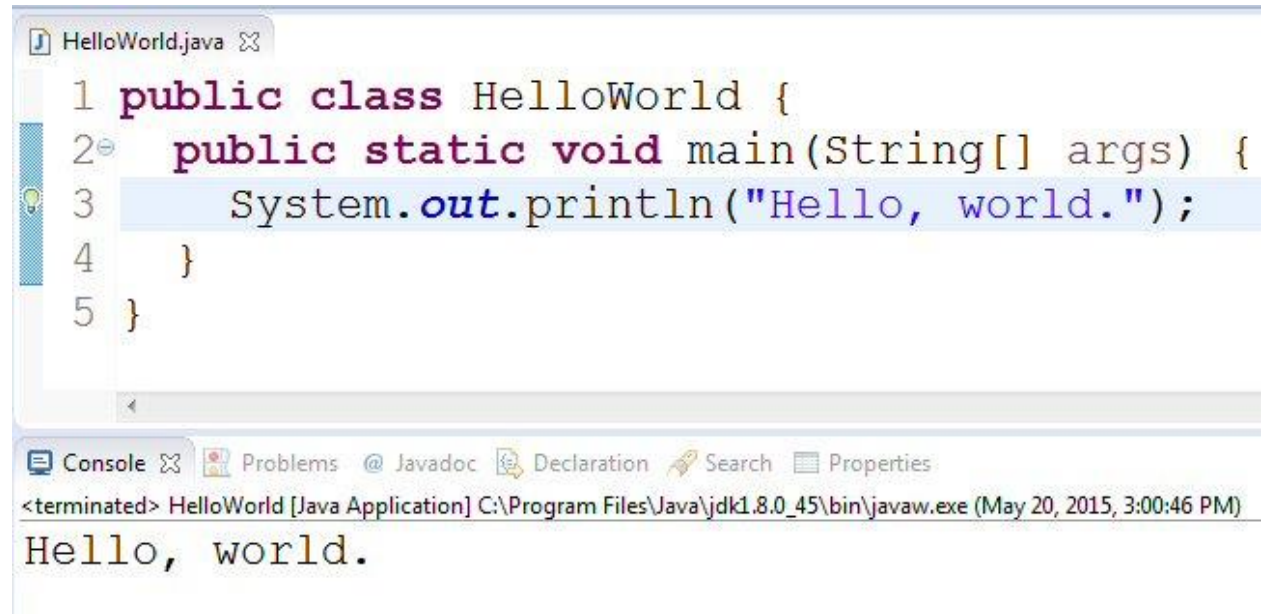
# Executing HelloWorld

- **In Eclipse (recommended)**

- Compiling
  - Done automatically whenever you save a file
- Executing
  - R-click inside window, then Run As → Java Application
  - You can also click green arrow at top of Eclipse
- Output (see “Console” tab at bottom)
  - Hello, World

- **Manually (rare)**

- Compiling
  - `javac HelloWorld.java`
- Executing
  - `java HelloWorld`
- Output
  - Hello, World



The screenshot shows the Eclipse IDE interface. The top editor window displays the code for `HelloWorld.java`:

```
1 public class HelloWorld {  
2     public static void main(String[] args) {  
3         System.out.println("Hello, world.");  
4     }  
5 }
```

Below the editor, the 'Console' tab is active, showing the output of the program:

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jdk1.8.0_45\bin\javaw.exe (May 20, 2015, 3:00:46 PM)  
Hello, world.
```



# A Few Eclipse Tricks

- **Making a new project**
  - File → New → Project → Java → Java Project
- **Making new package**
  - R-click project, New → Package
- **Making a new class**
  - R-click package, New → Class
- **Autocompletion**
  - Type part of a class or method name, Control-Space
- **Inserting main method**
  - Type the word “main”, then Control-Space
- **Inserting `System.out.println`**
  - Type the word “sysout”, then Control-Space
- **Renaming a class, variable, or method**
  - Select class, variable, or method, R-click, Refactor → Rename
    - Will also change all places that refer to it

# Summary

- **Downloading Java**
  - <http://www.oracle.com/technetwork/java/javase/downloads/>
- **Bookmarking the Java API**
  - <http://docs.oracle.com/javase/8/docs/api/> (or .../7/...)
- **Downloading Eclipse**
  - <http://eclipse.org/downloads/>
- **Downloading sample projects**
  - <http://www.coreservlets.com/>
    - Click on Java Programming tutorial on top left
  - Import with File → Import → Existing Projects ...
- **Executing a class that has “main”**
  - R-click in code, Run As → Java Application