Java for Inadequate Individuals (Vol. 2)

Chris Kannmacher
Andrew Dennison
Braxton Reichard

Table of Contents

Chapter 1

Introduction

Programming isn't overnight

Before we get started, just a heads up. This is something you will need to practice on your own and learn through your own independent projects. But we will love you like COD players love camping, and we will help you through this. Most of us taught Java to ourselves, so I recommend spending some time at home to work on this.

Google is your best friend

There is a lot to coding in general, so much that we might miss some in this book. You are going to have to do a lot of research to find out how to do whatever it is you are trying to do. When you are stuck and you don't know what to do and this book does not answer your problem, then google it. There are so many resources out there to help you, all you have to do is find them.

Wrap your head around it!

There is one thing that must be made clear.... If you do not understand something please re-read the book. If you just continue to read and do without understanding, you will get nothing from this experience. If you feel confident that you could explain the program to your friend who possibly does not know about programming, then you are ready to move on! Also, if you do not understand something and no matter how many times you read the book, just follow the last paragraph.

Have fun y'all

Coding is very rewarding if you have the patience. You can do so much with it. Like, I can think of, like, ***this - - - many*** things off the top of my head.

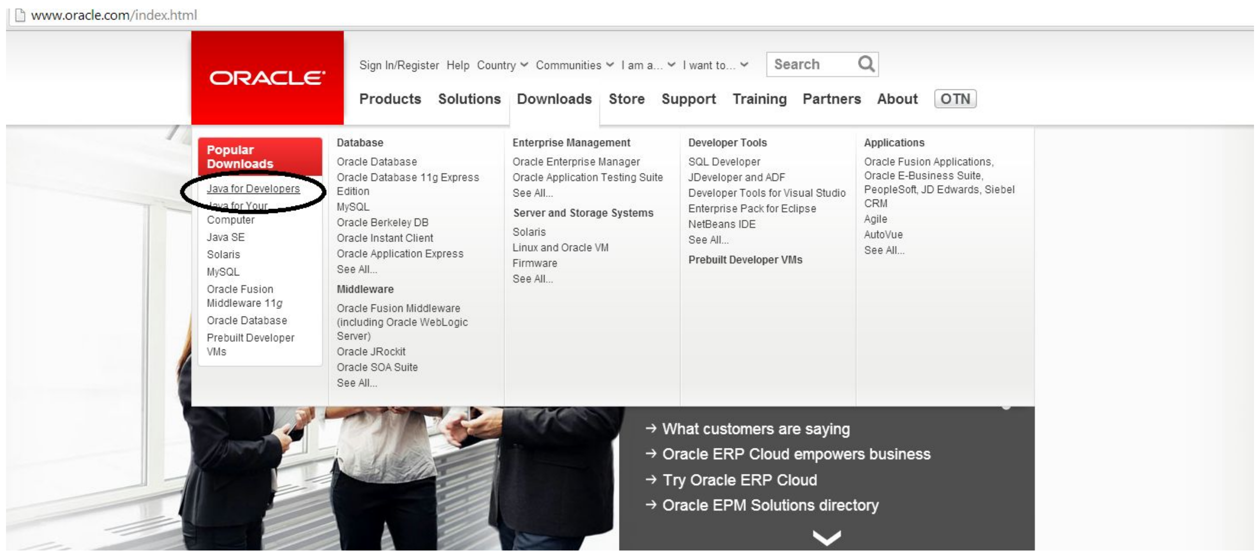
Setting Up Java

Downloading Java

*** Note: If you are doing this at school they may block the website so an alternate source of Wi-fi may be required.**

***Note: This is only for Windows. Macintosh is not the best in our opinion.**

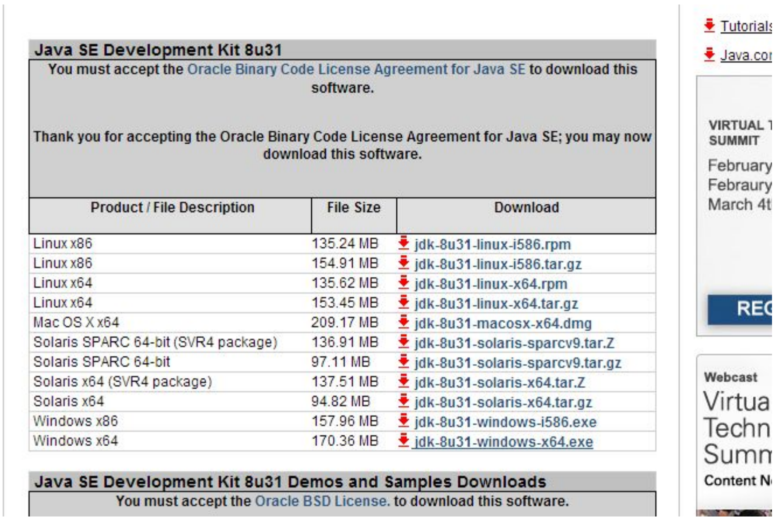
1.) The first step in installing your java **IDE** (**I**ntegrated **D**evelopment **E**nvironment) is to go to **www.Oracle.com**. When you have completed this task you must hover your mouse over the downloads button and select **Java for Developers**.


www.oracle.com/index.html
ORACLE
Sign In/Register Help Country Communities I am a... I want to... Search
Products Solutions Downloads Store Support Training Partners About OTN
Popular Downloads
Java for Developers
Java for Your Computer
Java SE
Solaris
MySQL
Oracle Fusion Middleware 11g
Oracle Database
Prebuilt Developer VMs
Database
Oracle Database
Oracle Database 11g Express Edition
MySQL
Oracle Berkeley DB
Oracle Instant Client
Oracle Application Express
See All...
Middleware
Oracle Fusion Middleware (including Oracle WebLogic Server)
Oracle JRockit
Oracle SOA Suite
See All...
Enterprise Management
Oracle Enterprise Manager
Oracle Application Testing Suite
See All...
Server and Storage Systems
Solaris
Linux and Oracle VM
Firmware
See All...
Developer Tools
SQL Developer
JDeveloper and ADF
Developer Tools for Visual Studio
Enterprise Pack for Eclipse
NetBeans IDE
See All...
Prebuilt Developer VMs
Applications
Oracle Fusion Applications, Oracle E-Business Suite, PeopleSoft, JD Edwards, Siebel CRM
Agile
AutoVue
See All...
→ What customers are saying
→ Oracle ERP Cloud empowers business
→ Try Oracle ERP Cloud
→ Oracle EPM Solutions directory

2.) Click **Download JDK**, there should be a Java image for you to click on in order to download this package.

3.) Now you will be greeted with alot of downloads, please choose the one that is for your **OS** (**O**perating **S**ystem). Now it should be installing. In this version we will be installing windows 7 64 bit edition.

***Note:**
Please install the right version or it will not work!
Also if you do not know your version please click the windows Start Menu -> Computer-> System Properties and it should show your version in that window.



Java SE Development Kit 8u31

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.

Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u31-linux-i586.rpm
Linux x86	154.91 MB	jdk-8u31-linux-i586.tar.gz
Linux x64	135.62 MB	jdk-8u31-linux-x64.rpm
Linux x64	153.45 MB	jdk-8u31-linux-x64.tar.gz
Mac OS X x64	209.17 MB	jdk-8u31-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	136.91 MB	jdk-8u31-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.11 MB	jdk-8u31-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.51 MB	jdk-8u31-solaris-x64.tar.Z
Solaris x64	94.82 MB	jdk-8u31-solaris-x64.tar.gz
Windows x86	157.96 MB	jdk-8u31-windows-i586.exe
Windows x64	170.36 MB	jdk-8u31-windows-x64.exe

Java SE Development Kit 8u31 Demos and Samples Downloads

You must accept the Oracle BSD License. to download this software.

4.) Now go into your **downloads section** of your computer and **run the .exe file** that you installed for your JDK package. And select run.

5.) Go through the installation process, **note** some machines may already have something similar installed and it could go through an re-install or update process, yet the steps we shall take shall will be very similar. When you are done with the installation it will ask if you wish to change the destination folder. Just choose a place in which you will remember it to be located. After this it should finish the installation process.

Overview:

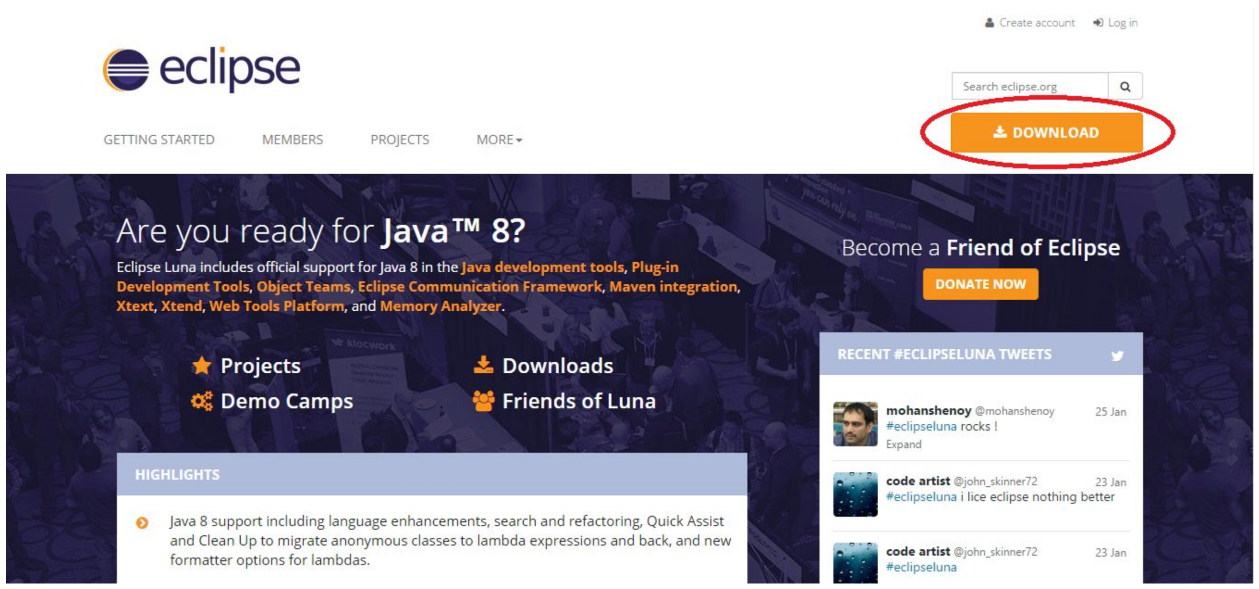
Congratulations you have managed the install process! Now we shall continue on so you can be the hacker typer in every programmer's dreams....

Installing Eclipse

You read that right

***Note: Download Eclipse. not Netbeans, though it is a good IDE as well.**

1.) Now, let's navigate to eclipse.org



Create account Log in

eclipse

Search eclipse.org

GETTING STARTED MEMBERS PROJECTS MORE

DOWNLOAD

Are you ready for **Java™ 8?**

Eclipse Luna includes official support for Java 8 in the **Java development tools**, **Plug-in Development Tools**, **Object Teams**, **Eclipse Communication Framework**, **Maven integration**, **Xtext**, **Xtend**, **Web Tools Platform**, and **Memory Analyzer**.

Projects Downloads
Demo Camps Friends of Luna

Become a **Friend of Eclipse**

DONATE NOW

HIGHLIGHTS

Java 8 support including language enhancements, search and refactoring, Quick Assist and Clean Up to migrate anonymous classes to lambda expressions and back, and new formatter options for lambdas.

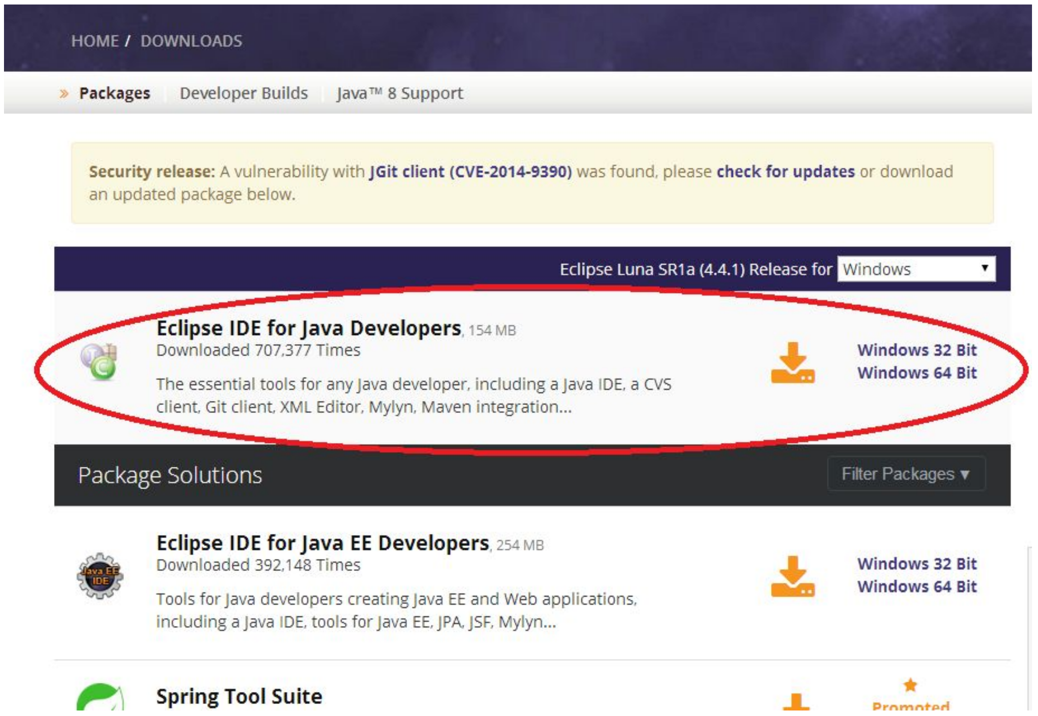
RECENT #ECLIPSELUNA TWEETS

mohanshenoy @mohanshenoy 25 Jan
#eclipseluna rocks !
Expand

code artist @john_skinner72 23 Jan
#eclipseluna i lice eclipse nothing better

code artist @john_skinner72 23 Jan
#eclipseluna

2.) Go to the big orange download button as shown above to begin. You will then see this screen.



HOME / DOWNLOADS

» **Packages** Developer Builds Java™ 8 Support

Security release: A vulnerability with **JGit client (CVE-2014-9390)** was found, please **check for updates** or download an updated package below.

Eclipse Luna SR1a (4.4.1) Release for Windows

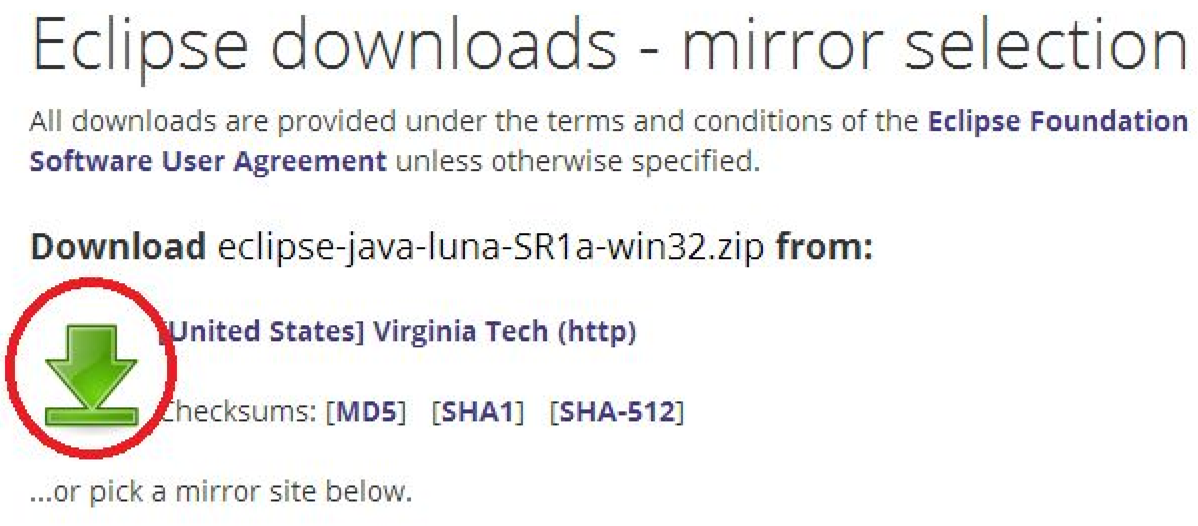
Eclipse IDE for Java Developers, 154 MB
Downloaded 707,377 Times
The essential tools for any Java developer, including a Java IDE, a CVS client, Git client, XML Editor, Mylyn, Maven integration...
Windows 32 Bit
Windows 64 Bit

Package Solutions Filter Packages

Eclipse IDE for Java EE Developers, 254 MB
Downloaded 392,148 Times
Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...
Windows 32 Bit
Windows 64 Bit

Spring Tool Suite
Promoted

3.) Go into the Eclipse IDE for Java Developers icon and choose either Windows 32 or 64 bit. If you use another operating system, go to the drop-down box in the purple row and select your OS.



Eclipse downloads - mirror selection

All downloads are provided under the terms and conditions of the **Eclipse Foundation Software User Agreement** unless otherwise specified.

Download eclipse-java-luna-SR1a-win32.zip **from:**

[United States] Virginia Tech (http)

Checksums: **[MD5]** **[SHA1]** **[SHA-512]**

...or pick a mirror site below.

4.) Click the green arrow to begin the download. After you press that button, you will download an exe to install eclipse. Once you open that file, extract it to wherever you like. Inside the extracted file will be eclipse.exe. Run that.

Once that is open, you can really begin your journey to being a programming genius.

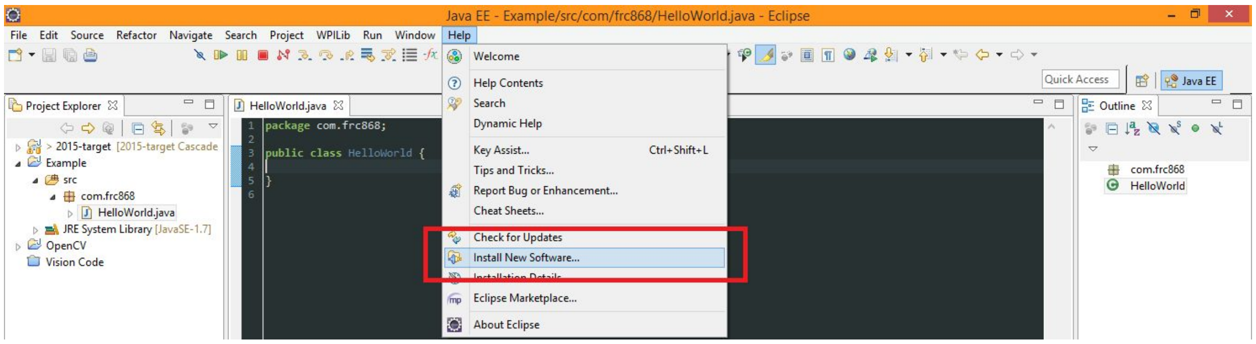
Overview:

Now you have Eclipse so you can code stuff and make awesome programs.

Installing FRC Plugins

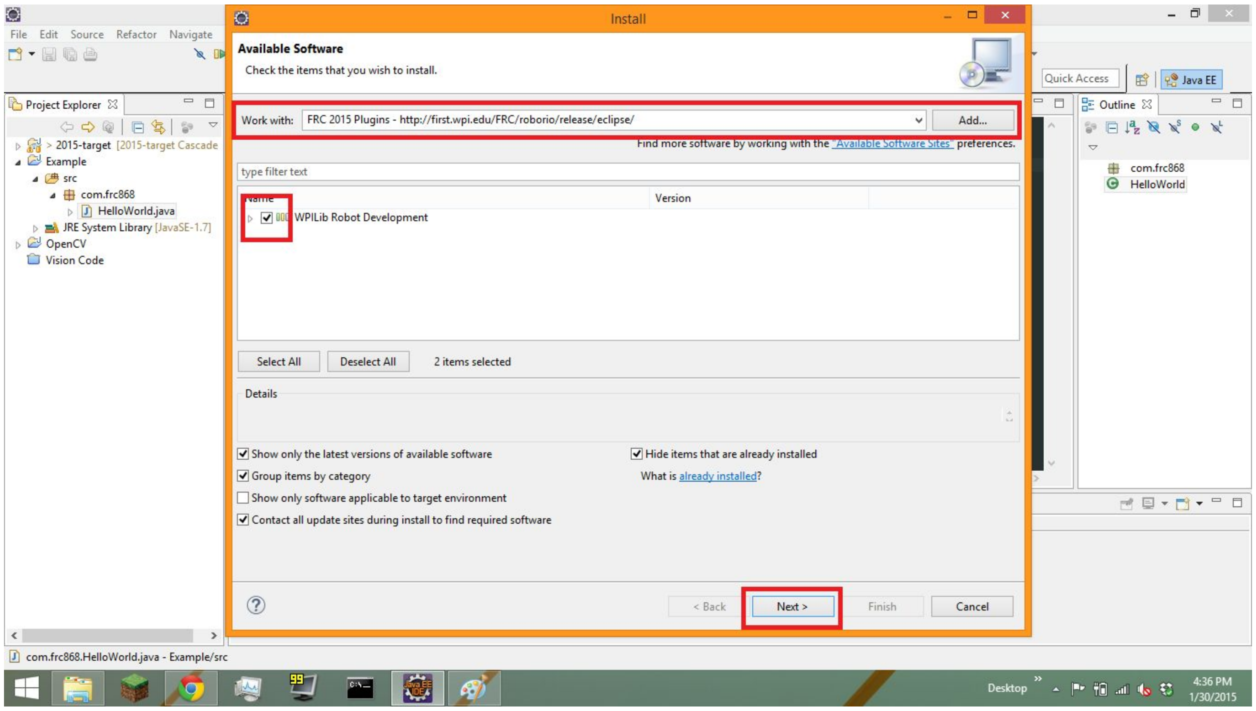
PSYCH

1.) We've got one more thing to do before we begin coding. Open up Eclipse and navigate to **Help -> Install New Software**.



```
package com.frc868;

public class HelloWorld {

}
```

2.) From there, type in the following link in the address bar and click next:
http://first.wpi.edu/FRC/roborio/release/eclipse/
Be sure to check the box with **"WPLib Robot Development."**



Work with: FRC 2015 Plugins - http://first.wpi.edu/FRC/roborio/release/eclipse/

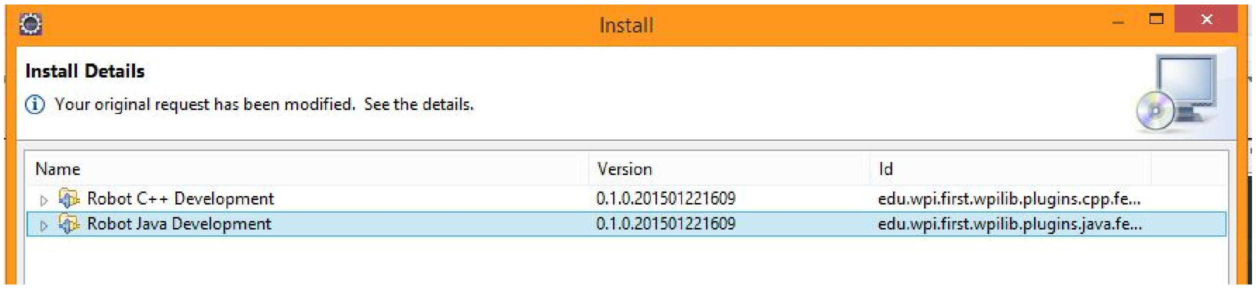
[✓] WPILib Robot Development

2 items selected

[✓] Show only the latest versions of available software
[✓] Hide items that are already installed
[✓] Group items by category
[ ] Show only software applicable to target environment
[✓] Contact all update sites during install to find required software

com.frc868.HelloWorld.java - Example/src

3.) Select Robot Java Development and click next.

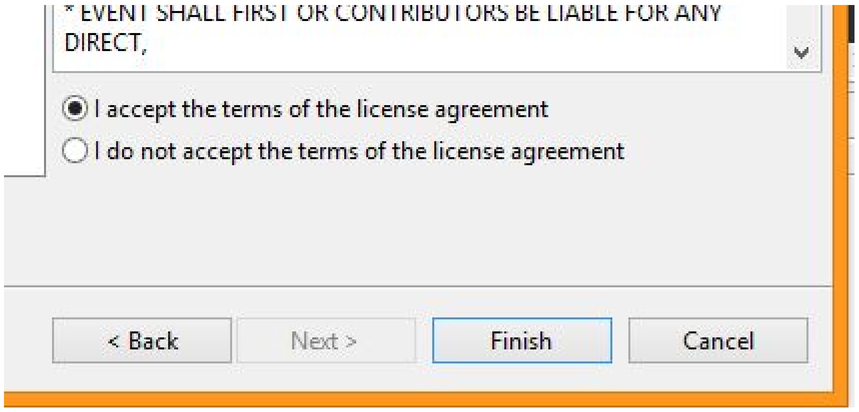


Install Details

Your original request has been modified. See the details.

Name	Version	Id
Robot C++ Development	0.1.0.201501221609	edu.wpi.first.wpilib.plugins.cpp.fe...
Robot Java Development	0.1.0.201501221609	edu.wpi.first.wpilib.plugins.java.fe...

Always click accept, especially when you didn't read the fine print. Trust me, there's nothing interesting there.



* EVENT SHALL FIRST OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT,

[✓ I accept the terms of the license agreement] I do not accept the terms of the license agreement

4.) Finish, and let the plugins download.

Now, we can seriously begin. No joke.

OVERVIEW:

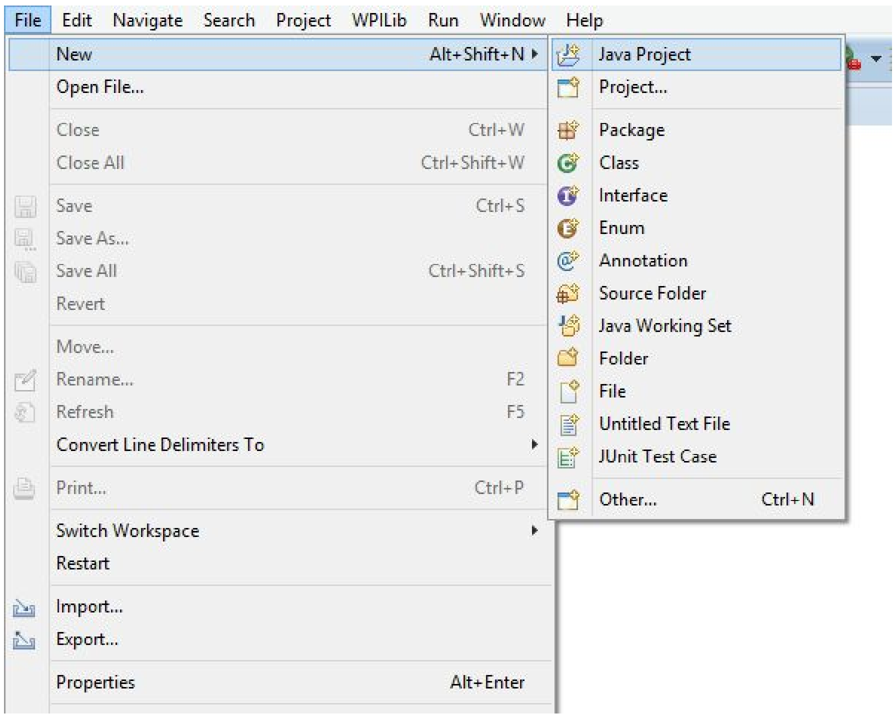
Y'all just installed FRC Plugins, and can use the super useful stuff contained within the Pit of 100 Trials ***cough*** er, plugins.

CHAPTER 2

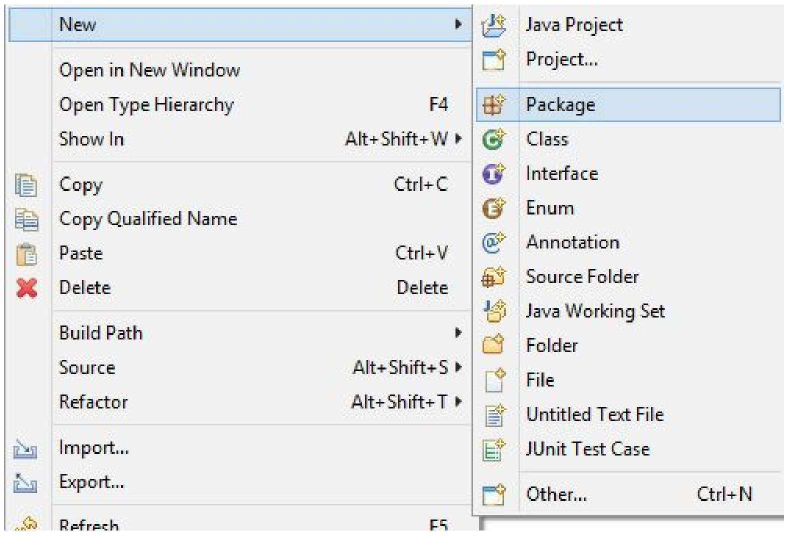
Intro to Programming

Creating a Project

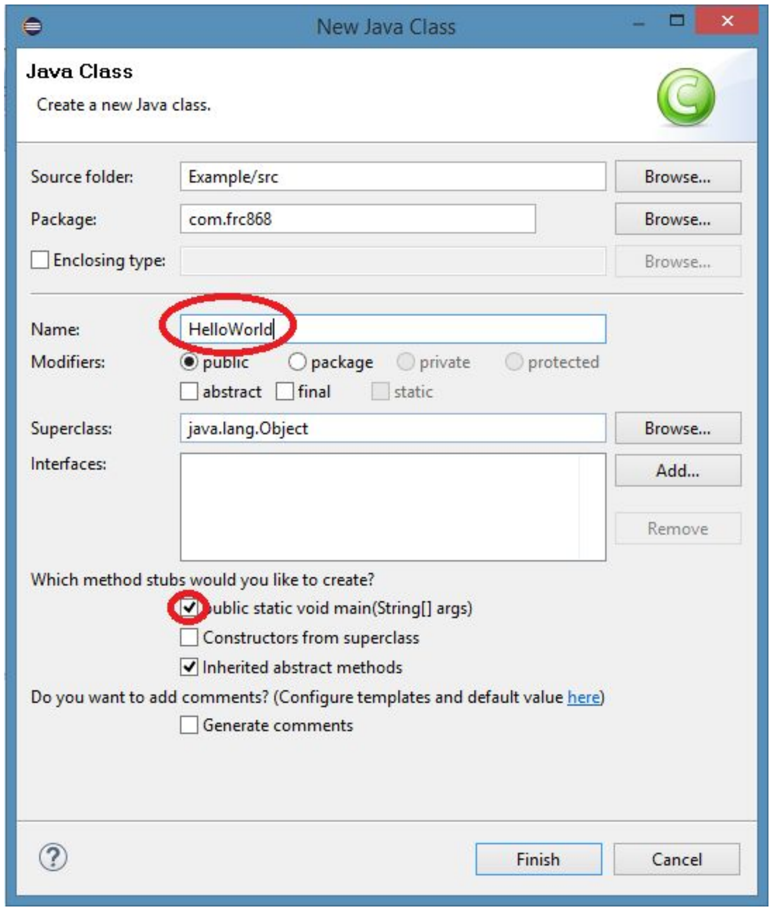
1.) To create a project you need to go to **File -> New -> Java Project**



2.) Then name it whatever you want. For this example we are going to call it **Example**. After it has been made, click the drop-down arrow and right click on the src file and click **New -> Package**



3.) Next, name the package whatever you want. For this example I will call it **com.frc868**. All that means is there is a main folder **com** and inside of that is another file **frc868**. You don't have to call it that, but we suggest adding your team name or number to the name. A good way to think of packages is that the project is a hard drive on a computer, then the packages are folders inside of that drive, and classes are documents where code is being written. Once you have the package go to **New -> Class**. This window will open.



Java Class

Create a new Java class.

Source folder: Example/src

Package: com.frc868

☐ Enclosing type:

Name: HelloWorld

Modifiers: ● public ○ package ○ private ○ protected

☐ abstract ☐ final ☐ static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

☑ public static void main(String[] args)

☐ Constructors from superclass

☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

☐ Generate comments

4.) Once this window pops up, name the class that you are going to make. For this example it will be called **HelloWorld**. Also, make sure that the method stub *public static void main(String[] args)* is checked. That is called the main method and it will be explained later. Once that is done click finish and you have a java class.

Primitive Data Types

Primitive data types is just a fancy name for the basic building blocks of java. A primitive data type is a variable type that comes with java and stores different types of data. For example, an **int** (short for **int**eger) stores any whole number. Here is a list of the other primitive data types:

Double : Stores any decimal (i.e. 0.543)

Int : Stores any whole number (i.e. 7) **NO DECIMAL**

Unfortunately, **int**'s have a range of what they can store. They can store in a ±$\mathbf{2^{31}}$ range. Thus we have a **long**. **Long**'s can store much more data than an **int**. In fact, there are several variations of **int**'s that store different amounts of data.

Byte : Stores numbers between **-128** and **127**

Short : Stores numbers between **-32,768** and **32,767**

Int : Stores numbers between $\mathbf{-2^{31}}$ and $\mathbf{2^{31} - 1}$

Long : Stores numbers between $\mathbf{-2^{63}}$ and $\mathbf{2^{31} - 1}$

The smaller the number, the less data it uses. However, the data usage difference is very minimal, so for this book we will always be using **int**.

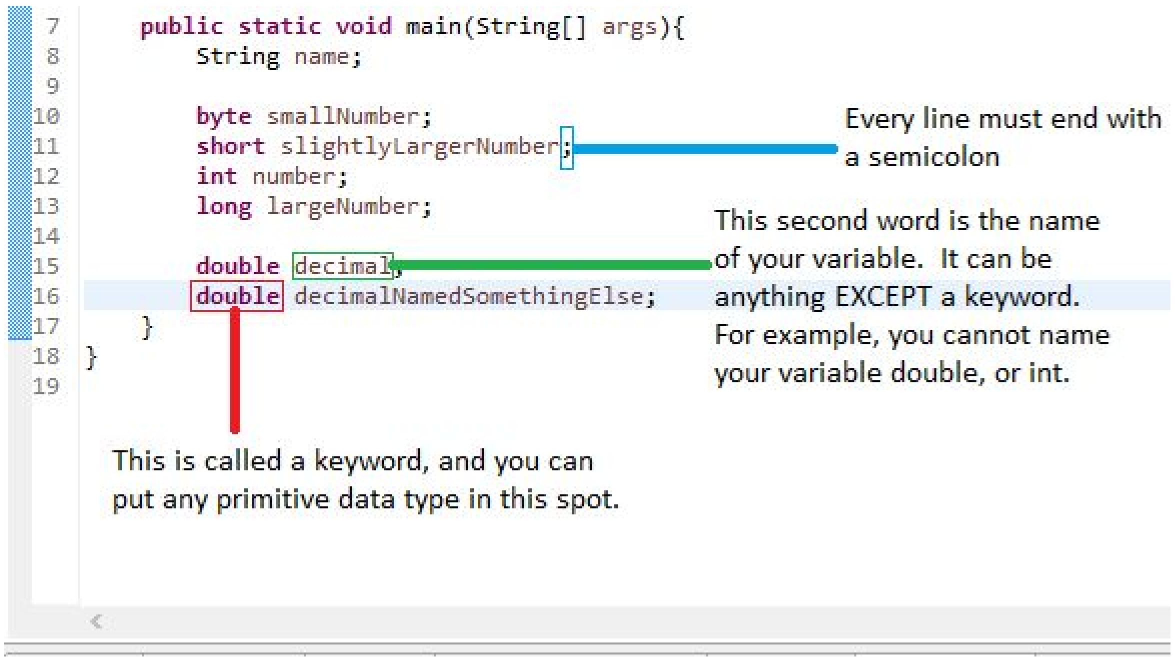
Also, a version of **double** is available called **float** , however, we will not be using it as it is inaccurate with decimals and isn't useful to us.

Another primitive data type is **boolean**. **Boolean**'s store only a true or false. For those of you who know what binary is, it is essentially binary where true is 1 and false is 0.

Finally, any text should be stored in a **String** or **Char**. **String**'s can store all data in any combination, for example, "TechHOUNDS 868" is considered one string. On the other hand, **Char**'s can store only one letter, for example, 'A' or 'd'.

//In code

The process to create a variable inside code is simple. Go inside the main method and type any data type, followed by a name, followed by a semicolon. For example:



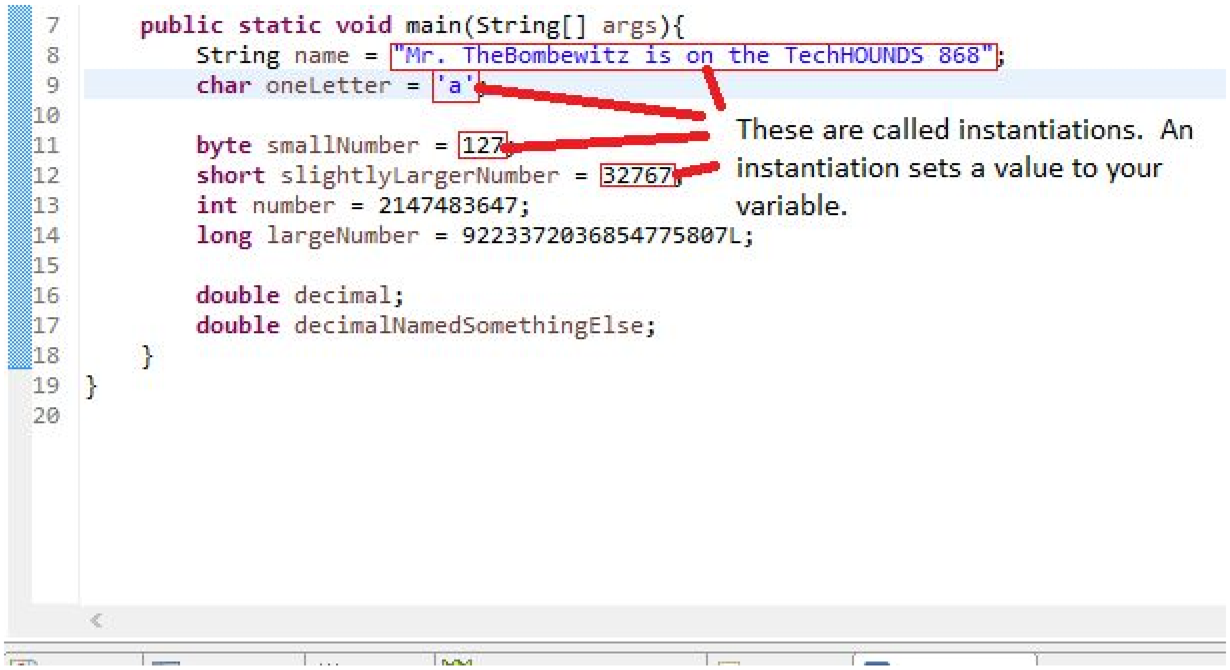
```
    public static void main(String[] args){
        String name;

        byte smallNumber;
        short slightlyLargerNumber;
        int number;
        long largeNumber;

        double decimal;
        double decimalNamedSomethingElse;
    }
}
```

Every line must end with a semicolon

This second word is the name of your variable. It can be anything EXCEPT a keyword. For example, you cannot name your variable double, or int.

This is called a keyword, and you can put any primitive data type in this spot.

***NOTE** ALL** of your lines must end with a semicolon. There are exceptions to this, but they are a bit more complicated and we won't discuss them until later.
This process has created a variable. However, you would of course like to store something in these variables! Just add an equals sign to the end, followed by the specific data. For example:

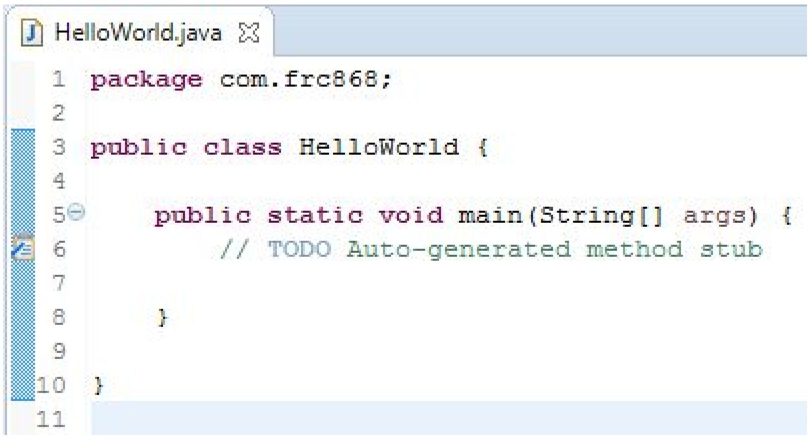


```
    public static void main(String[] args){
        String name = "Mr. TheBombewitz is on the TechHOUNDS 868";
        char oneLetter = 'a';

        byte smallNumber = 127;
        short slightlyLargerNumber = 32767;
        int number = 2147483647;
        long largeNumber = 9223372036854775807L;

        double decimal;
        double decimalNamedSomethingElse;
    }
}
```

These are called instantiations. An instantiation sets a value to your variable.

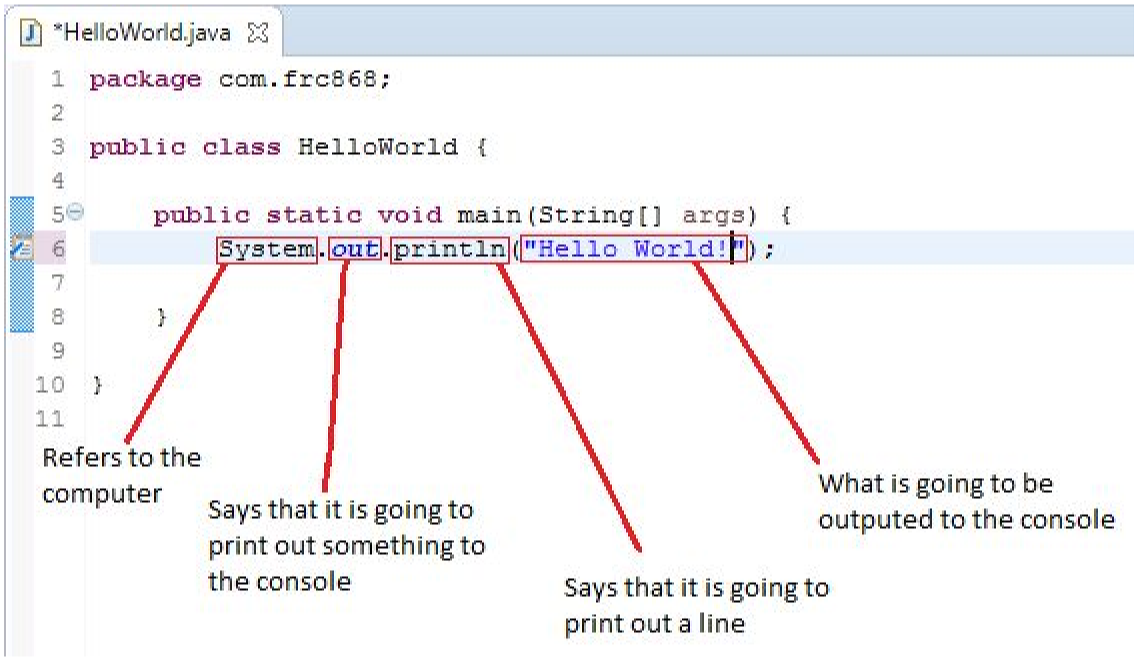
HelloWorld

It is time to create your first program in java. If you went along with the making a project section, then you have a project and a class named HelloWorld or whatever you called it. After it has opened, it should look like this.



```java
package com.frc868;

public class HelloWorld {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

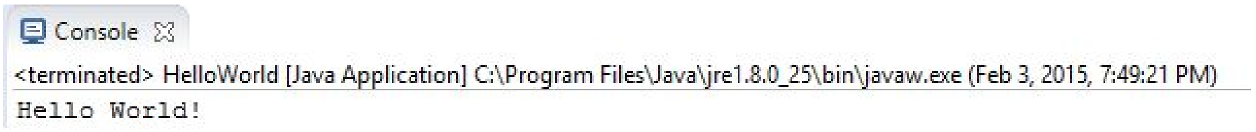
Once you have this, delete the "// TODO Auto-generated method stub" because you don't need it. After that, write the following line; "**System.out.println("Hello World!");**". It is broken down in the picture below.



*HelloWorld.java
Refers to the computer
Says that it is going to print out something to the console
Says that it is going to print out a line
What is going to be outputed to the console

```java
package com.frc868;

public class HelloWorld {

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

    }

}
```

If you run it this should output in the console below.

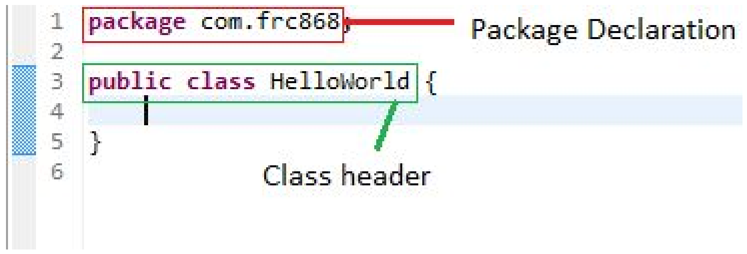


```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (Feb 3, 2015, 7:49:21 PM)
Hello World!
```

Remember those variables we talked about last lesson? You can use this **System.out.println** to see what is stored inside these variables! For example:

Classes

Classes are the core of java; They store all code. In fact, you created a class at the beginning of this chapter when you created your project. Think of a class as a chair. The chair looks nice on the outside, and has a function: to support you when you sit. But inside, the chair has various properties to it, such as how many legs it has, or what color wood it is. We call these properties fields; They store information about the class. The string you made two sections ago is an example of a field. It stores information, in this case text, about the class. You could make a string that stores the word "oak" to show the chair is made of oak wood. This concept might be a little hard to grasp at first, so don't get frustrated if it doesn't make sense.



```
1  package com.frc868;
2  
3  public class HelloWorld {
4  	
5  }
6  
```

This is an example of an empty class. The first line, the package declaration, says where your class is located. This can change depending on what you named your project at the beginning of the chapter, but it will always say package at the beginning and have a semicolon at the end.

The second line of code (on line 3) is a class header. This essentially declares a class.

Public has the same meaning as it does in English; It says that your class is public, or is visible to other classes. This isn't important now but will be later.

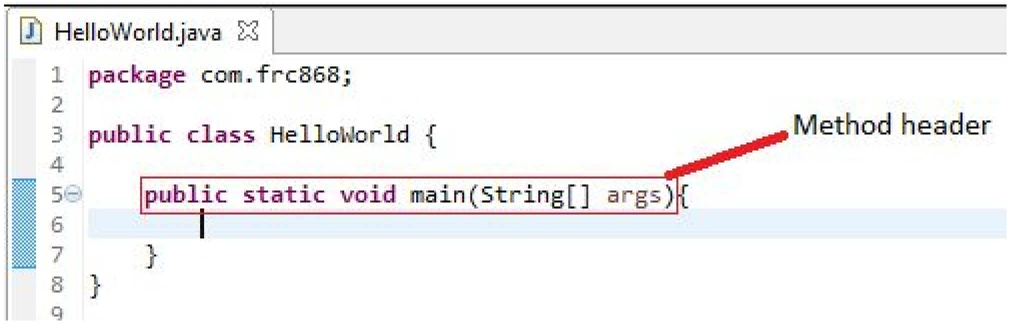
The second word, class, is very important. It says that this is a class, similar to how the package declaration has package in it.

The third word is whatever you named your class. The important thing here is that you type the exact same name in your code as what you named it when you made your class.

Also, you must have the brackets following this statement. The brackets specify where the code is; Whenever you run this class, it will only look inside of those brackets for your code.

Methods

Methods are a way of further dividing code inside a class. They make it easier to read code and provide a quick summary of the code contained within. For example, every class needs a main method:



HelloWorld.java

```
package com.frc868;

public class HelloWorld {

    public static void main(String[] args){

    }
}
```

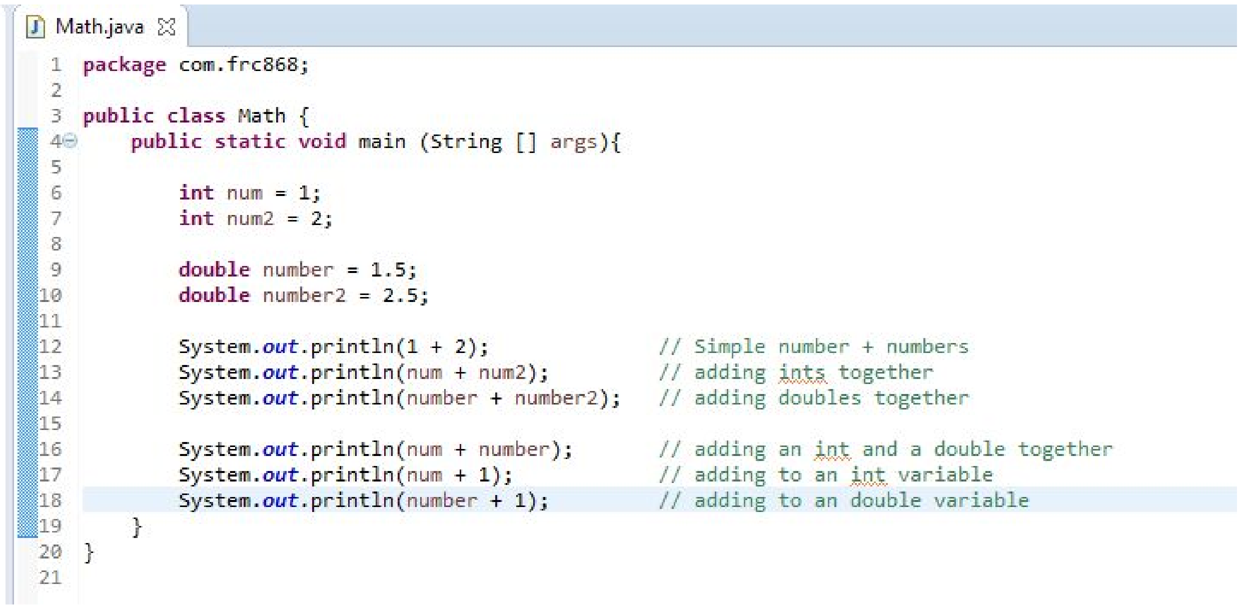
Remember that visibility we just talked about, where classes are declared

OBJECTS

Objects are an instance of a class. When you made strings, you made an instance of the String class that comes with java.

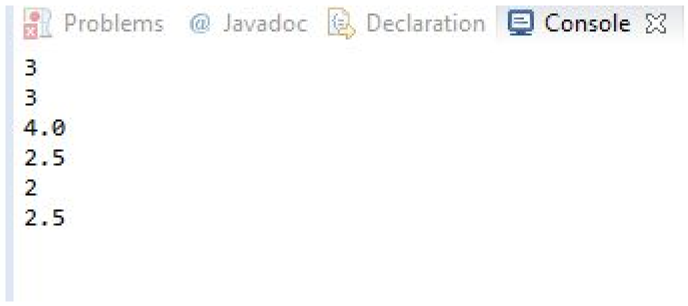
Simple Math:

Now that you know variables let us begin by creating a Math class inside your package and type out the code displayed below:



```java
package com.frc868;

public class Math {
    public static void main (String [] args){

        int num = 1;
        int num2 = 2;

        double number = 1.5;
        double number2 = 2.5;

        System.out.println(1 + 2);              // Simple number + numbers
        System.out.println(num + num2);         // adding ints together
        System.out.println(number + number2);   // adding doubles together

        System.out.println(num + number);       // adding an int and a double together
        System.out.println(num + 1);            // adding to an int variable
        System.out.println(number + 1);         // adding to an double variable
    }
}
```

As you can see above we are using both **integers** and **Doubles** and combining both of them in different combinations in order to produce a number. As well as this you can produce a double by adding an int and a double together, but you may not use any doubles in the creation of an int.



```
3
3
4.0
2.5
2
2.5
```

As you can see in the output, you may add a variable to a normal number in order to produce a number. As well as this you can add int and doubles in order to produce an double, yet you may not be able to produce an int from adding a double to an int, or an double to a double.

Strings:

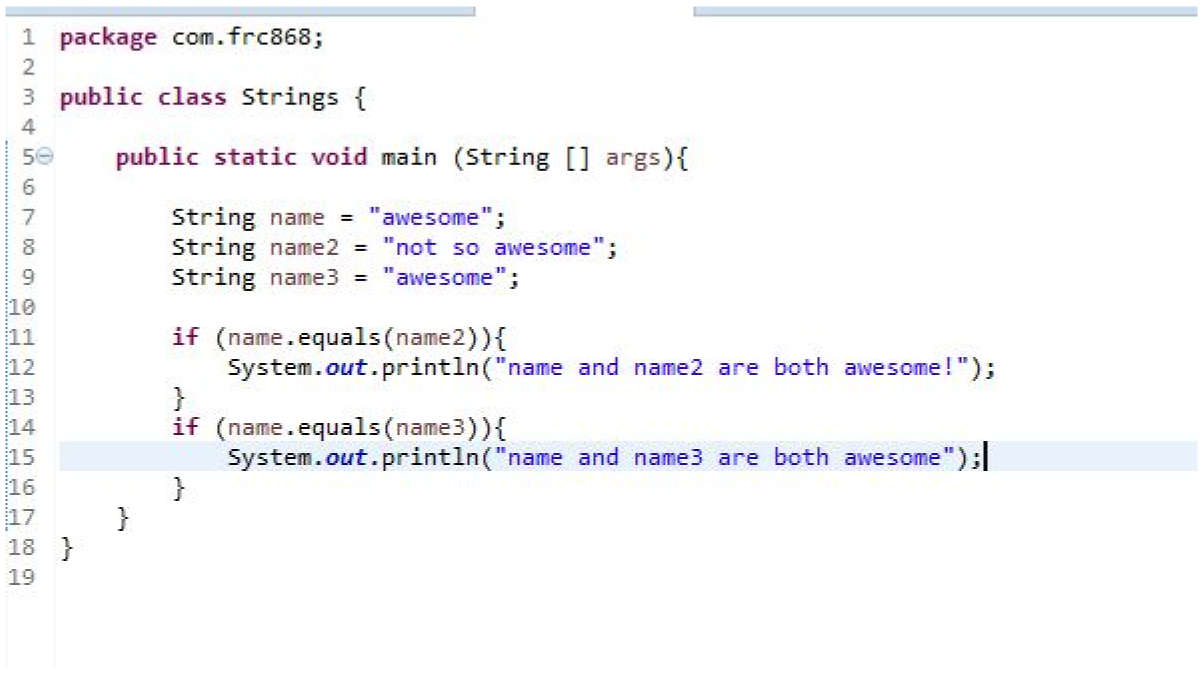
String are the basis for implementing text in a program, yet with this comes new rules you must learn in order to truly utilize these variables.

First off let us explain how to implement a string……

String name = "Awesome";

As you can see that is how you declare a String, by defining it as one, then by declaring what it should be stored as, "**name"** what we use in this case and it stores the text "**Awesome**".

Now that you have how to store it down, let us practice by storing two different variables and see if they equal each other….



```java
package com.frc868;

public class Strings {

    public static void main (String [] args){

        String name = "awesome";
        String name2 = "not so awesome";
        String name3 = "awesome";

        if (name.equals(name2)){
            System.out.println("name and name2 are both awesome!");
        }
        if (name.equals(name3)){
            System.out.println("name and name3 are both awesome");
        }
    }
}
```

As you can see **<random String>.equals(<random different String>)** checks if these two Strings are equal to each other. Now we shall move on to learn about both **.substring()** and **.length** methods.

"Advanced" Math:

Now we will further analyze Math by changing variables, as well as this we will be adding **loops** and handling user input in our math class in order to create a calculator program!

"Classes 101":

Modern languages such as Java and C++ use a system called *object orientatioon to*