I. Download necessary softwares

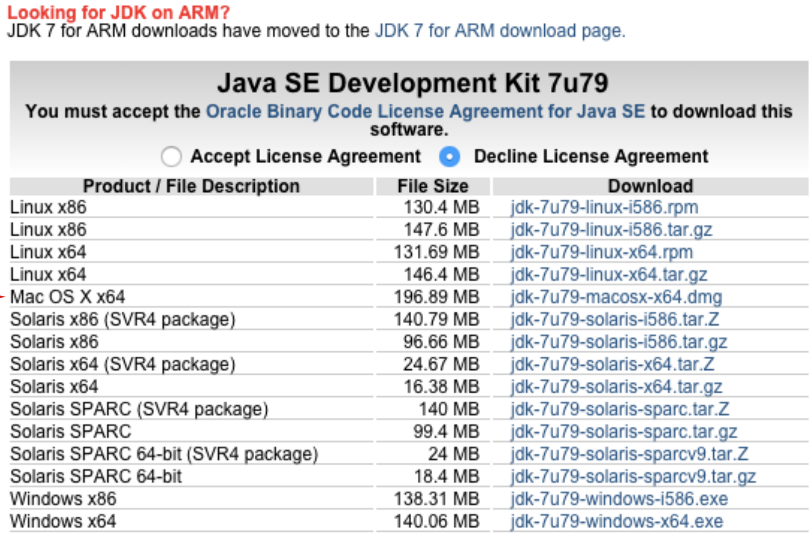
Necessary Softwares	Version used in this guide	Download link
OS	MAC	NA
JDK	jdk-7u79-macosx-x64	http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html
Selenium standalone jar	selenium-server-standalone-2.51.0	http://www.seleniumhq.org/download/
Chrome driver	chromedriver_mac32.zip	http://chromedriver.storage.googleapis.com/index.html?path=2.21/
Eclipse	eclipse-jee-mars-1-macosx-cocoa-x86_64.tar	https://www.eclipse.org/downloads/?osType=macosx

NB: Few links given in the table are applicable for MAC alone, you will have to download the appropriate version of software based on your operating system.

Downloading Instructions

1. JDK

Follow the link given in the table above to download the latest version of JDK.



Looking for JDK on ARM?
JDK 7 for ARM downloads have moved to the JDK 7 for ARM download page.

Java SE Development Kit 7u79

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

○ Accept License Agreement ● Decline License Agreement

Product / File Description	File Size	Download
Linux x86	130.4 MB	jdk-7u79-linux-i586.rpm
Linux x86	147.6 MB	jdk-7u79-linux-i586.tar.gz
Linux x64	131.69 MB	jdk-7u79-linux-x64.rpm
Linux x64	146.4 MB	jdk-7u79-linux-x64.tar.gz
Mac OS X x64	196.89 MB	jdk-7u79-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.79 MB	jdk-7u79-solaris-i586.tar.Z
Solaris x86	96.66 MB	jdk-7u79-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.67 MB	jdk-7u79-solaris-x64.tar.Z
Solaris x64	16.38 MB	jdk-7u79-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	140 MB	jdk-7u79-solaris-sparc.tar.Z
Solaris SPARC	99.4 MB	jdk-7u79-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	24 MB	jdk-7u79-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.4 MB	jdk-7u79-solaris-sparcv9.tar.gz
Windows x86	138.31 MB	jdk-7u79-windows-i586.exe
Windows x64	140.06 MB	jdk-7u79-windows-x64.exe

Choose appropriate downloadable file based on your operating system.

Steps

1. Download and save the file.

2. Double click to run the installable and follow the instructions to complete the installations.

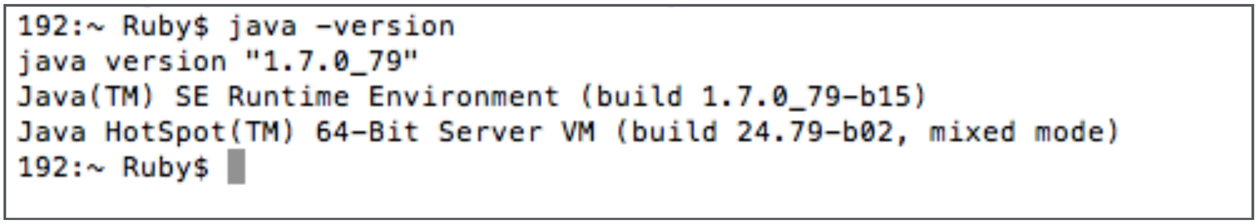
3. Installation is easy and straight forward but in case you need any kind of assistance, follow the guidelines mentioned in the link below;

http://docs.oracle.com/javase/7/docs/webnotes/install/mac/mac-jdk.html

4. To ensure proper installation of java, open - command prompt (Windows) OR terminal (MAC) and type following command;

```
java -version
```

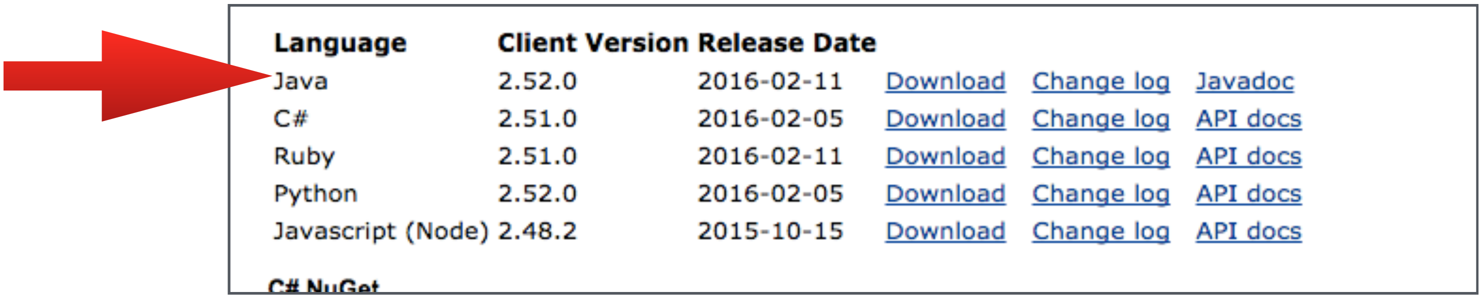
You will get the version number and other details as shown below,



```
192:~ Ruby$ java -version
java version "1.7.0_79"
Java(TM) SE Runtime Environment (build 1.7.0_79-b15)
Java HotSpot(TM) 64-Bit Server VM (build 24.79-b02, mixed mode)
192:~ Ruby$
```

2. SELENIUM JAR

Follow the link given in the table to download the latest version of Selenium jar.



Language	Client Version	Release Date			
Java	2.52.0	2016-02-11	Download	Change log	Javadoc
C#	2.51.0	2016-02-05	Download	Change log	API docs
Ruby	2.51.0	2016-02-11	Download	Change log	API docs
Python	2.52.0	2016-02-05	Download	Change log	API docs
Javascript (Node)	2.48.2	2015-10-15	Download	Change log	API docs

C# NuGet

Steps

1. Download selenium jar for java (as shown above).

2. Leave it as such, we will need it later.

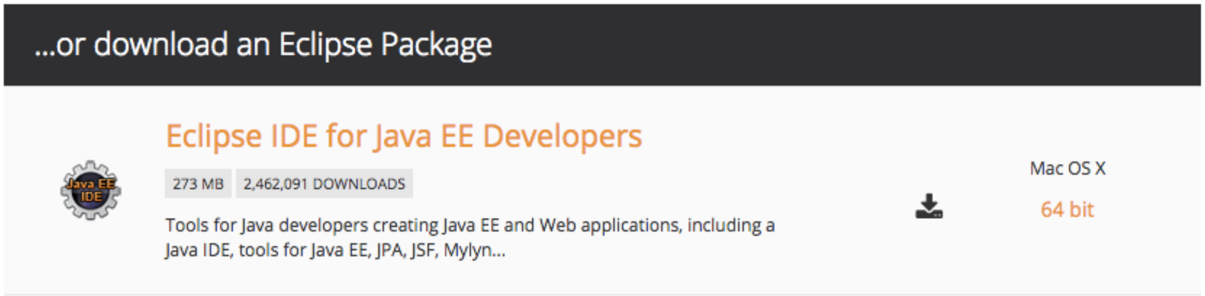
3. CHROME DRIVER

1. If you are not on mac machine, do a google search to download chrome driver that supports your operating system.

2. Unzip the file, you will get an executable file of chrome drive, leave it as such, we will need that in future.

4. ECLIPSE

Follow the link given in the table to download the latest version of eclipse.



...or download an Eclipse Package

Eclipse IDE for Java EE Developers

273 MB 2,462,091 DOWNLOADS

Tools for Java developers creating Java EE and Web applications, including a Java IDE, tools for Java EE, JPA, JSF, Mylyn...

Mac OS X

64 bit

1. If you are not on a mac machine, do a google search to download eclipse that supports your operating system.

2. Download and unzip the folder, you will get an icon (like below) saved to your folder.

Now that we have all the necessary softwares installed, we will move towards the second part of this guide.

II. Environment setup

Refer Video -> workspace_setup_1

NB: Your eclipse IDE will open to a welcome page and you will not have SeleniumDemo1 project displayed on project explorer.

III. Introduction to Basic JAVA

Learning Selenium doesn't require an in depth understanding of Java or OOPS concept. It does require some understanding of the language and syntax and other easy to learn stuffs though.

If you have any worries about not being a programmer, this is the last point to which you can carry those "mean" feelings.

We will learn a little bit about java now and get familiarized with the terms that we will often use while building our application.

1. Class, Object and Instance

Imagine two classroom's - A & B, you are in class A and both classes have same teacher. One day your teacher asks you to refer the notes from class B as she doesn't want to repeat the whole notes one more time.

How will you interact with class B and get the notes? you can't talk to the walls or board or windows to get the notes, you will have to talk to one person from class B and get the notes, right?

Classroom in this example is similar to a class in JAVA. It will have it's own variables and methods.

Student in this example is an object in JAVA. As each student represents it's class, each object also represents it's own class.

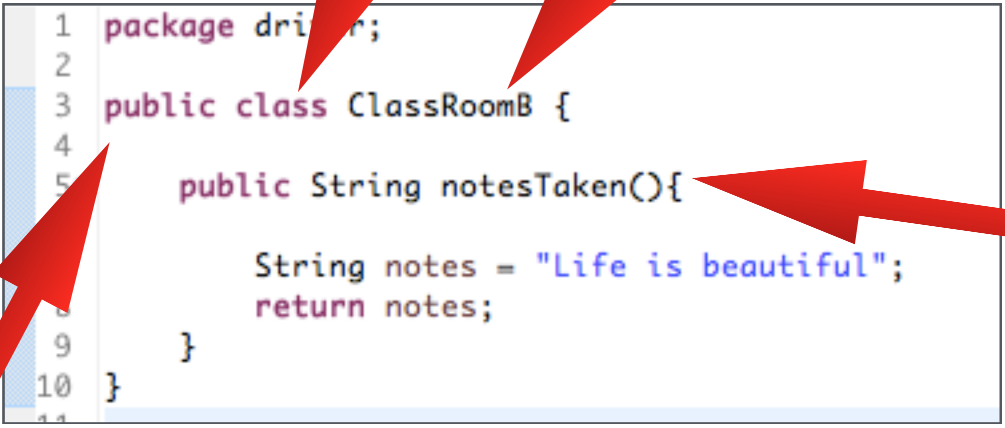
Interaction between two JAVA classes is done through **OBJECTS**. As each student in a class has a unique name, our objects also have their unique names.

new is the keyword used to create an object.

Lets look at the code for the above mentioned scenario,

Always mention keyword "**class**"

Java class corresponding to classroomB. Java **class name** should be always written in Mixed-case. First Letter capital.
Java class starts and ends with "{" and "}" .



```
package driver;

public class ClassRoomB {

    public String notesTaken(){

        String notes = "Life is beautiful";
        return notes;
    }
}
```

This is **access modifier**.

You may read more about access modifiers here but it's better not to worry about it now as you will be using "public" access modifiers for all selenium test cases.

notesTaken() is a method. A method opens and closes with "{" & "}".

Keep methods access modifier **public** as well.

"String" represents the return data type. If this method was to return an integer we would have written the method as

public **Integer**

Access Modifier:

Consider java class as a library in the real world and access modifiers as the type of library.

1. public library could be accessed by anyone so are public java classes.
2. private,protected library is limited to private members, similarly private,protected and default access modifiers imply restrictions on variables and inner classes.
3. **As a selenium tester, you need not worry about access modifiers. You will be writing public classes and public methods.**

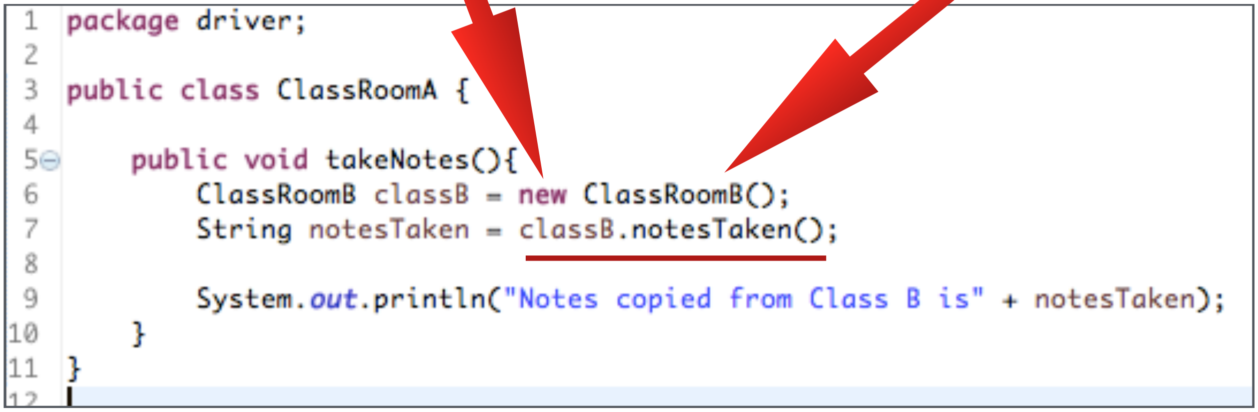
Let's look at the code of your class A. As you remember, you borrowed note from class B and prepared your own note.

★ By now, you know the syntax of a java class by heart, don't you? Write it with me,
It's *public class XxxYyy{.....}*

★ You also know the syntax of a method by heart, right? Again, write it with me,
public <return data type> xxxYyy(){......}

Using keyword "new" to create an object.

We have to call the method that's written in ClassRoomB. To access ClassRoomB, we will have to create an object of that class. Syntax to create an object is:
ClassName objectName = **new** ClassName();
classB is name of the object created here.



```
package driver;

public class ClassRoomA {

    public void takeNotes(){
        ClassRoomB classB = new ClassRoomB();
        String notesTaken = classB.notesTaken();

        System.out.println("Notes copied from Class B is" + notesTaken);
    }
}
```

classB.notesTaken() <————-> object.methodName()
A method written in one class (eg: notesTaken() is written in the class ClassRoomB) is accessed (made available) in another class (eg: class ClassRoomA) by using the syntax **object.methodName().**

Forget rest of the code as we will see all that while building our first application.

2. Let’s write our first Java program (Not Selenium)

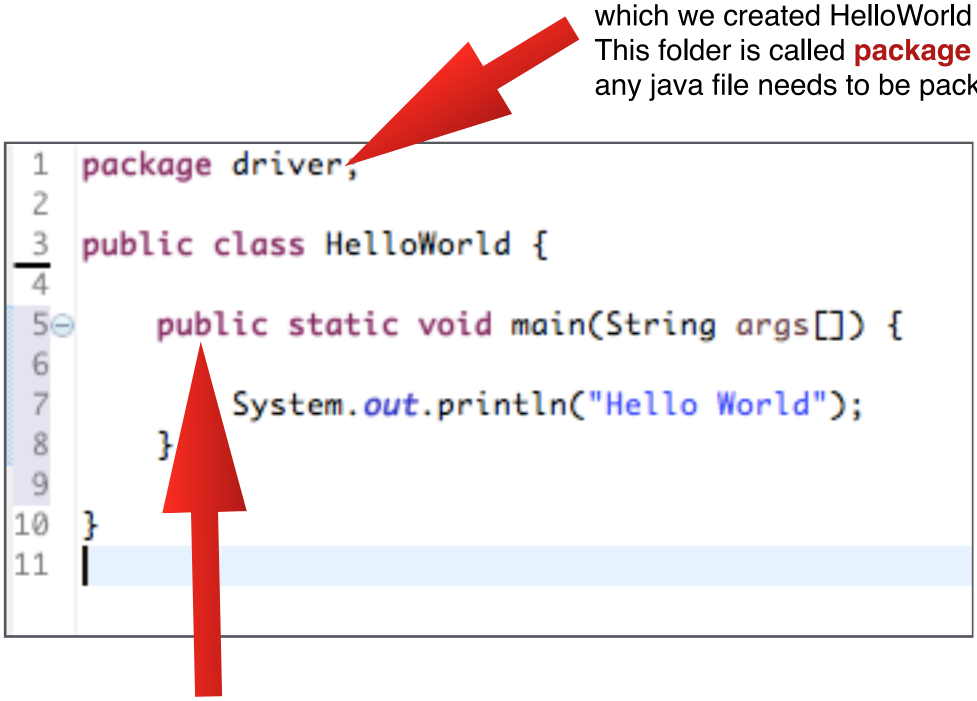
1. Open the file HelloWorld.java (As shown in the 1st workspace setup video)

2. Paste below written code in the HelloWorld class. (From here on, class refers to java class and not the classroom)

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

Now your HelloWorld class would look like the below image,

As you would have noticed, driver is the folder under which we created HelloWorld file.
This folder is called **package** in java. first statement in any java file needs to be package declaration.

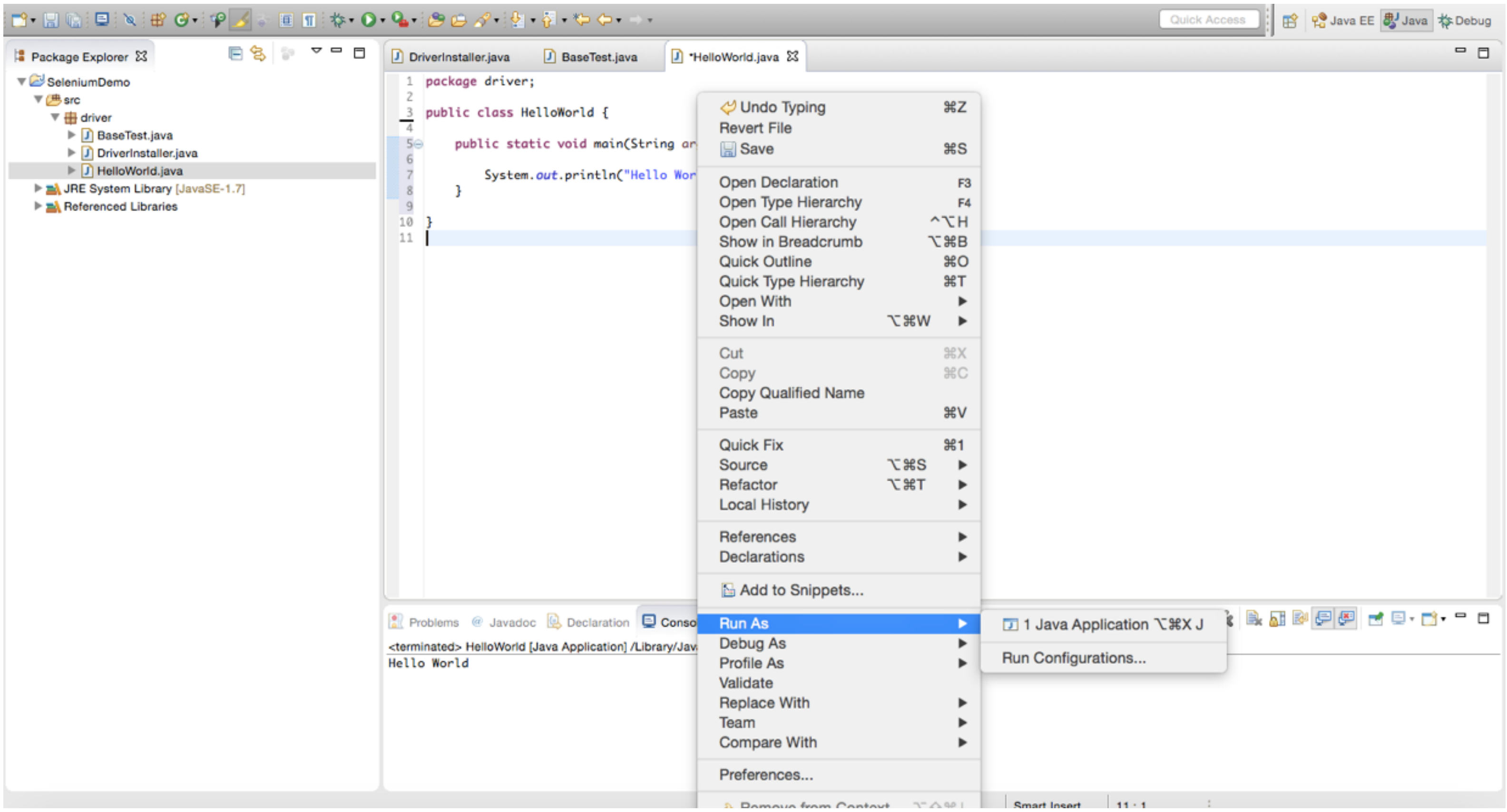


```
package driver;

public class HelloWorld {

	public static void main(String args[]) {

		System.out.println("Hello World");
	}

}

```

public static void main(String args[]) - This is the signature of main method.

- main method is the entry point of any java application. All other methods would be subsequently invoked from the main method.
- Every java application must have a main method.
- public static could be written as static public as well and args could be just any name you like, but the above signature is the conventional one.
- System.out.println is the command used to print output to the console.

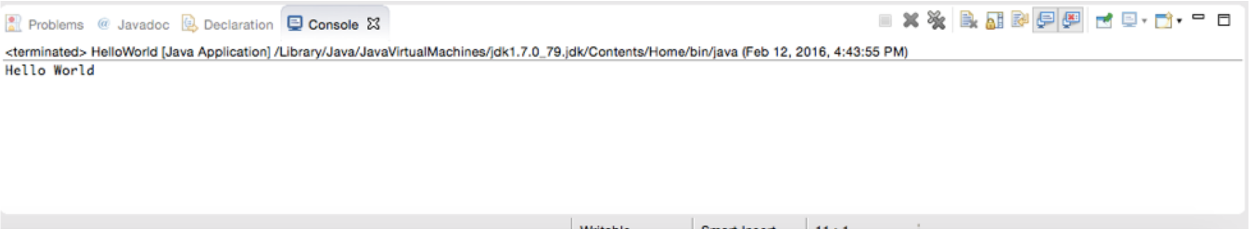
3. Right click on the HelloWorld java file and run the program as java application



```
package driver;

public class HelloWorld {

    public static void main(String ar

        System.out.println("Hello Wor
    }

}
```

```
<terminated> HelloWorld [Java Application] /Library/Jav
Hello World
```

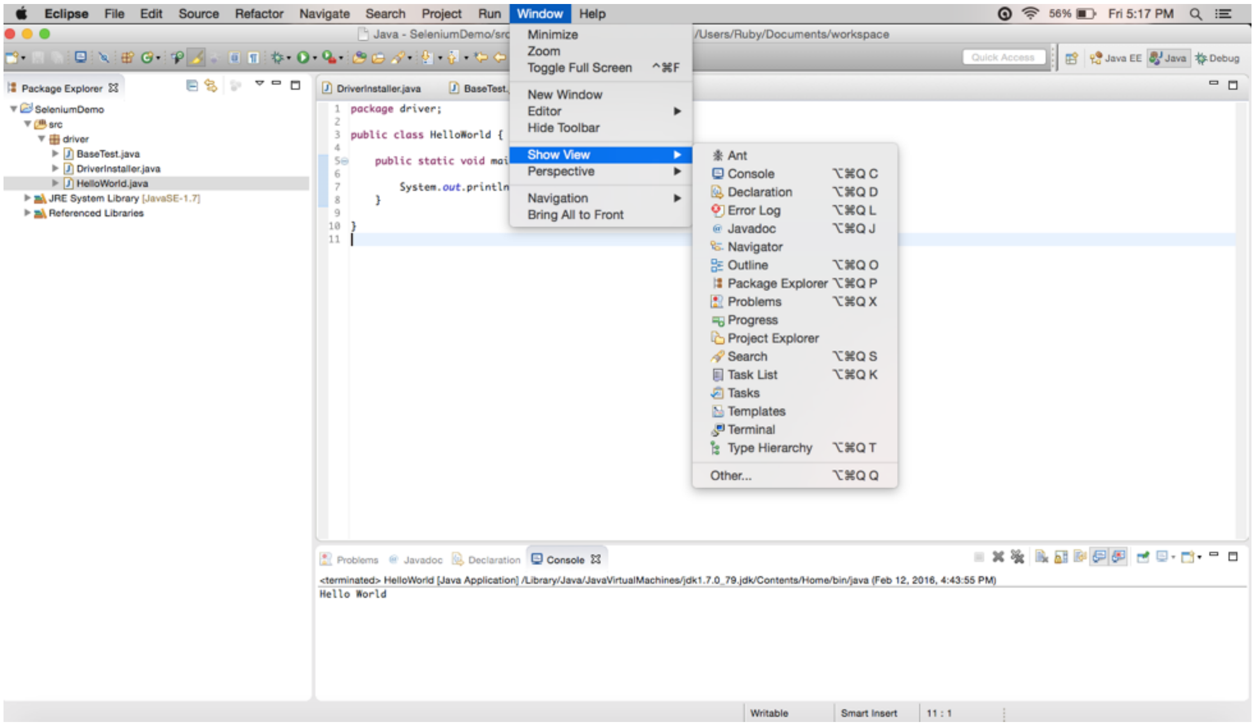
4. Output will be printed on console



```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java (Feb 12, 2016, 4:43:55 PM)
Hello World
```

That’s our first java program, Congratulations for coming this far!!!!!

NB: If you are not able to see the console view in your eclipse IDE, then select option window from menu (at the top) -> show view -> Console. (Image below)



```
package driver;

public class HelloWorld {

    public static void mai

        System.out.println
    }

}
```

```
<terminated> HelloWorld [Java Application] /Library/Java/JavaVirtualMachines/jdk1.7.0_79.jdk/Contents/Home/bin/java (Feb 12, 2016, 4:43:55 PM)
Hello World
```

You may try changing the access modifiers,main method signature and other stuffs to get an idea on what works and what doesn't.

You may also go back to the beginning of this chapter, pick up a good book (like head first java) and make sure you understand the basic concepts of,

- ★ Class
- ★ Object
- ★ Package
- ★ Signature and importance of main method
- ★ How to create an object
- ★ Access modifiers
- ★ Signature of **method** and **class**

Even if you can't code anything right now, it's fine, learn to read and understand the java HelloWorld class. Identify the method, classname, syntax etc.

IV. Introduction to Selenium

Selenium is best described by it's founders as a suite of tools to automate web browsers.

Selenium as you know is a testing framework for web applications. It supports most of the modern web browsers and could be run on windows, linux and mac machines.

Selenium tests could be written in any of the below mentioned languages

1. Groovy
2. Perl
3. PHP
4. Python
5. Ruby
6. Java
7. C#

We will be writing the test cases in JAVA programming language.

Why Selenium?

☑ It's open source.

☑ Supports large number of languages.

☑ Supports multiple browsers and operating systems.

☑ Integrates with other testing frameworks like TestNG and Junit.

☑ Integrates with Jenkins and Hudson for continuous integration.

☑ Integrates with ANT and Maven.

REMEMBER THESE....

- Entry point for any java application is main() method so we will write all test cases within main() method.
- Each test case will be written as a separate method, 10 test cases will become 10 methods.
- We will be writing public classes and public methods.
- There are n number of ways to validate the output of a test but we are using the basic and simplest form of all, print the result in console using command "*System.out.println*".

  (Best approach is to integrate with TestNG or Junit but that is out of the scope of this guide)

Build first Selenium project

Step 1 : Play below mentioned video to set the local environment to build selenium project.

NB: Remember the local path to selenium jar.

Refer Video -> workspace_setup_2

Step 2: Create a new file (like we created HelloWorld.java) and name it BaseTest.java

Step 3: Write down the main method.

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

}
```

Want complete explanation on the signature of main() method? Check here

Step 4: Copy and paste this code inside the main method

```
        try {


        } catch (final Exception excptn) {
            System.out.println("Exception caught at main : "+excptn);
        }
```

Now, your class will look like this

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

        try {


        } catch (Exception excptn) {
            System.out.println("Exception caught at main : "+excptn);
        }
}
```

EXPLANATION

No matter how well written a code is, it might break at some point due to some unforeseen reasons. As not-so-nerd people, it's on us to know the reason of any such failure.

Thus, we write all suspicious codes within try block, i.e, within the { and } of try block. It follows with catch block because there should be a way for our application to handle the issue caught in the try block.

catch (Exception excptn) { } tells the application what needs to be done with the exception that occurred in the try block.

In the above example, we will print the exception to console so we can fix the issue and run our application again.

Exception is a java class and *excptn* is just a name(you can name it cat or cow as well but stick to reasonable names).

Step 5: Create two variables as shown below

```
static WebDriver driver;
static Wait<WebDriver> wait;
```

Your class will now look like this;

```
package driver;

public class BaseTest {

    static WebDriver driver;
    static Wait<WebDriver> wait;


    public static void main(String[] args){

    try {


    } catch (final Exception excptn) {
        System.out.println("Exception caught at main : "+excptn);
    }
}
```

EXPLANATION

1. static WebDriver driver

static - Understand that a static method (main method here) can only access (use) static variables. (Curious cats might want to learn more about static variables and static methods, do a google search :))

WebDriver - This is an interface that comes in org.openqa.selenium package. Need you worry about all these explanations? Of course not, it’s enough to understand that we always need to use WebDriver, hence you will need to know the syntax used here.

driver - name of variable

2. static Wait<WebDriver> wait;

We will learn more about this in a short while

Step 6: With in the try block, start writing the REAL SELENIUM code.

1. System.setProperty(“webdriver.chrome.driver", “/Users/Ruby/Documents/chromedriver");

When using come browser, we need to explicitly set the driver path. When application tries to access the chrome browser, it will look for the location you specify in your application.

Replace “/Users/Ruby/Documents/chromedriver” by your local path to chrome driver. (you would have downloaded chrome driver at the beginning of this guide).

This step is valid for testing on chrome browser and internet explorer. Firefox browser doesn’t need driver path to be explicitly set.

If you are running test cases on internet explorer then use below mentioned code

```
System.setProperty("webdriver.ie.driver", “path to the IE driver”);
```

2. driver = new ChromeDriver();

driver variable we created in Step 5 is been assigned an object of ChromeDriver class. Again, need you worry about the inside details? Not at all (for now).

If you are running your test cases in chrome browser, assign ChromeDriver object to driver variable.

If you want to test on firefox browser, use below mentioned code,

```
driver = new FirefoxDriver()
```

Similarly, if you want to test on internet explorer, use below mentioned code,

```
driver = new InternetExplorerDriver();
```

3. driver.get(“http://www.google.com/“);

get(“URL”) is the method used to access a website.

In the example above we are trying to access google.

4. wait = new WebDriverWait(driver, 30);

This is where we use the wait variable created in step 5. This command will instruct the driver to hold on the activities for 30 seconds.

We want the google home page to load completely before sending any more command to the selenium driver, thus avoiding any exceptions.

5. driver.findElement(By.name("q")).sendKeys("ruby shiv\n”);

This step would instruct selenium find element on google web page that is named “q”. You can also instruct selenium driver to find web elements using id, xpath and other tags.

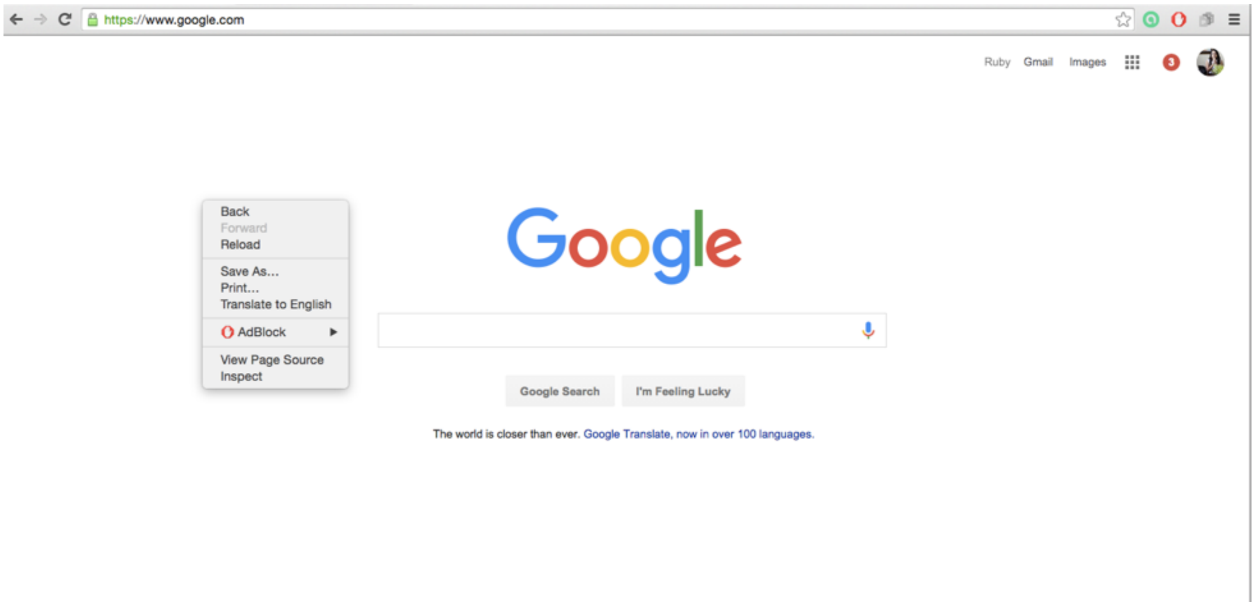
sendKeys(“”) command will send the text within the “ ” to the web elements.

eg: Here we are sending text “ruby shiv” to google’s search-box.

FAQ: How to find elements that is named q.

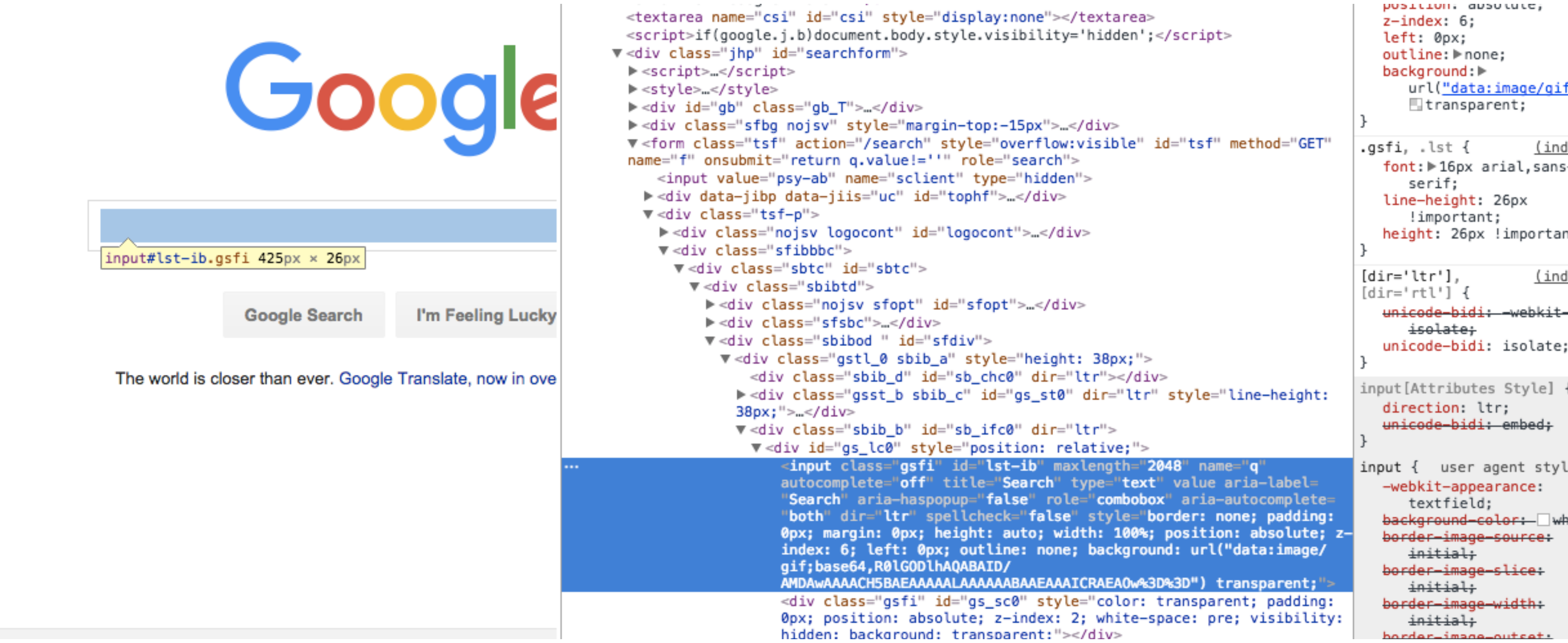
Answer: Follow these easy steps

1. Right click on the appropriate web element (textbox in this example) and click on inspect



The world is closer than ever. Google Translate, now in over 100 languages.

2. On the right side, you will find the id and name of the web elements.



The world is closer than ever. Google Translate, now in ove

```
<textarea name="csi" id="csi" style="display:none"></textarea>
<script>if(google.j.b)document.body.style.visibility='hidden';</script>
▼<div class="jhp" id="searchform">
  ▶<script>…</script>
  ▶<style>…</style>
  ▶<div id="gb" class="gb_T">…</div>
  ▶<div class="sfbg nojsv" style="margin-top:-15px">…</div>
  ▼<form class="tsf" action="/search" style="overflow:visible" id="tsf" method="GET"
  name="f" onsubmit="return q.value!=''" role="search">
      <input value="psy-ab" name="sclient" type="hidden">
    ▶<div data-jibp data-jiis="uc" id="tophf">…</div>
    ▼<div class="tsf-p">
      ▶<div class="nojsv logocont" id="logocont">…</div>
      ▼<div class="sfibbbc">
        ▼<div class="sbtc" id="sbtc">
          ▼<div class="sbibtd">
            ▶<div class="nojsv sfopt" id="sfopt">…</div>
            ▶<div class="sfsbc">…</div>
            ▼<div class="sbibod " id="sfdiv">
              ▼<div class="gstl_0 sbib_a" style="height: 38px;">
                  <div class="sbib_d" id="sb_chc0" dir="ltr"></div>
                ▶<div class="gsst_b sbib_c" id="gs_st0" dir="ltr" style="line-height:
                38px;">…</div>
                ▼<div class="sbib_b" id="sb_ifc0" dir="ltr">
                  ▼<div id="gs_lc0" style="position: relative;">
…                     <input class="gsfi" id="lst-ib" maxlength="2048" name="q"
                      autocomplete="off" title="Search" type="text" value aria-label=
                      "Search" aria-haspopup="false" role="combobox" aria-autocomplete=
                      "both" dir="ltr" spellcheck="false" style="border: none; padding:
                      0px; margin: 0px; height: auto; width: 100%; position: absolute; z-
                      index: 6; left: 0px; outline: none; background: url("data:image/
                      gif;base64,R0lGODlhAQABAID/
                      AMDAwAAAACH5BAEAAAAALAAAAAABAAEAAAICRAEAOw%3D%3D") transparent;">
                      <div class="gsfi" id="gs_sc0" style="color: transparent; padding:
                      0px; position: absolute; z-index: 2; white-space: pre; visibility:
                      hidden; background: transparent;"></div>
```

```
  position: absolute;
  z-index: 6;
  left: 0px;
  outline: ▶none;
  background: ▶
     url("data:image/gif
     ▨transparent;
}
.gsfi, .lst {          (ind
  font: ▶16px arial,sans-
     serif;
  line-height: 26px
     !important;
  height: 26px !importan
}
[dir='ltr'],           (ind
[dir='rtl'] {
  ~~unicode-bidi: -webkit-~~
     ~~isolate;~~
  unicode-bidi: isolate;
}
input[Attributes Style] {
  direction: ltr;
  ~~unicode-bidi: embed;~~
}
input {   user agent styl
  -webkit-appearance:
     textfield;
  ~~background-color: □wh~~
  ~~border-image-source:~~
     ~~initial;~~
  ~~border-image-slice:~~
     ~~initial;~~
  ~~border-image-width:~~
     ~~initial;~~
  ~~border-image-outset:~~
```

6. driver.findElement(By.name(“btnG")).click();

Inspect the search button google home page and check the corresponding button name (don’t get confused with btnK and btnG, use btnG or try using both to figure out the difference :))

7. try {
Thread.sleep(1000);
} catch (InterruptedException e) {
e.printStackTrace();
}

This step is optional and you may use *wait = new WebDriverWait(driver, 30);* instead.

8. String resultText = driver.findElement(By.tagName(“body”)).getText();

This command would fetch all the search results displayed under the tagName “body” and save the result in a string variable named resultText.

9. final boolean result = resultText.contains(“rubyshiv.com");

This command validates the presence of “rubyshiv.com” in the resultText string and the result is saved as boolean variable.

boolean value for any variable can hold either one of these values - true or false, i.e, variable result can only have values **true** or **false**.

10. driver.close();

We are done with the selenium driver for now, so lets close that instance.

11. System.out.println("Test result of checkWebsiteExists() is :"+(result ? “PASS”:”FAIL"));

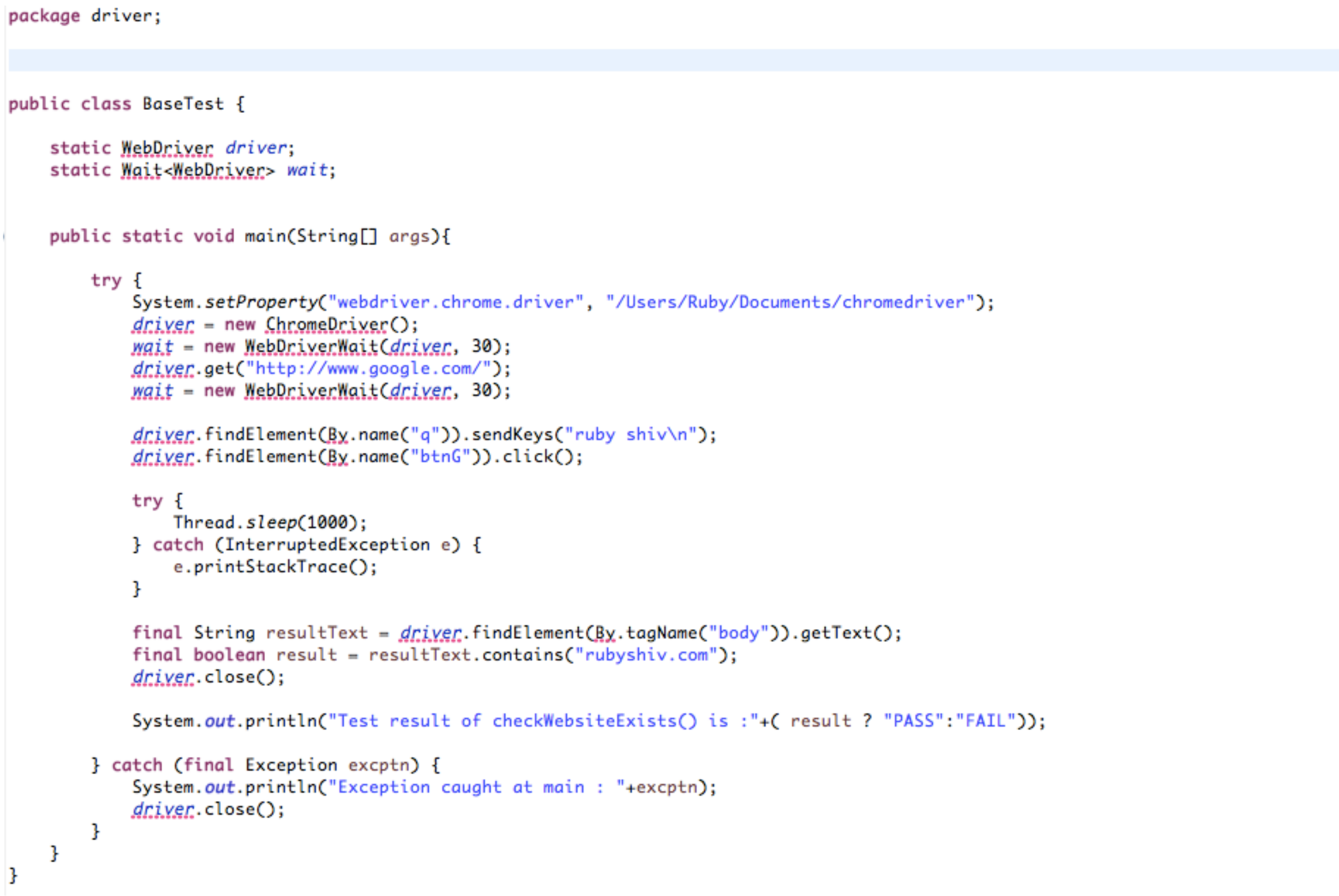
So,in this last step, we print the final output by printing PASS or FAIL on the console.

EXPLANATION: result? “PASS”: “FAIL”

This is a java expression that means, if true then string value is PASS else it is FAIL. As we saved the final result into a boolean variable “result”, it would either have value- true or false.

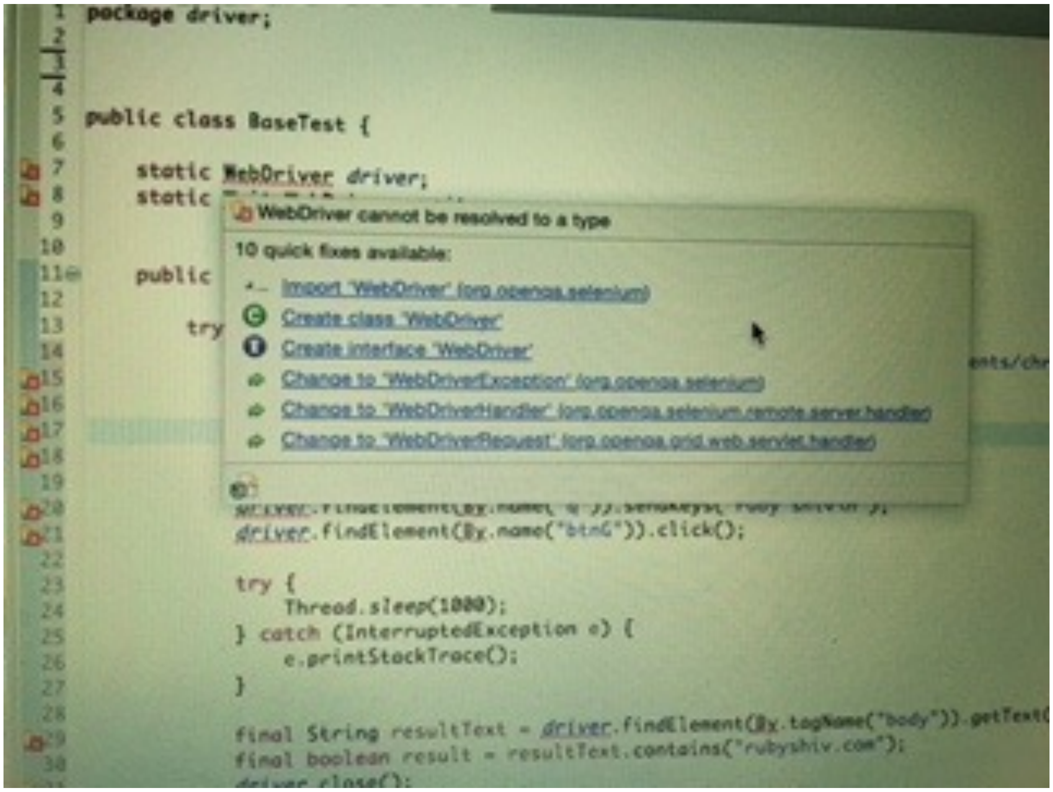
NB: import appropriate packages. Eclipse will show error in red, indicating a missing import. Move cursor over these red indicators and you will find the import commands been suggested by eclipse. (As shown in pics below)

At this point your class will look like the pic below,



```
package driver;


public class BaseTest {

    static WebDriver driver;
    static Wait<WebDriver> wait;


    public static void main(String[] args){

        try {
            System.setProperty("webdriver.chrome.driver", "/Users/Ruby/Documents/chromedriver");
            driver = new ChromeDriver();
            wait = new WebDriverWait(driver, 30);
            driver.get("http://www.google.com/");
            wait = new WebDriverWait(driver, 30);

            driver.findElement(By.name("q")).sendKeys("ruby shiv\n");
            driver.findElement(By.name("btnG")).click();

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            final String resultText = driver.findElement(By.tagName("body")).getText();
            final boolean result = resultText.contains("rubyshiv.com");
            driver.close();

            System.out.println("Test result of checkWebsiteExists() is :"+( result ? "PASS":"FAIL"));

        } catch (final Exception excptn) {
            System.out.println("Exception caught at main : "+excptn);
            driver.close();
        }
    }
}
```

Move the cursor over these red indicators and you will find these options, click on the import option and the error will be resolved.



```
package driver;


public class BaseTest {

    static WebDriver driver;
    static
```

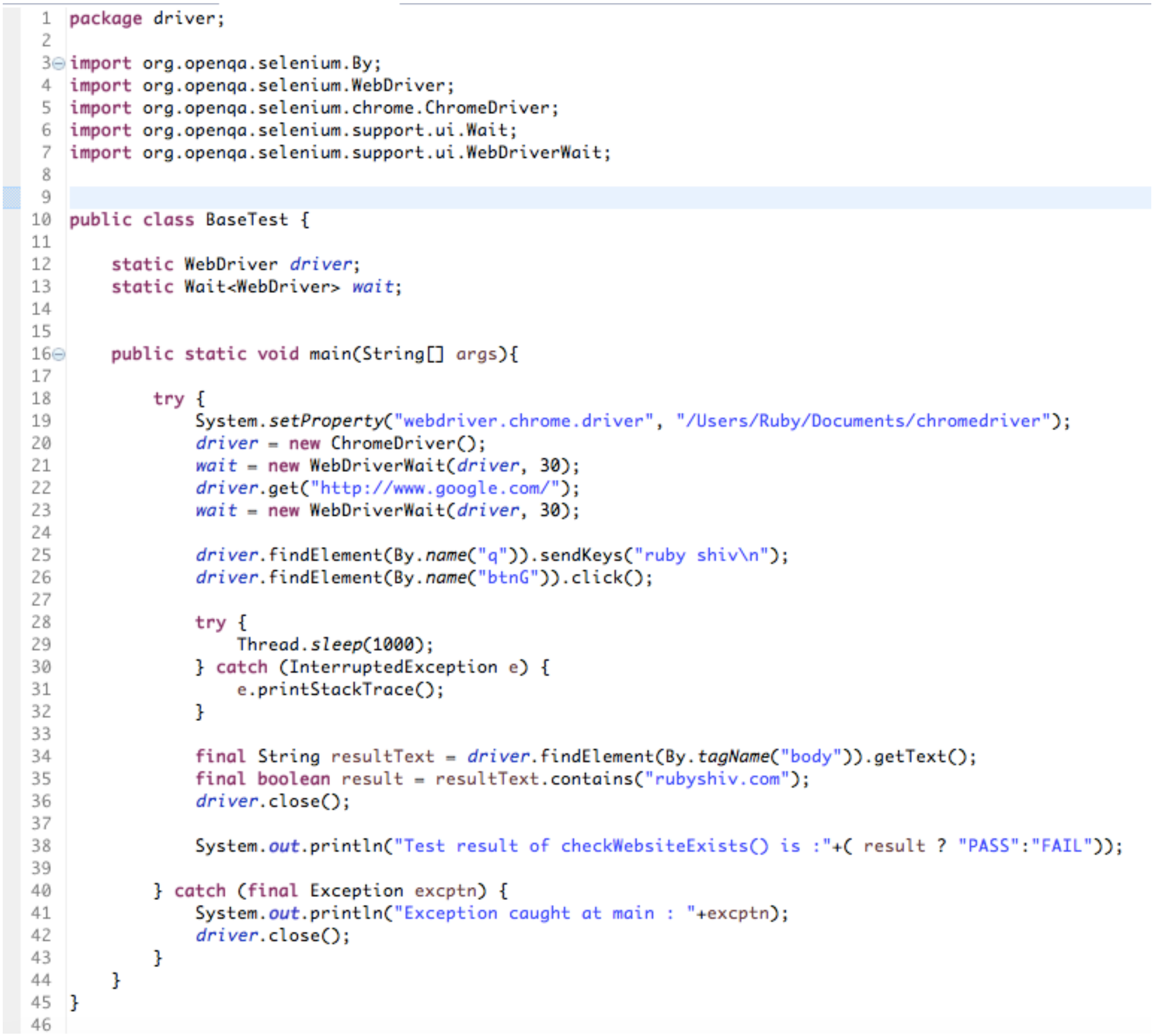
WebDriver cannot be resolved to a type

10 quick fixes available:

- Import 'WebDriver' (org.openqa.selenium)
- Create class 'WebDriver'
- Create interface 'WebDriver'
- Change to 'WebDriverException' (org.openqa.selenium)
- Change to 'WebDriverHandler' (org.openqa.selenium.remote.server.handler)
- Change to 'WebDriverRequest' (org.openqa.grid.web.servlet.handler)

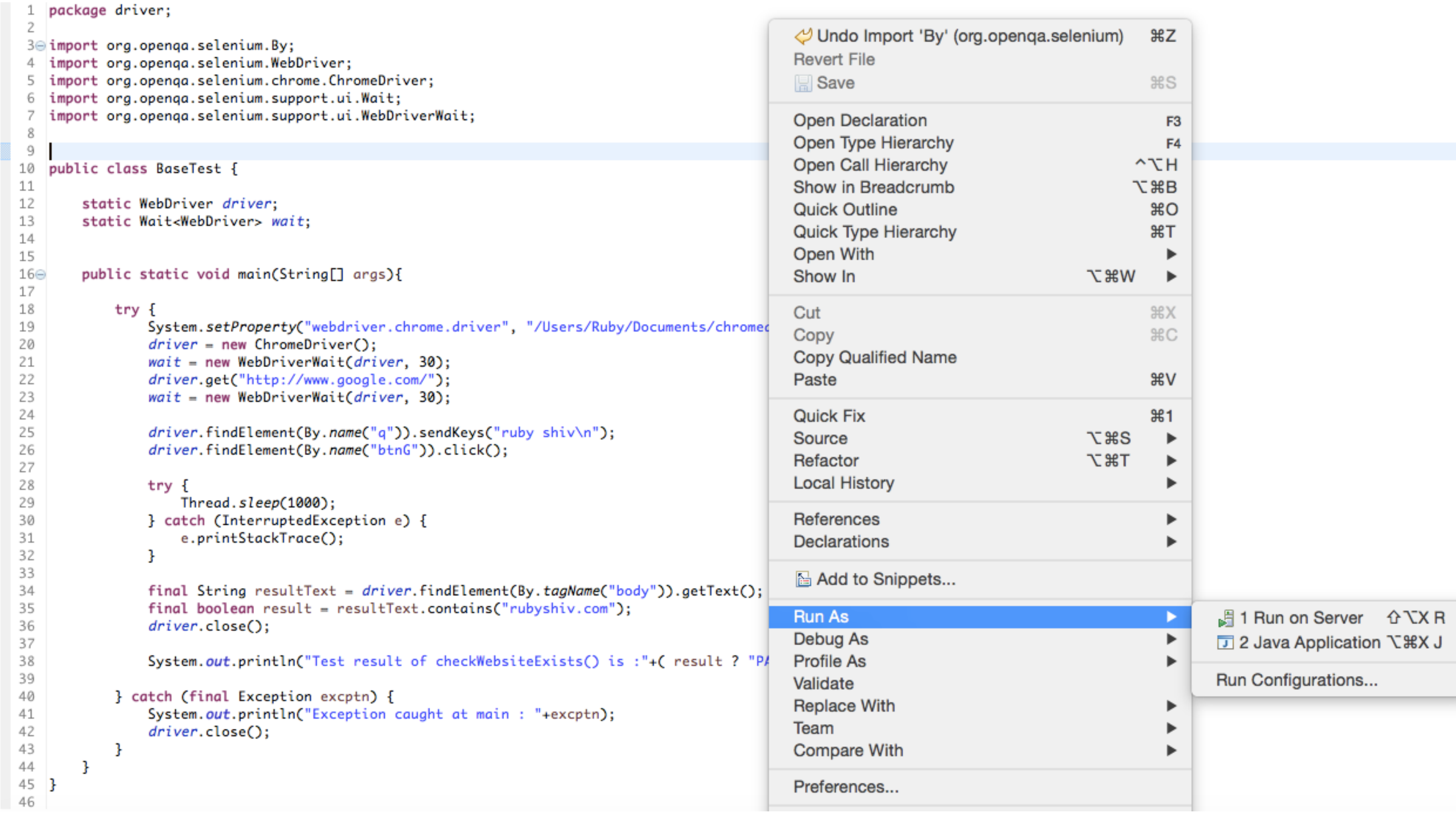
```
            driver.findElement(By.name("btnG")).click();

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            final String resultText = driver.findElement(By.tagName("body")).getText(
            final boolean result = resultText.contains("rubyshiv.com");
```

After all the error’s are resolved your application will look like this,



```
package driver;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;


public class BaseTest {

    static WebDriver driver;
    static Wait<WebDriver> wait;


    public static void main(String[] args){

        try {
            System.setProperty("webdriver.chrome.driver", "/Users/Ruby/Documents/chromedriver");
            driver = new ChromeDriver();
            wait = new WebDriverWait(driver, 30);
            driver.get("http://www.google.com/");
            wait = new WebDriverWait(driver, 30);

            driver.findElement(By.name("q")).sendKeys("ruby shiv\n");
            driver.findElement(By.name("btnG")).click();

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            final String resultText = driver.findElement(By.tagName("body")).getText();
            final boolean result = resultText.contains("rubyshiv.com");
            driver.close();

            System.out.println("Test result of checkWebsiteExists() is :"+( result ? "PASS":"FAIL"));

        } catch (final Exception excptn) {
            System.out.println("Exception caught at main : "+excptn);
            driver.close();
        }
    }
}

```

Right click on the file and run as java application.



```
package driver;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Wait;
import org.openqa.selenium.support.ui.WebDriverWait;


public class BaseTest {

    static WebDriver driver;
    static Wait<WebDriver> wait;


    public static void main(String[] args){

        try {
            System.setProperty("webdriver.chrome.driver", "/Users/Ruby/Documents/chromed
            driver = new ChromeDriver();
            wait = new WebDriverWait(driver, 30);
            driver.get("http://www.google.com/");
            wait = new WebDriverWait(driver, 30);

            driver.findElement(By.name("q")).sendKeys("ruby shiv\n");
            driver.findElement(By.name("btnG")).click();

            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            final String resultText = driver.findElement(By.tagName("body")).getText();
            final boolean result = resultText.contains("rubyshiv.com");
            driver.close();

            System.out.println("Test result of checkWebsiteExists() is :"+( result ? "PA

        } catch (final Exception excptn) {
            System.out.println("Exception caught at main : "+excptn);
            driver.close();
        }
    }
}

```

NB: You can add more seconds to wait instructions in case you feel the browser actions are too fast for the eyes.

Check that your browser is not running under any other open applications.

At this point, you have successfully built your first selenium application. You have learned the basic elements needed to build a selenium application.

Next step? There are many selenium applications and examples available for free over the internet. Pick any and practice more complex examples.