**Task 15: Automation Testing**

1. **Explain the difference between Selenium IDE, Selenium Web Driver and Selenium Grid.**

***Answer:***

**Selenium IDE (Integrated Development Environment):-**

Selenium IDE (Integrated Development Environment) is a Firefox Plugin. It is the Simplest Framework in the Selenium Suite. It allows us to record and playback the scripts. Even though we can create scripts using Selenium IDE, we need to use Selenium Web Driver to write more advanced and robust test cases.

Operating System Support – Windows, Mac OS, Linux

Browser Support- Mozilla Firefox

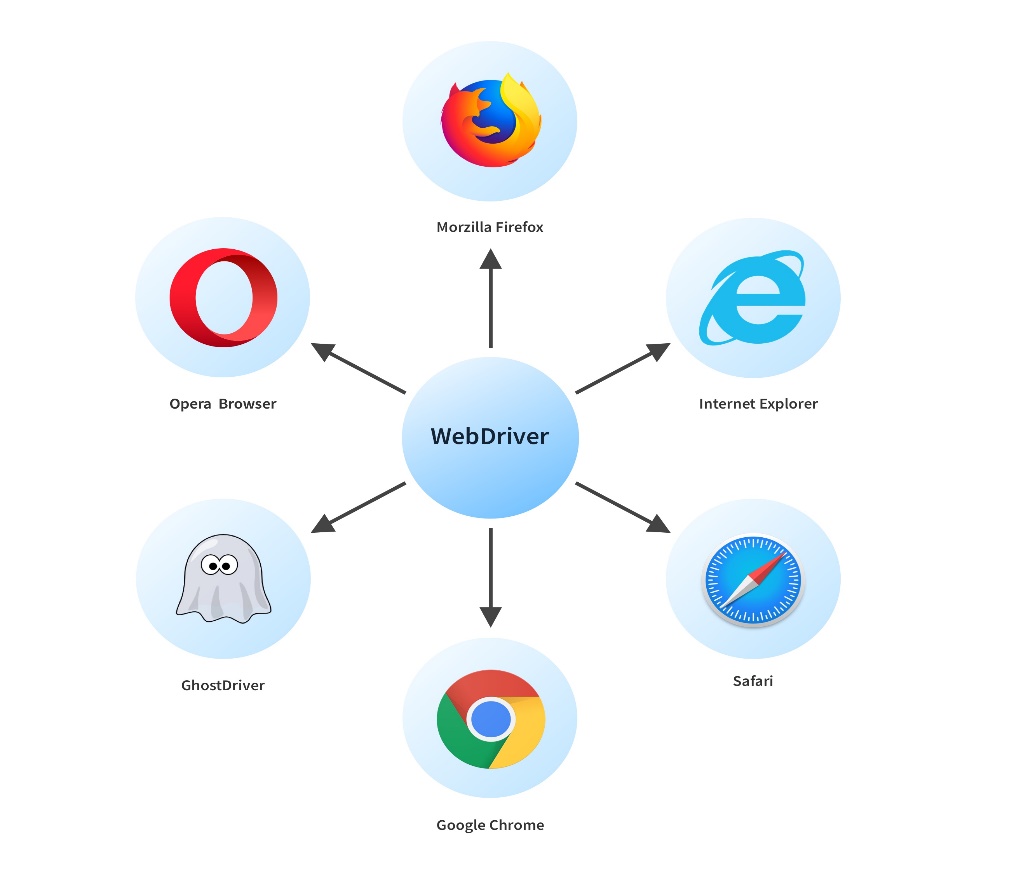
**Selenium Web Driver:-**

Selenium Web Driver AKA Selenium 2 is a browser automation framework that accepts commands and sends them to a browser. It is implemented through a browser specific driver. It controls the browser by directly communicating with it. Selenium Web Driver supports java, c#, PHP, python, Perl, Ruby.

JUnit and TestNG are the open source frameworks supported by the selenium web driver.

Operating System Support – Windows, Mac OS, Linux, Solaris

Browser Support – Mozilla Firefox, Internet Explorer, Google Chrome



**Selenium Grid:-**

Selenium Grid is a tool used together with Selenium RC to run tests on different machines against different browser in parallel. That is, running multiple tests at the same time against different machines running different browsers and operating systems.

A hub is a server which controls the test execution in multiple machines. Node is the machine which is attached to the hub and there can be multiple nodes in a single selenium Grid.

The typical differences are tabulated below:

|  |  |  |
| --- | --- | --- |
| **Selenium IDE** | **Selenium WebDriver** | **Selenium Grid** |
| Requires no programming to use | Requires programming skills | It allows you to run yours tests on different operating Systems. |
| Simple installation and quick start capabilities in under 30 minutes | Harder to learn and will take roughly a week to become an effective user | Selenium Grid reducing time for testing. |
| Possible to run CI | Well integrated with CI | Selenium Grid can set up a hub and multiple nodes. |
| Exports Into a programming language | Looping constructs, recursion, machine learning and AI possibilities | Hub coordinates the test distribution. |
| Requires Some CSS/XPath knowledge | Requires web programming knowledge | The nodes execute the tests on different. |
| Visual interface | Writing computer code | This is useful for cross browser and cross platform testing |
| Can’t perform Setup(beware of data change) | May be easier to version control | Web application works on consistently across various environment |
| In-browser FF and Chrome | IN-CI anything write-once and runs in any browser | It run different operating System. |

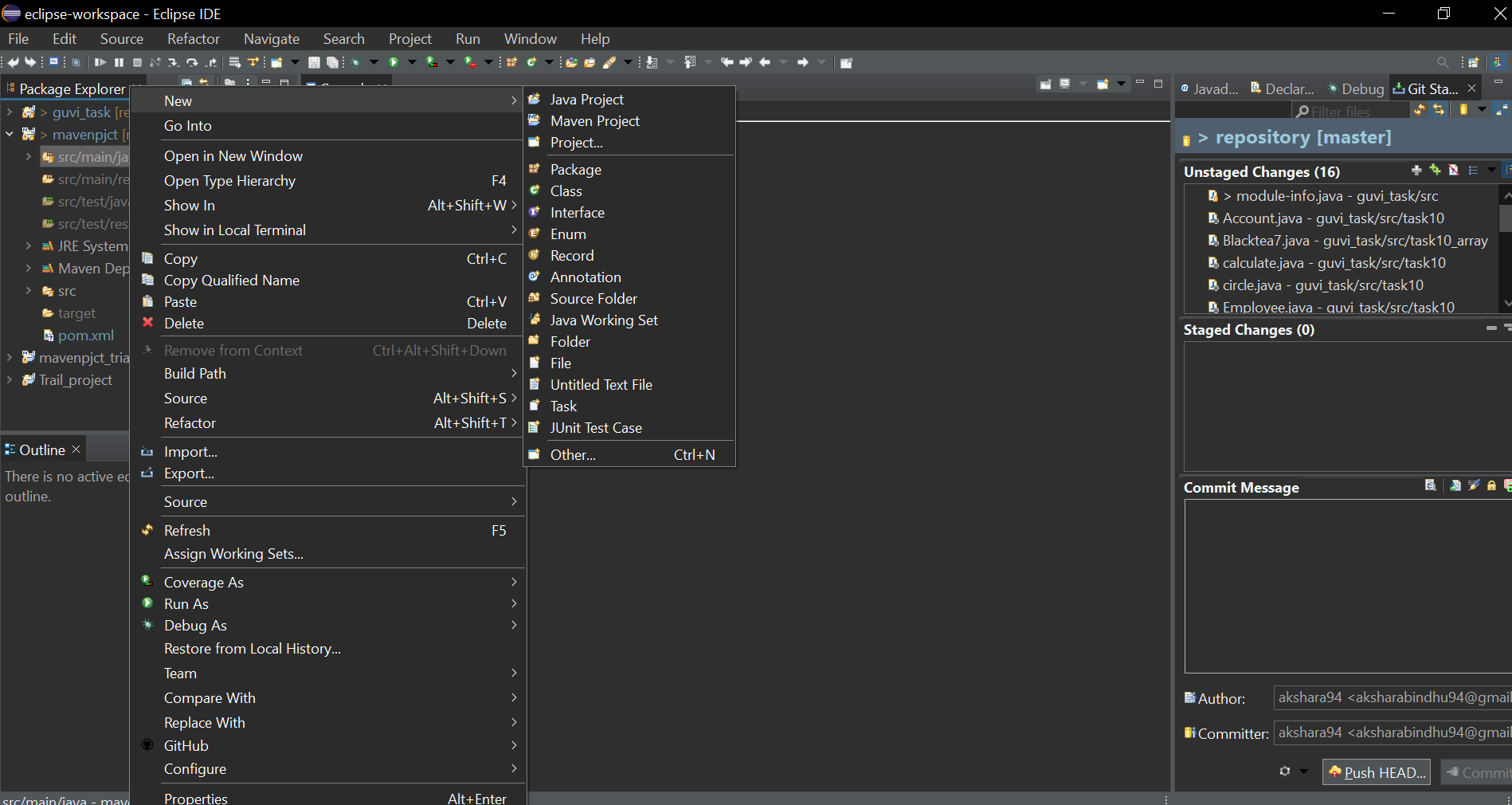
1. **Write a Selenium script in java to open Google and Search for “Selenium Browser Driver.”**

***Answer:***

Chrome browser implements the WebDriver protocol using an executable called **ChromeDriver.exe**. This executable start a server on your system which in turn is responsible for running the test scripts in Selenium.

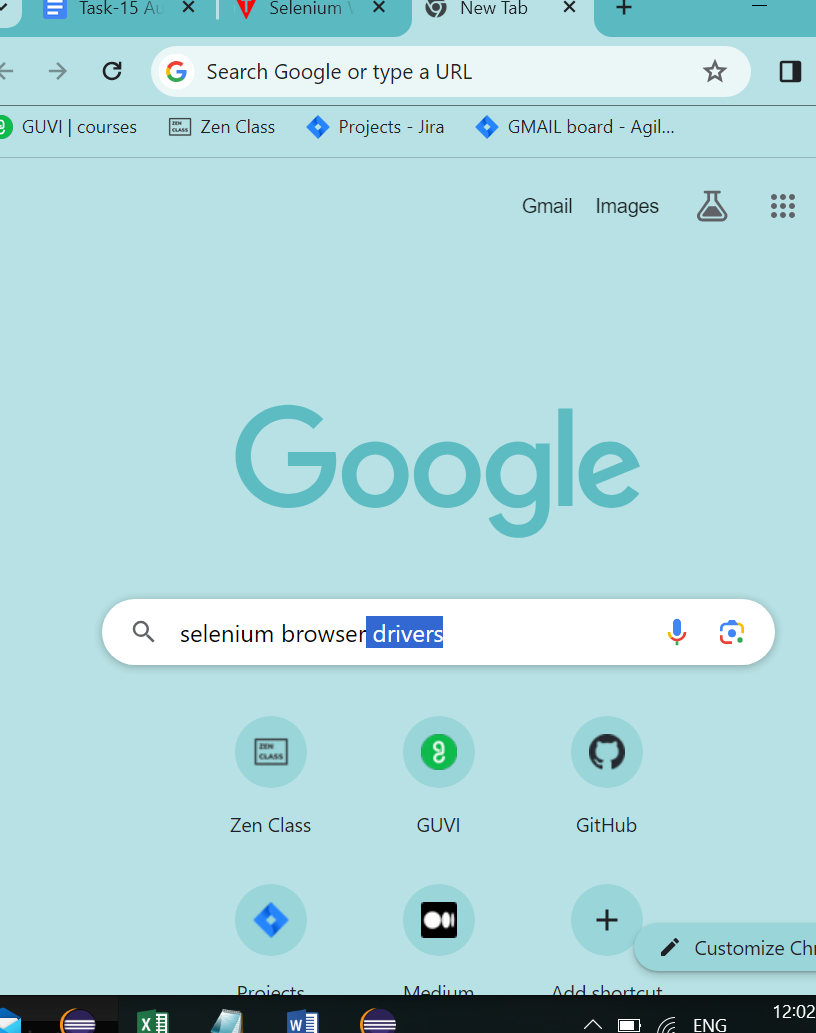
**Step 1:** Right click on the "src" folder and create a new Class File from New > Class.

Give Class name and click on "Finish" button.

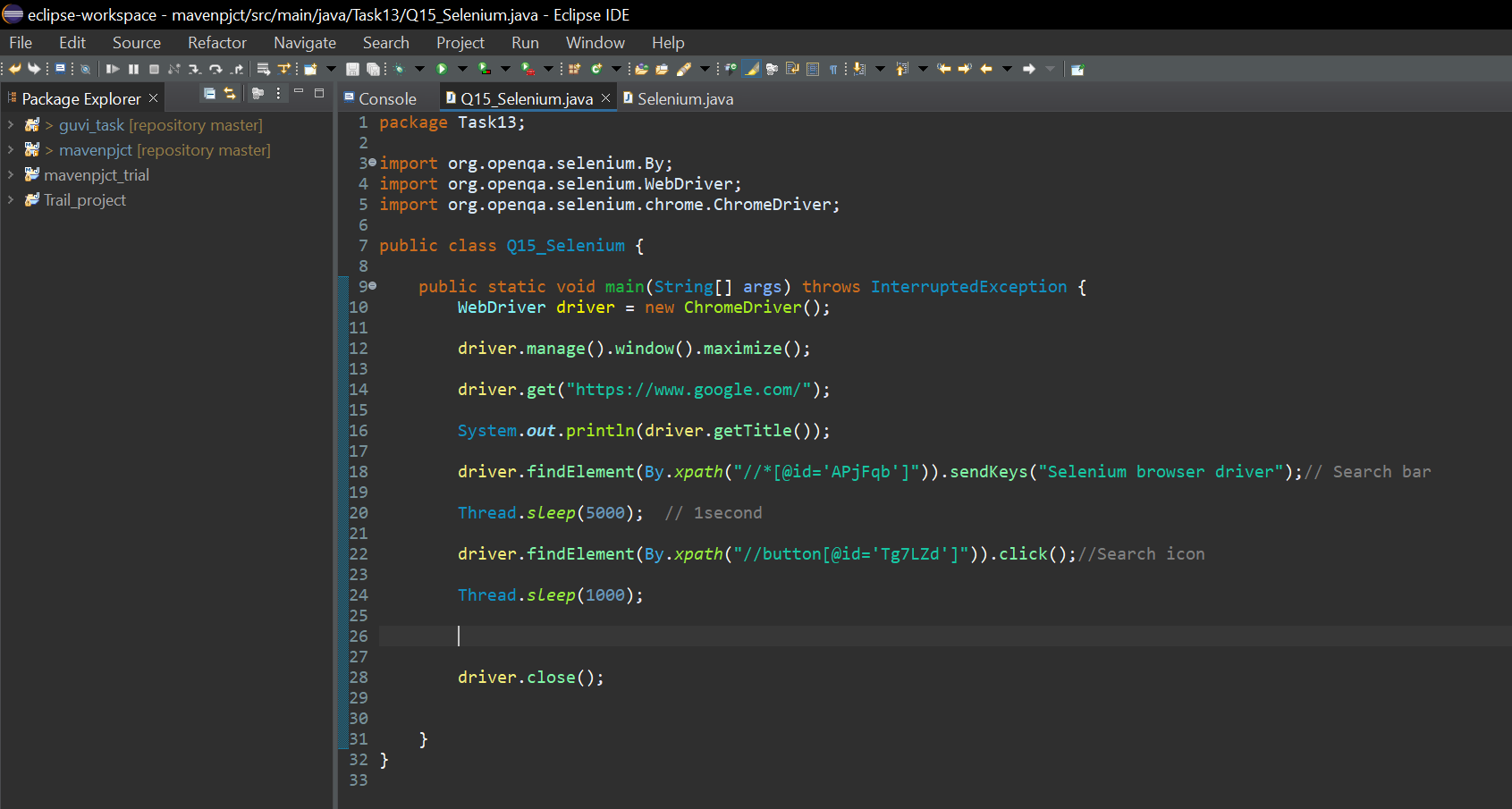
****

**Step2.** Open URL: <https://sites.google.com/a/chromium.org/chromedriver/downloads> in your browser.

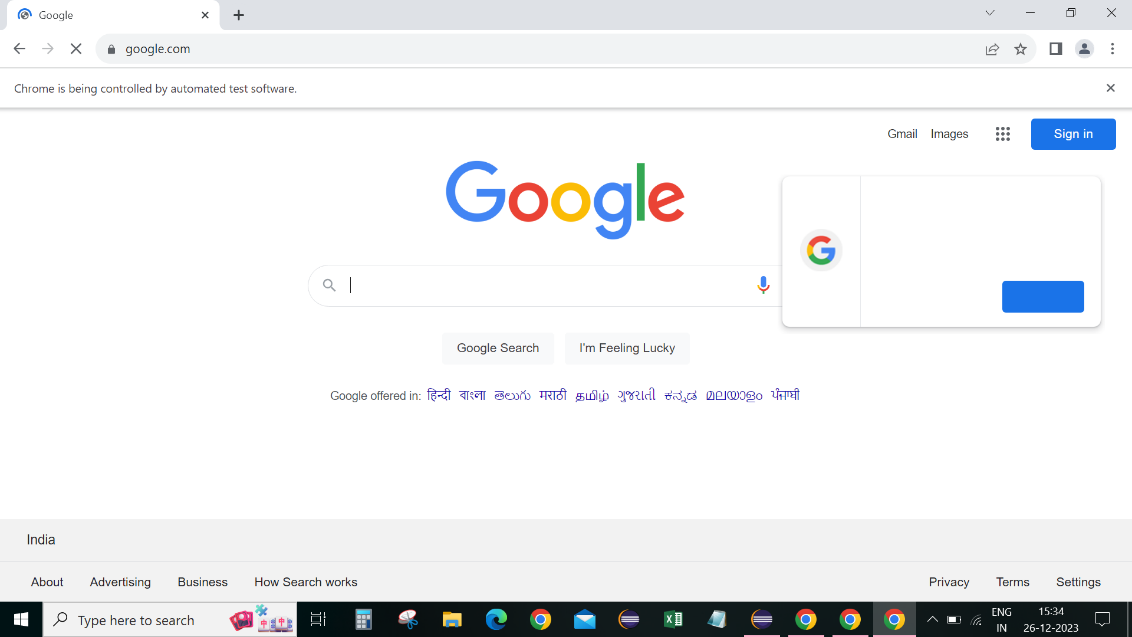
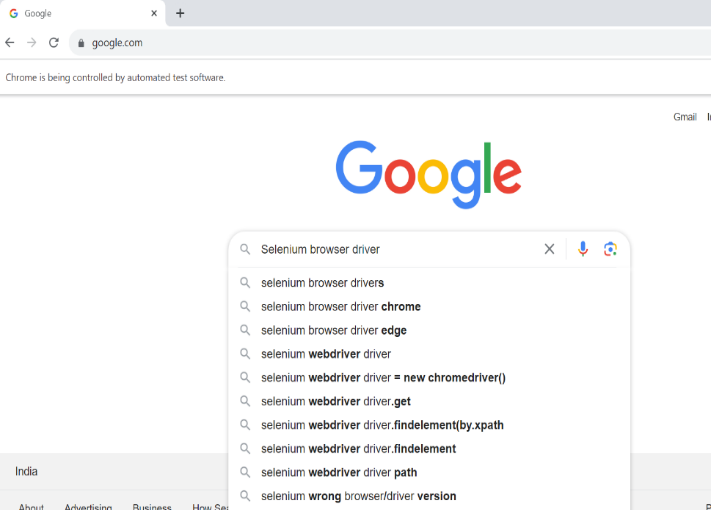
**Step3**. Click on the "ChromeDriver 2.41" link. It will redirect to the directory of ChromeDriver executables files. Download the driver based on the operating system working with.



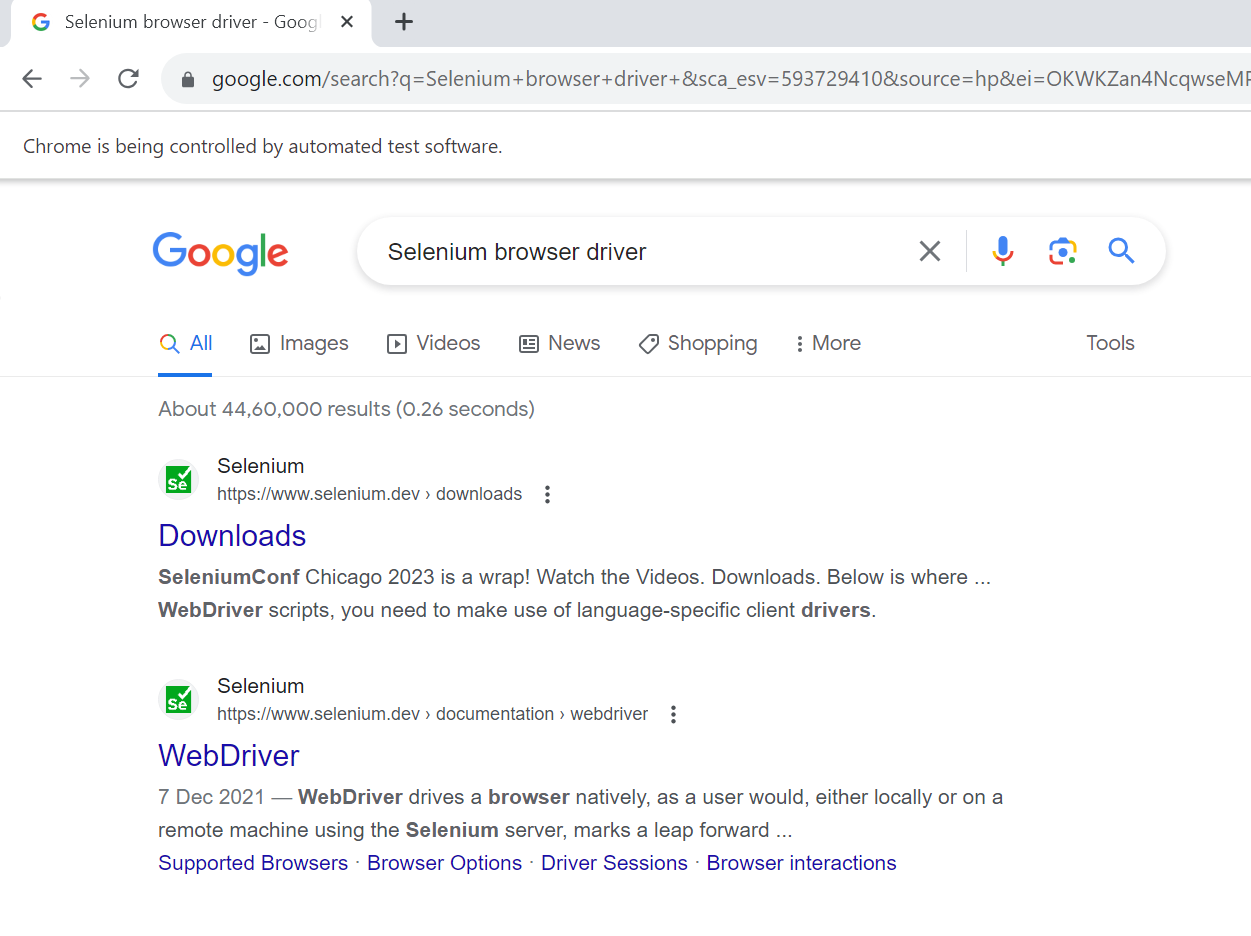
**Step 4:** Type the program in Eclispe and run the program



**Step 5:** Output of opening google and typing selenium browser

**Step 6:** Search results



1. **What is Selenium? How it is useful in Automation Testing?**

***Answer:***

Selenium is an open-source, automated testing tool used to test web applications across various browsers. Selenium can only test web applications, unfortunately, so desktop and mobile apps can’t be tested. However, other tools like Appium and HP’s QTP can be used to test software and mobile applications.

* Selenium is easy to use since it’s primarily developed in JavaScript
* Selenium can test web applications against various browsers like Firefox, Chrome, Opera and safari.
* Tests can be coded in several programming languages like Java, Python, Perl, PHP, and Ruby
* Selenium is platform-independent, meaning it can deploy on Windows and Linux.
* Selenium can be integrated with tools like J unit and Testing for test Management.

**Useful to Automation Testing:-**

Automation testing aims to improve testing efficiency, accuracy and speed by automating repetitive and time-consuming manual testing tasks. Automation Testing or test automation uses open sources or paid automated testing tools to test software applications quickly and efficiently, and without human intervention.

* **Increase in test coverage:** More scripts can be tested at the same time, increasing test coverage. It helps save time and takes the load off of manual testers.
* **Greater accuracy:** Continuous testing increases the chances of errors while done manually, but in automated testing, repetitive tests can be performed with the same precision.
* **Cost saving:** With automated testing, the test cases run at a faster speed and bugs are identified early on in the development cycle and fixed. Despite high initial coasts, once the automated framework is set up, there is an overall reduction in the costs.
* **Reduces regression testing time:** Automated regression practised with testing tools allows manual testers to be freed from running monotonous regression tests. The testers get time to perform value-add tasks.
* **Performs tasks that cannot be done by manual testers:** Certain tests Such as the controlled web application tests can be simulated with automated testing, but cannot be performed by manual testers.
* **Time Saving:** With automated testing, manual scripts are also automated and regression testing time is reduced. As tests are run 24 hours, automated testing saves time. Automated testing saves significant time by executing tests faster compared to manual testing.
* **Reusability of test scripts:** In automated testing, the same script can be used with minor changes. Thus, the reusability of test scripts facilitates the process of testing, and also, the same scripts can be stored and reused for repeating the test when the need arises.
* **Helpful to testers:**Since tests can run automatically when the source code changes and notify the testers of any issue, it is very helpful to both testers and developers.
* **Return on investment is high:** Automation help businesses complete the testing process faster with greater accuracy and coverage, thereby resulting in a high return on investment.

1. **What are all Browser driver used in selenium?**

***Answer:***

Selenium browser drivers are native to each browser, interacting with the browser by establishing a secure connection. Selenium supports different browser drivers such as ChromeDriver, GeckoDriver, Microsoft Edge WebDriver, SafariDriver, and InternetExplorerDriver.

**Chrome specific functionality**:

ChromeDriver is a separate executable that Selenium WebDriver uses to control chrome. It is maintained by the Chromium team with help from WebDriver contributors. If you are unfamiliar with selenium WebDriver, you should check out the Selenium Site.

These are capabilities and features specific to Google Chrome browsers.

* To execute tests in the Chrome browser using Selenium WebDriver, you need to use ChromeDriver.
* ChromeDriver is a separate executable used by Selenium WebDriver to control the Chrome browser.
* Download the latest version of ChromeDriver and control the Chrome browser during your tests
* Use the version of ChromeDriver that matches the version of your Chrome browser.

**Edge specific functionality:**

A WebDriver implementation that controls an Edge browser running on the local machine. It requires an edgedriver executable to be available in PATH.

* One possible WebDriver framework you can use is Selenium WebDriver, which is part of the Selenium, suite of tools. Selenium WebDriver is an open source testing framework that can be used on any platform, and provides language bindings for java, python 3, c#, Ruby, and JavaScript.
* If you use Selenium to automate Microsoft Edge, You must use Selenium 4; Selenium 3 is no longer supported.
* These are capabilities and features specific to Mozilla Firefox browsers.

**Firefox specific functionality:**

A Firefox Driver for selenium is a web driver for the Firefox browser that allows you to programmatically interact with a web page in a Firefox browser using selenium. It helps you automate various actions as if you were interacting with the website manually. This can be useful for tasks such as Web scraping, automation testing and automating repetitive tasks on a website.

Marionette is an automation driver for Mozilla's Gecko engine. It can remotely control either the UI or the internal JavaScript of a Gecko platform, such as Firefox. These are capabilities and feature specific to Mozilla Firefox browsers.

**Significance of using GeckoDriver:**

GeckoDriver is used as a proxy between Selenium and the Firefox browser for the automated browser tests. Since Selenium is designed to be cross-browser compatible, GeckoDriver enables Selenium to interact with Firefox in a similar way it interacts with other browsers like Chrome, Internet Explorer and Edge.

Firefox version in Selenium: The best version of Firefox to use with Selenium depends on the specific requirements of the project, as well as the versions of Selenium and the Firefox browser being used. Selenium supports a wide range of Firefox versions, but the most recent version of the Firefox is recommended as it will have the latest features and bug fixes. It’s also important to note that the version of the firefox used shall be compatible with the Selenium WebDriver.

**IE Specific functionality:**

The Selenium IE Driver (also known as Selenium Internet Explorer Driver) is a stand- alone server that implements the Web Driver protocol. This Server acts as a link between selenium script and internet explorer.

The driver supports running 32-bit and 64-bit versions of the browser. The choice of how to determine which “bit-ness” to use in launching the browser depends on which version of the IEDriverServer.exe is launched. If the 32-bit version of IE Driverserver.exe is launched, the 32-bit version of IE will be launched. Similarly, if the 64-bit version of IEDriverServer.exe is launched, the 64-bit version of IE will be launched. Runs in a real browser and supports JavaScript

* The InternetExplorer driver will only work on Windows.
* Comparatively slow (though still pretty snappy.

**Safari specific functionality:**

Run safaridriver --enable once in a terminal to enable Safari's WebDriver.  (If you're upgrading from a previous macOS release, you may need to prefix the command with sudo . Now you can use --safari to run your SeleniumBase tests on Safari.

WebDriver is a REST API. It hosts a local web server that accepts REST-style HTTP requests, so it can accept automation commands from a wide variety of test setups.

To support WebDriver without sacrificing a user’s privacy or security, Safari’s driver provides extra safeguards to ensure that test execution is isolated from normal browsing data and from other test runs. The driver is available in Safari 10 or later.

Safari Browser implements the WebDriver protocol using SafariDriver. The SafariDriver is the link between your tests in Selenium and the Safari Browser. SafariDriver has been implemented as a plugin in safari browser and this provides a perfect match of client and server machine where SafariDriverServer acts as server and Selenium-Java/Language binding acts as client. These are capabilities and features specific to Apple Safari browsers.

1. **What are the steps to create a simple Web Driver Script? Explain with code.**

***Answer:***

The Steps to Create a Simple WebDriver Script is detailed below:

**Set Up Development Environment:**

* Install Java Development Kit (JDK)
* Set up a Java Integrated Development Environment (IDE) like Eclipse or IntelliJ.
* Download the Selenium WebDriver Java bindings and ChromeDriver.

**Create a New Java Project:**

* Open your IDE and create a new Java project.

**Configure Project with Selenium JARs:**

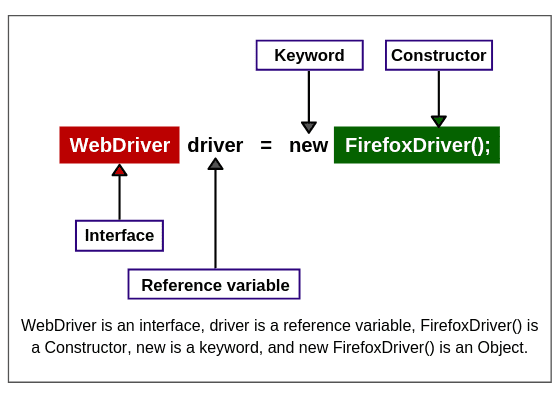
* Add the Selenium WebDriver JARs to project’s build path.
* Add the external JARs from the downloaded Selenium libraries.

**Write Selenium WebDriver Script:**

* Create a new Java class in your project.
* Write the script using WebDriver to interact with a web page.
* Set Up WebDriver (ChromeDriver in this example).
* Download ChromeDriver from [ChromeDriver Downloads](https://sites.google.com/chromium.org/driver/).
* Set the system property to the path of the ChromeDriver executable.

The Step by step of webdriver Script is detailed below along with the code:

* Firstly, open the Eclipse IDE.
* Create a Java Project. Go to File > New > Java Project and write “Selenium WebDriver Project”.
* Create a Package. Go to src > Right-click > New > Package and declare a name “newProject”.
* Now create a Java class. Go to package “newProject”  > Right-click > New > Class and declare a class name “FirstTestScript”.
* Now create an object of a class FirefoxDriver by taking WebDriver reference. Since WebDriver is an interface, we can implement all methods of WebDriver interface. It’s not possible to create an object of the interface directly.



* Open URL of webpage in the browser by calling get () command. It takes a string URL as a parameter and returns nothing. The syntax is as follows:

String URL = <https://www.google.co.in>;

driver. get(URL);

* Wait for the whole page to be load in browser by using the following syntax.

driver. manage().timeouts().implicitly Wait(60, TimeUnit. Seconds);

* To close browser, we will call close () command by using a reference variable driver. The syntax is as follows:

driver.close();

**Running the Java Test Case:**

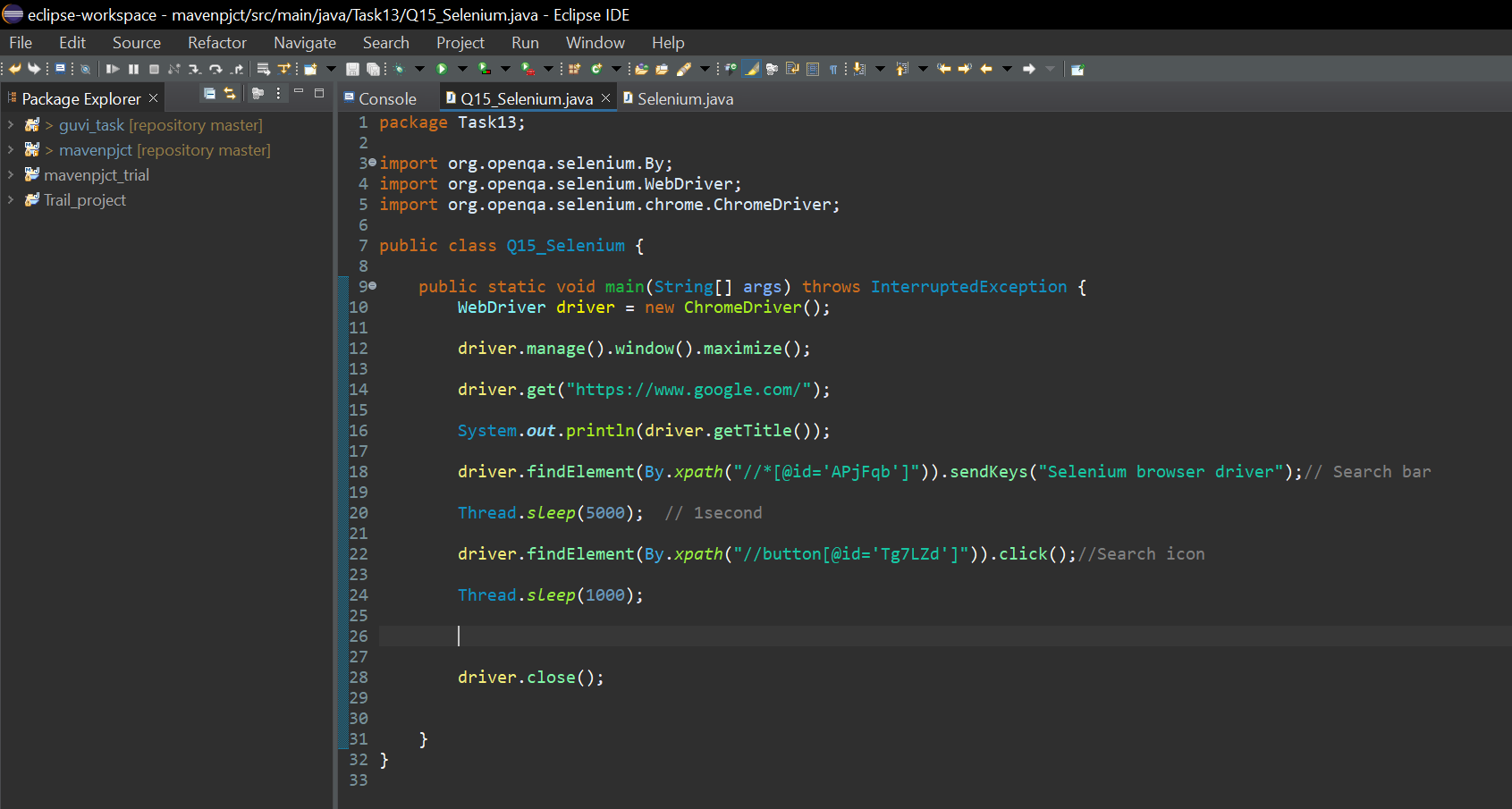
After resolving all errors, the code of automated test case needs to be executed in the Eclipse IDE. There are two ways to execute the test case. They are as follows:

* On Eclipse’s menu bar, click Run > Run As > Java Application or Right click on the eclipse code, click Run As > Java Application.
* Press Ctrl+F11 to run the entire code.

After a few seconds, a chrome browser will open and Selenium browser driver will launch the Google page with the help of your script. After launching completely, the browser will be closed after waiting a few seconds.

Once the execution is finished, a message “Website is opened successfully” will display on the console section of the Eclipse IDE.

Refer the code below executed. The snaps of the output are uploaded in GitHub and shared via the task link.



**---**