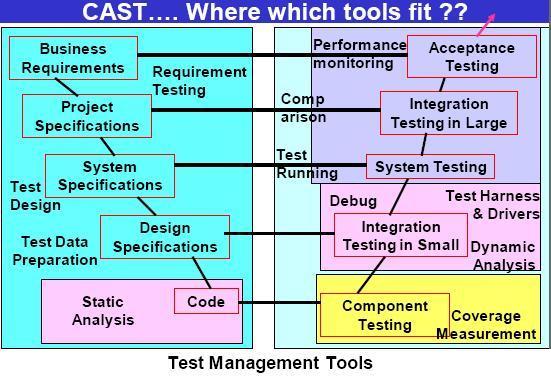
**Notes for MCA-II (Semester- III)**

**Subject :- Software Testing & Quality Assurance**

**(Subject Code:- IT-33)**

**Chapter 6 :-** **Tool Support for Testing**

* **6.1 Computer Aided Software Testing tools (CAST):**
* CAST tools are software applications used in the process of software testing.
* Such tools are available from various vendors and there are different tools for different types of testing, as well as for test management.
* 'Cast is also a multimedia professional development tool or multimedia database'.
* **CAST tools are available for all stages of SDLC phases or Types of CAST tools:**

****

**Requirements Testing Tools:**

* It is an automated support for verification & Validation of requirement models.
* It is responsible for :
* Consistency Checking
* Animation (like What-if scenarios)
* Collaborative Review of Requirements
* Traceability to design, Code, Tests
* Applicable at requirement & Specification Phase
* It is a requirements management tool supporting testing.

**2. Static Analysis Tool:**

* Used at coding phase.
* Provide information about the quality of software.
* Here code is examined, not executed
* Uses objective measures like cyclomatric complexity, nesting levels, size etc.
* It’s a form of automated inspection.

**3. Test Design Tools:**

* Used at requirement & Design Specification Phase.
* Generates test inputs (from a formal specification, from code etc.)

**4. Test Data Preparation Tools:**

* Used at requirement & Design Specification Phase.
* Used for data manipulation like selection from existing data base or files, edited from other sources, created according to some rules etc.

**5. Test Running Tools:**

* This tool is used at system level.
* Used for interface to the software being tested.
* Provide test scripts in a programmable language.
* Most often used to automate the regression testing.
* Here data, test inputs & expected results held in test repositories.
* Two types of test running tools are available:
* Character Based: (Capture keystrokes & screen responses)
* GUI Based: Simulates user interaction for WIMP applications, Capture mouse movement**,** button clicks, capture screens characters, object states, etc.

**6. Comparison Tools:**

* Applicable for system level, Integration at large & Acceptance level.
* It detects differences between actual test results & expected results w.r.t. screens, characters, masking & filtering.
* Test running tools normally include comparison tools for files or databases.
* These are standalone comparison tools for files or data bases.

**7. Test Harness & Drivers:**

* Used to exercise software which does not have user interface (Yet).
* Use to run groups of automated tests or comparisons.
* Often custom build.
* Used with simulators.

**8. Performance Testing Tools:**

* Used to test the load generation.
* To observe the number of transactions.
* To measure response time.

**9. Dynamic Analysis Tools:**

* Provide run-time information on software like allocation, memory leak, pointer arithmetic faults etc.
* Such tools perform network management like transaction rates measurements, bandwidth usage, potential network bottlenecks etc.

**10. Debugging Tools:**

* Used by programmers when investigating, fixing & Testing faults.
* Used to reproduce faults & examine program execution in detail.

**11. Test Management Tools:**

* Used for the management of test wares like test plans, test specifications, test results etc.
* Used for the project management of the test process i.e. estimation, schedule tests, log results etc.
* Work as incident management tools.
* Used for traceability of tests to requirements & designs.

**12. Coverage Tools:**

* Measure that what part of the software structure were executed by tests.
* Coverage starts with static analysis but executed & so is dynamic.
* Tests are run through the instrumented code.
* Measures different types of coverage like statement, branch, condition, path etc.
* **6.1 Types of Test tools**

The [software testing](https://www.javatpoint.com/software-testing-tutorial) tools can be categorized, depending on the licensing (paid or commercial, open-source), technology usage, type of testing.

**The software testing tools can be divided into the following:**

* Test management tool
* Bug tracking tool
* Automated testing tool
* Performance testing tool
* Cross-browser testing tool
* Integration testing tool
* Unit testing tool
* Mobile/android testing tool
* GUI testing tool
* Security testing tool

**Test management tool :-**

Test management tools are used to keep track of all the testing activity, fast data analysis, manage manual and automation test cases, various environments, and plan and maintain [manual testing](https://www.javatpoint.com/manual-testing)

**Bug tracking tool:-**

The defect tracking tool is used to keep track of the bug fixes and ensure the delivery of a quality product. This tool can help us to find the bugs in the testing stage so that we can get the defect-free data in the production server. With the help of these tools, the end-users can allow reporting the bugs and issues directly on their applications.

### Automation testing tool :-

This type of tool is used to enhance the productivity of the product and improve the accuracy. We can reduce the time and cost of the application by writing some test scripts in any programming language.

### Performance testing tool :-

Performance or Load testing tools are used to check the load, stability, and scalability of the application. When n-number of the users using the application at the same time, and if the application gets crashed because of the immense load, to get through this type of issue, we need load testing tools.

### Cross-browser testing tool:-

This type of tool is used when we need to compare a web application in the various web browser platforms. It is an important part when we are developing a project. With the help of these tools, we will ensure the consistent behavior of the application in multiple devices, browsers, and platforms.

### Integration testing tool:-

This type of tool is used to test the interface between modules and find the critical bugs that are happened because of the different modules and ensuring that all the modules are working as per the client requirements.

### Unit testing tool:-

This testing tool is used to help the programmers to improve their code quality, and with the help of these tools, they can reduce the time of code and the overall cost of the software.

### Mobile/android testing tool:-

We can use this type of tool when we are testing any mobile application. Some of the tools are open-source, and some of the tools are licensed. Each tool has its functionality and features.

### GUI testing tool:-

GUI testing tool is used to test the User interface of the application because a proper [GUI](https://www.javatpoint.com/gui-full-form)

(Graphical User Interface) is always useful to grab the user's attention. These type of tools will help to find the loopholes in the application's design and makes its better.

### Security testing tool:-

The security testing tool is used to ensure the security of the software and check for the security leakage. If any security loophole is there, it could be fixed at the early stage of the product. We need this type of the tool when the software has encoded the security code which is not accessible by the unauthorized users.

* **6.2 Effective Use of Tools: Potential Benefits and Risks:-**

**Benefits:**  
Test automation and tool support for testing can:  
- facilitate issues identification present in code via static or dynamic analysis

(playback/capture tools)

- allow to quickly assess software quality and delivers deliver real and lasting benefits as it is minimizing human effort in running low-value test cases.   
- facilitate access to testing information and detailed results (% coverage tests cases passed, incident rates, system behavior, performance..)

- make the test execution quicker, easier, consistent (objective assessment, response times logged) and less repetitive. For example, data driven testing and keyword driven testing are scripting technique used with test automation where all test inputs and expected results or keywords are stored in a table or spreadsheet and are used by the control script, it implies therefore automated testing of various use cases.

Potential benefits of using tools include:

* Repetitive work is reduced (e.g., running regression tests, re-entering the same test data, and checking against coding standards)
* Greater consistency and repeatability (e.g., tests executed by a tool in the same order with the same frequency, and tests derived from requirements)
* Objective assessment (e.g., static measures, coverage)
* Ease of access to information about tests or testing (e.g., statistics and graphs about test progress, incident rates and performance)

**Risks**:  
  
Test automation and tool support for testing can:

- lead to underestimating the time, cost, and effort required for introducing/maintaining any tool and achieving significant benefits additionally to the risk of setting unrealistic objectives (usability, functional testing).  
- create over-reliance and disturb fit for purpose testing, increase risks in neglecting version control and forget about system interoperability.

Risks of using tools include:

* Unrealistic expectations for the tool (including functionality and ease of use)
* Underestimating the time, cost and effort for the initial introduction of a tool (including training and external expertise)
* Underestimating the time and effort needed to achieve significant and continuing benefits from the tool (including the need for changes in the testing process and continuous improvement of the way the tool is used)
* Underestimating the effort required to maintain the test assets generated by the tool
* Over-reliance on the tool (replacement for test design or use of automated testing where manual testing would be better)
* Neglecting version control of test assets within the tool
* Neglecting relationships and interoperability issues between critical tools, such as requirements management tools, version control tools, incident management tools, defect tracking tools and tools from multiple vendors
* Risk of tool vendor going out of business, retiring the tool, or selling the tool to a different vendor
* Poor response from vendor for support, upgrades, and defect fixes
* Risk of suspension of open-source / free tool project
* Unforeseen, such as the inability to support a new platform
* **6.3 Introduction of a tool into an organization:-**

**Introducing the selected tool into an organization starts with a pilot project, which has the following objectives:**

* Learn more detail about the tool
* Evaluate how the tool fits with existing processes and practices, and determine what would need to change
* Decide on standard ways of using, managing, storing and maintaining the tool and the test assets (e.g., deciding on naming conventions for files and tests, creating libraries and defining the modularity of test suites)
* Assess whether the benefits will be achieved at reasonable cost

**Success factors for the deployment of the tool within an organization include**:

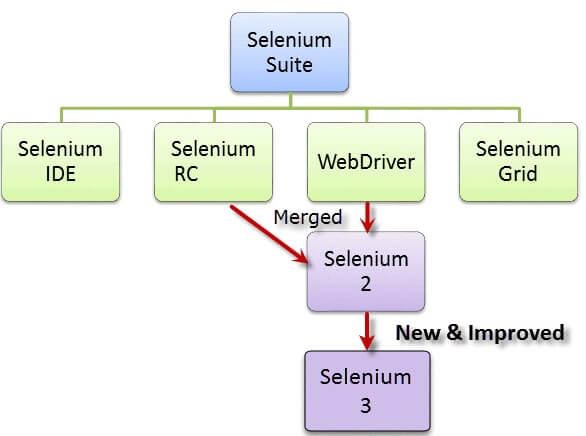
* Rolling out the tool to the rest of the organization incrementally
  + Adapting and improving processes to fit with the use of the tool
  + Providing training and coaching/mentoring for new users
* Defining usage guidelines
* Implementing a way to gather usage information from the actual use
* Monitoring tool use and benefits
* Providing support for the test team for a given tool

Gathering lessons learned from all teams

* **6.4.1 Selenium -WebDriver and Test NG:-**

Selenium is a free (open-source) automated testing framework used to validate web applications across different browsers and platforms. You can use multiple programming languages like Java, C#, Python etc to create Selenium Test Scripts. Testing done using the Selenium testing tool is usually referred to as Selenium Testing.

Selenium Software is not just a single tool but a suite of software, each piece catering to different Selenium QA testing needs of an organization

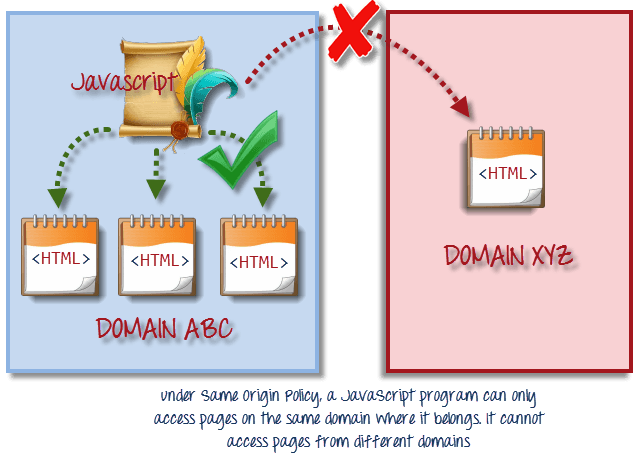


Primarily, Selenium was **created by Jason Huggins in 2004**. An engineer at ThoughtWorks, he was working on a web application that required frequent testing. Having realized that the repetitious [Manual Testing](https://www.guru99.com/manual-testing.html) of their application was becoming more and more inefficient, he created a[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)program that would automatically control the browser’s actions. He named this program as the “**JavaScriptTestRunner**.”

Seeing potential in this idea to help automate other web applications, he made JavaScriptRunner open-source which was later re-named as **Selenium Core**.

## The Same Origin Policy Issue:-

**Same Origin policy prohibits JavaScript code from accessing elements from a domain that is different from where it was launched**. Example, the HTML code in www.google.com uses a JavaScript program “randomScript.js”. The same origin policy will only allow randomScript.js to access pages within google.com such as google.com/mail, google.com/login, or google.com/signup. However, it cannot access pages from different sites such as yahoo.com/search or rediff.com because they belong to different domains.



This is the reason why prior to Selenium RC, testers needed to install local copies of both Selenium Core (a JavaScript program) and the web server containing the web application being tested so they would belong to the same domain

## Birth of Selenium Remote Control (Selenium RC):-

Unfortunately; testers using Selenium Core had to install the whole application under test and the web server on their own local computers because of the restrictions imposed by the **same origin policy.**So another ThoughtWork’s engineer, **Paul Hammant**, decided to create a server that will act as an HTTP proxy to “trick” the browser into believing that Selenium Core and the web application being tested come from the same domain. This system became known as the **Selenium Remote Control** or **Selenium 1**.

## Birth of Selenium Grid:-

Selenium Grid was developed by **Patrick Lightbody** to address the need of minimizing test execution times as much as possible. He initially called the system “**Hosted QA**.” It was capable of capturing browser screenshots during significant stages, and also of **sending out Selenium commands to different machines simultaneously.**

## Birth of Selenium IDE

**Shinya Kasatani**of Japan created **Selenium IDE**, a Firefox extension that can automate the browser through a record-and-playback feature. He came up with this idea to further increase the speed in creating test cases. He donated Selenium IDE to the Selenium Project in **2006**.

## Birth of WebDriver

**Simon Stewart**created WebDriver circa **2006** when browsers and web applications were becoming more powerful and more restrictive with JavaScript programs like Selenium Core. **It was the first cross-platform testing framework that could control the browser from the OS level.**

## Birth of Selenium 2

In **2008**, the whole Selenium Team decided to merge WebDriver and Selenium RC to form a more powerful tool called **Selenium 2**, with **WebDriver being the core**. Currently, Selenium RC is still being developed but only in maintenance mode. Most of the Selenium Project’s efforts are now focused on Selenium 2.

## So, Why the Name Selenium?

**The Name “Selenium”** came from a joke which Jason cracked once to his team. During Selenium’s development, another [automated testing](https://www.guru99.com/automation-testing.html) framework was popular made by the company called **Mercury Interactive**(yes, the company who originally made QTP before it was acquired by HP). Since ***Selenium is a well-known antidote (Medicine) for Mercury poisoning, Jason suggested that name and his teammates took it***. So that is how we got to call this framework up to the present.



**The entire Selenium Software Testing Suite is comprised of four components:**

**Selenium IDE**, a Firefox add-on that you can only use in creating relatively simple test cases and test suites.

**Selenium Remote Control**, **also known as Selenium 1**, which is the first Selenium tool that allowed users to use programming languages in creating complex tests.

**WebDriver**, the newer breakthrough that allows your test scripts to communicate directly to the browser, thereby controlling it from the OS level.

**Selenium Grid** is also a tool that is used with Selenium RC to execute parallel tests across different browsers and operating systems.

***Selenium RC*** *and* ***WebDriver*** *was merged**to form* ***Selenium 2***.

## What is Selenium Webdriver?

**Selenium Webdriver** is an open-source collection of APIs which is used for testing web applications. The Selenium Webdriver tool is used for automating web application testing to verify that it works as expected or not. It mainly supports browsers like Firefox, Chrome, Safari and Internet Explorer. It also permits you to execute cross-browser testing.

WebDriver also enables you to **use a programming language** in creating your test scripts (not possible in Selenium IDE).

The WebDriver proves itself to be **better than both Selenium IDE and Selenium RC** in many aspects. It implements a more modern and stable approach in automating the browser’s actions. WebDriver, unlike Selenium RC, does not rely on JavaScript for Selenium Automation Testing. **It controls the browser by directly communicating with it.**

You can now use **conditional operations** like if-then-else or switch-case. You can also perform looping like do-while.



## Difference between Selenium RC and Webdriver

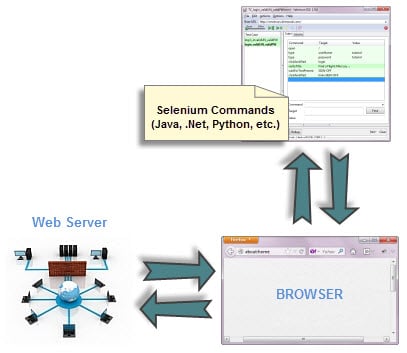
Before the advent of WebDriver in  2006, there was another, **automation tool called Selenium Remote Control.**Both WebDriver and Selenium RC have following features:

* They both allow you to use a programming language in designing your test scripts.
* They both allow you to run your tests against different browsers.

## 1.Architecture

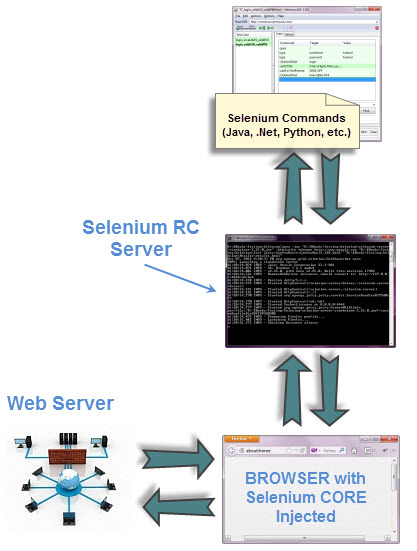
**WebDriver’s architecture is simpler than Selenium RC’s**.

* It controls the browser from the OS level
* All you need are your programming language’s IDE (which contains your Selenium commands) and a browser.



##### **Selenium RC’s architecture is way more complicated.**

* You first need to launch **a separate application called Selenium Remote Control (RC) Server** before you can start testing
* The Selenium RC Server **acts as a “middleman” between your Selenium commands and your browser**
* When you begin testing, Selenium RC Server “injects” a **Javascript program called Selenium Core** into the browser.
* Once injected, Selenium Core will start receiving instructions relayed by the RC Server from your test program.
* When the instructions are received, **Selenium Core will execute them as Javascript commands.**
* The browser will obey the instructions of Selenium Core and will relay its response to the RC Server.
* The RC Server will receive the response of the browser and then display the results to you.
* RC Server will fetch the next instruction from your test script to repeat the whole cycle.



## 2. Speed



**WebDriver is faster than Selenium RC since it**speaks directly to the browser uses the browser’s own engine to control it.



Selenium RC is slower since it uses a Javascript program called Selenium Core. **This Selenium Core is the one that directly controls the browser, not you.**

## 3. Real-life Interaction

##### Introduction to WebDriver & Comparison with Selenium RC

**WebDriver interacts with page elements in a more realistic way.** For example, if you have a disabled text box on a page you were testing, WebDriver really cannot enter any value in it just as how a real person cannot.



Selenium Core, just like other[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)codes, can access disabled elements. In the past, Selenium testers complain that Selenium Core was able to enter values to a disabled text box in their tests. Differences in API

## 4. API



Selenium RC’s API is more matured but contains redundancies and often confusing commands. For example, most of the time, testers are confused whether to use type or typeKeys; or whether to use click, mouseDown, or mouseDownAt. Worse, different browsers interpret each of these commands in different ways too!

WebDriver’s API is simpler than Selenium RC’s. It does not contain redundant and confusing commands.

## 5. Browser Support



**WebDriver can support the headless HtmlUnit browser**

HtmlUnit is termed as “headless” because it is an invisible browser – it is GUI-less.

It is a very fast browser because no time is spent in waiting for page elements to load. This accelerates your test execution cycles.

Since it is invisible to the user, it can only be controlled through automated means.

**Selenium RC cannot support the headless HtmlUnit browser.**It needs a real, visible browser to operate on.

## Limitations of WebDriver

WebDriver Cannot Readily Support New Browsers

Remember that WebDriver operates on the OS level. Also, remember that different browsers communicate with the OS in different ways. If a new browser comes out, it may have a different process of communicating with the OS as compared to other browsers. So, **you have to give the WebDriver team quite some time to figure that new process out** before they can implement it on the next WebDriver release.

## Summary

* WebDriver is a tool for testing web applications **across different browsers** using different programming languages.
* You are now able to make powerful tests because WebDriver **allows you to use a programming language** of your choice in designing your tests.
* WebDriver is **faster than Selenium RC** because of its simpler architecture.
* WebDriver **directly talks to the browser** while Selenium RC needs the help of the RC Server in order to do so.
* WebDriver’s API is more**concise** than Selenium RC’s.
* WebDriver **can support HtmlUnit** while Selenium RC cannot.

## The drawbacks of WebDriver are:

* + It **cannot readily support new browsers**, but Selenium RC can.
  + It **does not have a built-in command** for automatic generation of test results.

# How to Download & Install Selenium WebDriver

Selenium installation is a 3 step process:

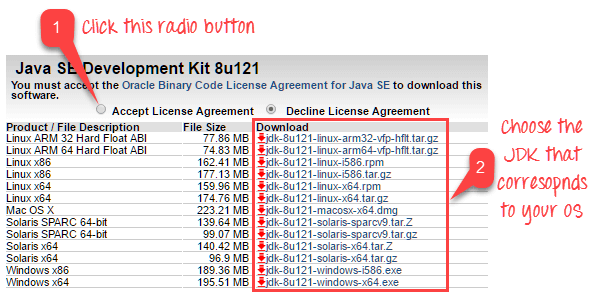
1. Install Java SDK
2. Install Eclipse
3. Install Selenium Webdriver Files

## Step 1 – Install Java on your computer

Download and install the **Java Software Development Kit (JDK)**

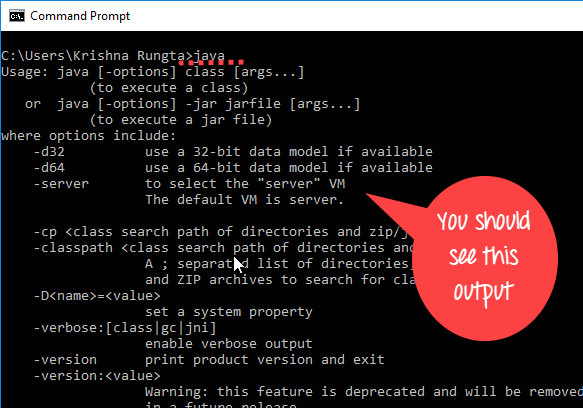


**Next –**



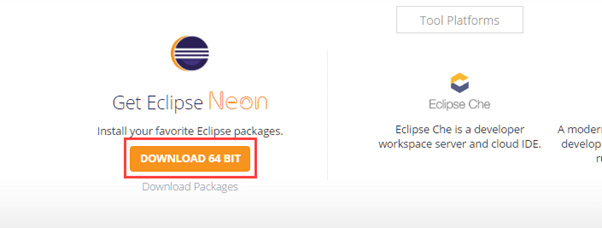
This JDK version comes bundled with Java Runtime Environment (JRE), so you do not need to download and install the JRE separately.

Once installation is complete, open command prompt and type “java”. If you see the following screen you are good to move to the next step

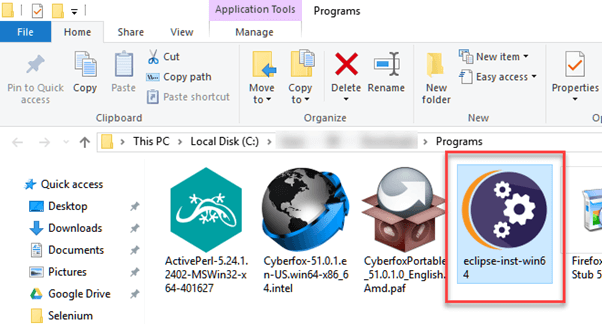


## Step 2 – Install Eclipse IDE

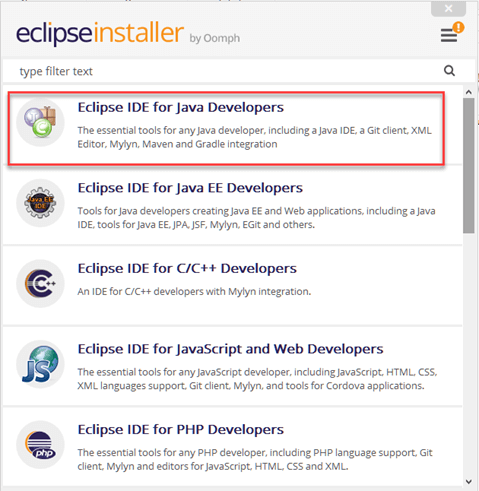
Download latest version of **“Eclipse IDE for Java Developers”** . Be sure to choose correctly between Windows 32 Bit and 64 Bit versions.



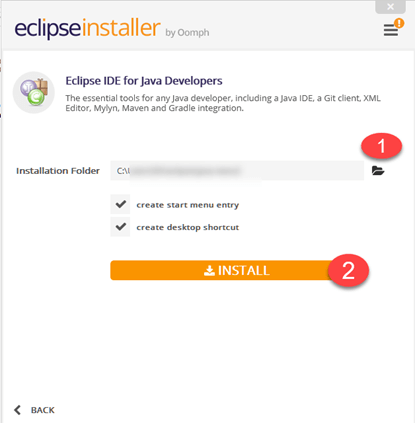
You should be able to download an exe file named “eclipse-inst-win64” for Setup.



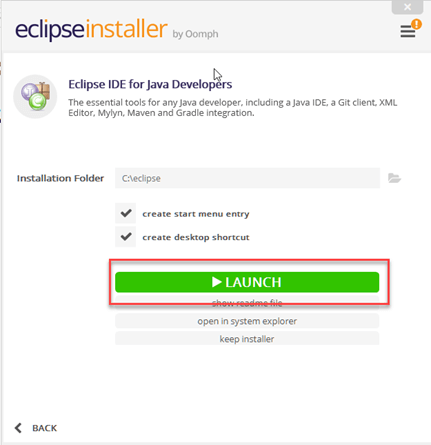
Double-click on file to Install the Eclipse. A new window will open. Click Eclipse IDE for Java Developers.



After that, a new window will open which click button marked 1 and change path to “C:\eclipse”. Post that Click on Install button marked 2



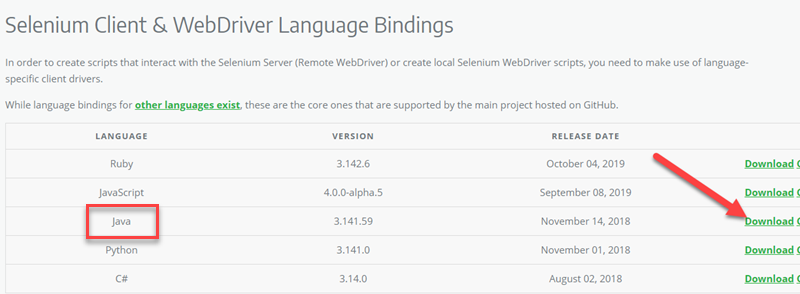
After successful completion of the installation procedure, a window will appear. On that window click on Launch



This will start eclipse neon IDE for you.

**Step 3 – Download the Selenium Java Client Driver**

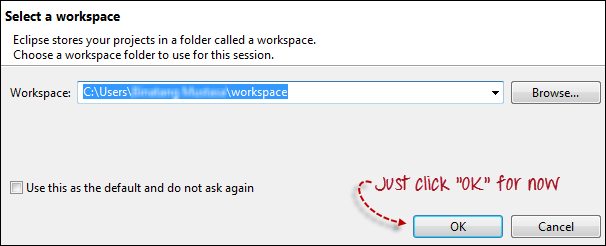
You can download **Selenium Webdriver for Java Client Driver** [here](https://selenium.dev/downloads/). You will find client drivers for other languages there, but only choose the one for Java.



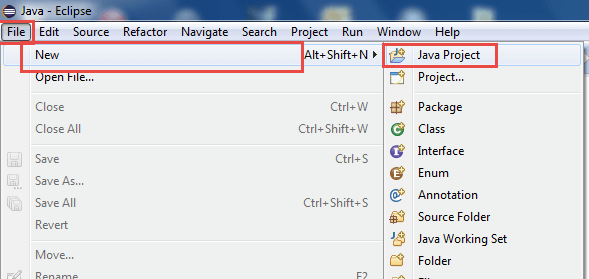
This download comes as a ZIP file named “selenium-3.14.0.zip”. For simplicity of Selenium installation on Windows 10, extract the contents of this ZIP file on your C drive so that you would have the directory “C:\selenium-3.14.0\”. This directory contains all the JAR files that we would later import on Eclipse for Selenium setup.

## Step 4 – Configure Eclipse IDE with WebDriver

1. Launch the “eclipse.exe” file inside the “eclipse” folder that we extracted in step 2. If you followed step 2 correctly, the executable should be located on C:\eclipse\eclipse.exe.
2. When asked to select for a workspace, just accept the default location.

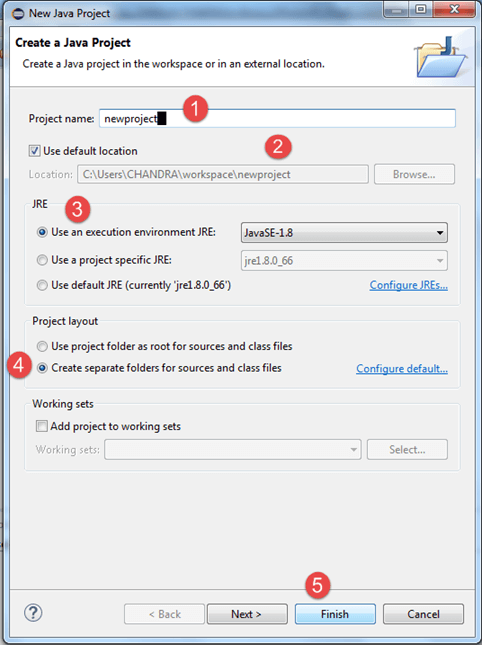


3. Create a new project through File > New > Java Project. Name the project as “newproject”.



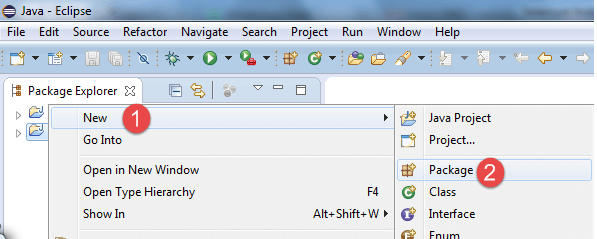
A new pop-up window will open enter details as follow

1. Project Name
2. Location to save project
3. Select an execution JRE
4. Select layout project option
5. Click on Finish button



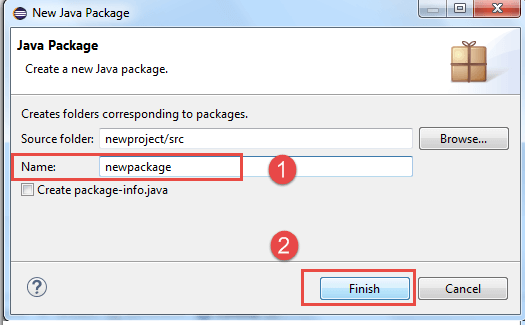
4. In this step,

1. Right-click on the newly created project and
2. Select New > Package, and name that package as “newpackage”.

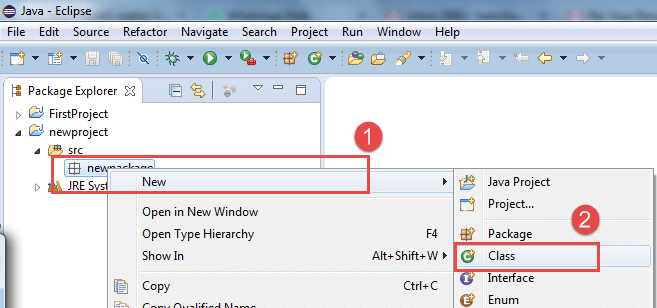


A pop-up window will open to name the package,

1. Enter the name of the package
2. Click on Finish button

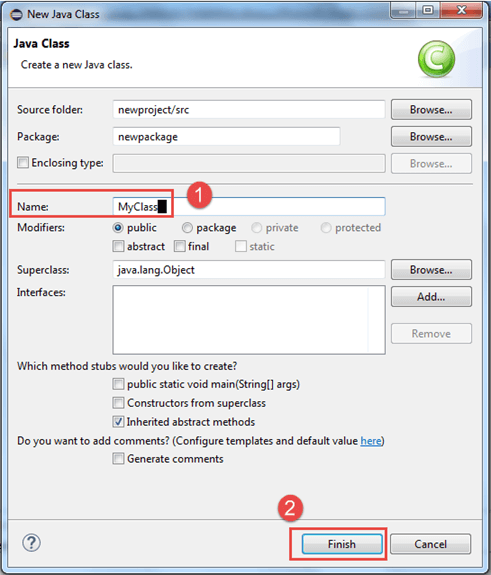


5. Create a new Java class under newpackage by right-clicking on it and then selecting- New > Class, and then name it as “MyClass”. Your Eclipse IDE should look like the image below.

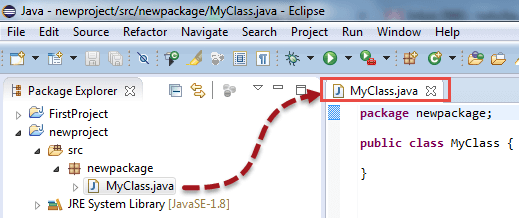


When you click on Class, a pop-up window will open, enter details as

1. Name of the class
2. Click on Finish button



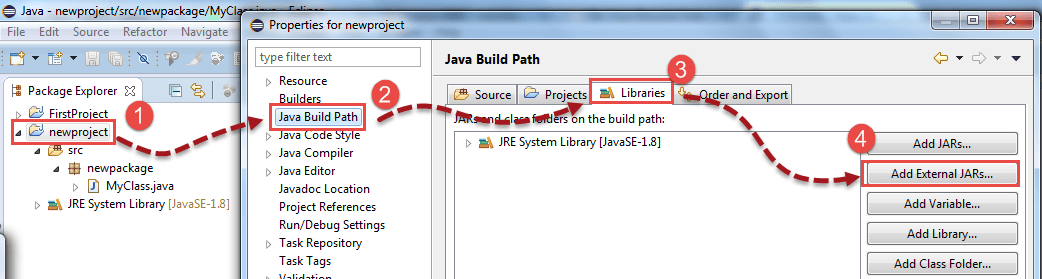
This is how it looks like after creating class.



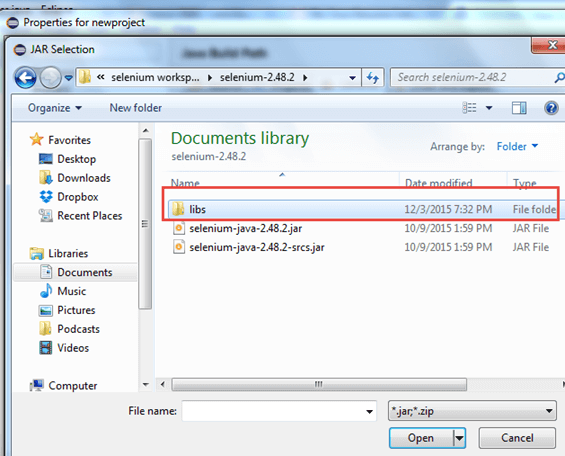
Now selenium WebDriver’s into Java Build Path

In this step,

1. Right-click on “newproject” and select **Properties**.
2. On the Properties dialog, click on “Java Build Path”.
3. Click on the **Libraries** tab, and then
4. Click on “Add External JARs..”

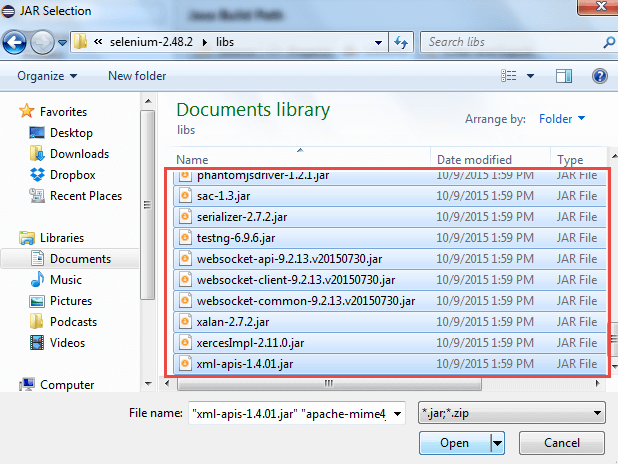


When you click on “Add External JARs..” It will open a pop-up window. Select the JAR files you want to add.

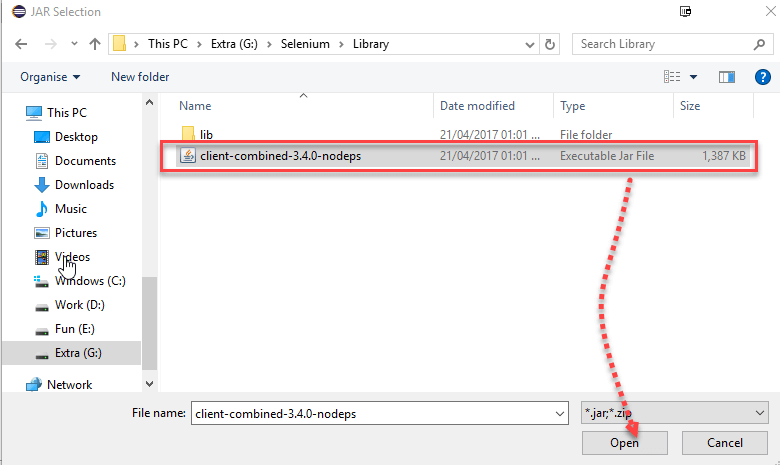


After selecting jar files, click on OK button.

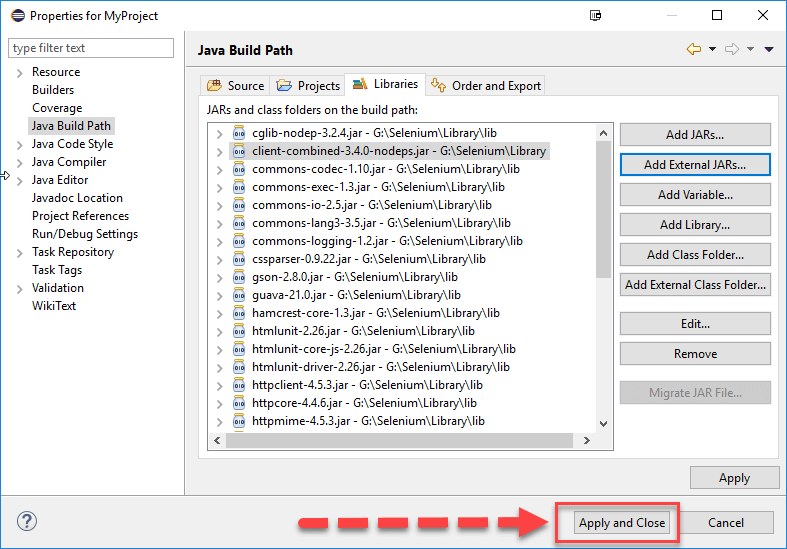
Select all files inside the lib folder.



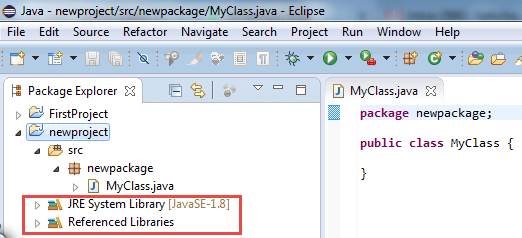
Select files outside lib folder



Once done, click “Apply and Close” button



6. Add all the JAR files inside and outside the “libs” folder. Your Properties dialog should now look similar to the image below.



7. Finally, click OK and we are done importing Selenium libraries into our project.

### Different Drivers

**HTMLUnit and Firefox are two browsers that WebDriver can directly automate** – meaning that no other separate component is needed to install or run while the test is being executed. For other browsers, a separate program is needed. That program is called as the **Driver Server**.

A driver server is different for each browser. For example, Internet Explorer has its own driver server which you cannot use on other browsers. Below is the list of driver servers and the corresponding browsers that use them.

You can download these drivers

| **Browser** | **Name of Driver Server** | **Remarks** |
| --- | --- | --- |
| HTMLUnit | HtmlUnitDriver | WebDriver can drive HTMLUnit using HtmlUnitDriver as driver server |
| Firefox | Mozilla GeckoDriver | WebDriver can drive Firefox without the need of a driver server Starting Firefox 45 & above one needs to use gecko driver created by Mozilla for automation |
| Internet Explorer | Internet Explorer Driver Server | Available in 32 and 64-bit versions. Use the version that corresponds to the architecture of your IE |
| Chrome | ChromeDriver | Though its name is just “ChromeDriver”, it is, in fact, a Driver Server, not just a driver. The current version can support versions higher than Chrome v.21 |
| Opera | OperaDriver | Though its name is just “OperaDriver”, it is, in fact, a Driver Server, not just a driver. |
| PhantomJS | GhostDriver | PhantomJS is another headless browser just like HTMLUnit. |
| Safari | SafariDriver | Though its name is just “SafariDriver”, it is, in fact, a Driver Server, not just a driver. |

## Summary

Aside from a browser, you will need the following to start using WebDriver

* **Java Development Kit (JDK).** <https://www.oracle.com/java/technologies/javase-downloads.html>
* **Eclipse IDE** – [http://www.eclipse.org/downloads/](https://www.eclipse.org/downloads/)
* **Java Client Driver** – <https://www.selenium.dev/downloads/>

When starting a WebDriver project in Eclipse, do not forget to import the Java Client Driver files onto your project. These files will constitute your Selenium Library.

With new version of [Selenium](https://www.guru99.com/selenium-tutorial.html), there is no browser that you can automate without the use of a Driver Server.

## What is TestNG?

**TestNG**is an automation testing framework in which NG stands for “Next Generation”. TestNG is inspired by [JUnit](https://www.guru99.com/junit-tutorial.html)which uses the annotations (@). TestNG overcomes the disadvantages of JUnit and is designed to make [end-to-end testing](https://www.guru99.com/end-to-end-testing.html) easy.

Using TestNG, you can generate a proper report, and you can easily come to know how many test cases are passed, failed, and skipped. You can execute the failed test cases separately.

For example:

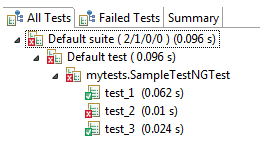
* Suppose, you have five test cases, one method is written for each test case (Assume that the program is written using the main method without using testNG). When you run this program first, three methods are executed successfully, and the fourth method is failed. Then correct the errors present in the fourth method, now you want to run only fourth method because first three methods are anyway executed successfully. This is not possible without using TestNG.
* The TestNG in Selenium provides an option, i.e., testng-failed.xml file in test-output folder. If you want to run only failed test cases means you run this XML file. It will execute only failed test cases.

## Why Use TestNG with Selenium?

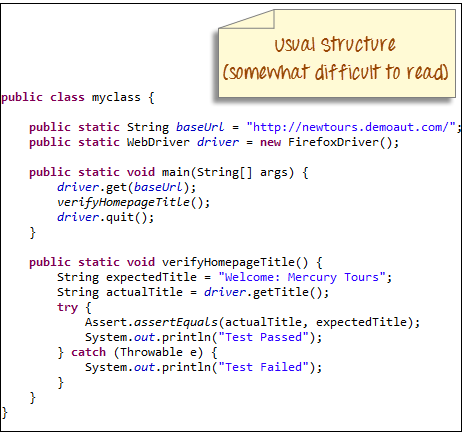
Default Selenium tests do not generate a proper format for the test results. Using TestNG in Selenium, we can generate test results.

Most Selenium users use this more than[Junit](https://www.guru99.com/junit-tutorial.html)because of its advantages. There are so many features of TestNG, but we will only focus on the most important ones that we can use in Selenium. Following are the key features of Selenium TestNG:

* Generate the report in a proper format including a number of test cases runs, the number of test cases passed, the number of test cases failed, and the number of test cases skipped.
* Multiple test cases can be grouped more easily by converting them into testng.xml file. In which you can make priorities which test case should be executed first.
* The same test case can be executed multiple times without loops just by using keyword called ‘invocation count.’
* Using testng, you can execute multiple test cases on multiple browsers, i.e., cross [browser testing](https://www.guru99.com/top-10-cross-browser-testing-tools.html).
* The TestNG framework can be easily integrated with tools like TestNG Maven, Jenkins, etc.
* Annotations used in the testing are very easy to understand ex: @BeforeMethod, @AfterMethod, @BeforeTest, @AfterTest
* WebDriver has no native mechanism for generating reports. TestNG can generate the report in a readable format like the one shown below.



* TestNG simplifies the way the tests are coded. There is no more need for a static main method in our tests. The sequence of actions is regulated by easy-to-understand annotations that do not require methods to be static.





* Uncaught exceptions are automatically handled by TestNG without terminating the test prematurely. These exceptions are reported as failed steps in the report.

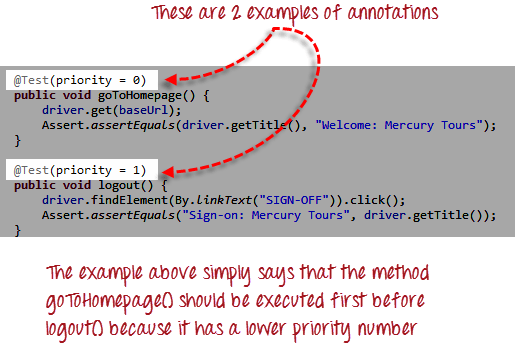
## Advantages of TestNG over JUnit

There are three major advantages of TestNG over JUnit:

* Annotations are easier to understand
* Test cases can be grouped more easily
* Parallel testing is possible

## What is Annotation in TestNG?

**Annotations in TestNG are lines of code that can control how the method below them will be executed**. They are always preceded by the @ symbol. A very early and quick TestNG Example is the one shown below.



It is just important to note for now that annotations in TestNG are easier to code and understand than in JUnit.

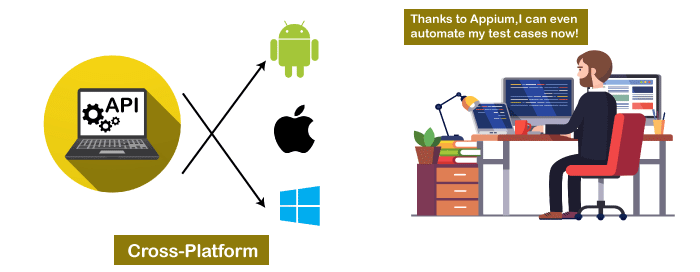
The ability to run tests in parallel is available in TestNG but not in JUnit, so the TestNG framework is more preferred for testers using Selenium Grid.

* **6.4.2 Appium :-**

**Appium** is an open-source automation mobile testing tool, which is used to test the application. It is developed and supported by Sauce Labs to automate native and hybrid mobile apps. It is a cross-platform mobile automation tool, which means that it allows the same test to be run on multiple platforms. Multiple devices can be easily tested by Appium in parallel.

In today's development area, the demand for mobile applications is high. Currently, people are converting their websites into mobile apps. Therefore, it is very important to know about mobile software automation testing technology and also stay connected with new technology. Appium is a mobile application testing tool that is currently trending in Mobile Automation Testing Technology.

Appium is used for automated testing of **native, hybrid, and web applications**. It supports automation test on the simulators (iOS) and emulators (Android) as well as physical devices (Android and iOS both). Previously, this tool mainly focused on IOS and Android applications that were limited to mobile application testing only. Few updates back, Appium declared that it would now support desktop application testing for windows as well.



Appium is very much similar to the **Selenium Webdriver** testing tool. So, if you already know Selenium Webdriver, Appium becomes very easy to learn. Appium has **NO dependency** on mobile device OS because it has a framework that converts the Selenium Webdriver commands to UIAutomator and UIAutomation commands for Android and iOS respectively, that depends on the device type rather than the OS type.

It supports several languages such as Java, PHP, Objective C, C#, Python, JavaScript with node.js, and Ruby, and many more that have Selenium client libraries. Selenium is the backend of Appium that provides control over the functionality of Selenium for testing needs.

**APPIUM** is a freely distributed open source mobile application UI[Testing](https://www.guru99.com/software-testing.html)framework. Appium allows native, hybrid and web application testing and supports automation test on physical devices as well as an emulator or simulator both. It offers cross-platform application testing, i.e. single API works for both Android and iOS platform test scripts.

It has **NO** dependency on Mobile device OS. Because APPIUM has framework or wrapper that translate[Selenium](https://www.guru99.com/selenium-tutorial.html)Webdriver commands into UIAutomation (iOS) or UIAutomator (Android) commands depending on the device type, not any OS type.

Appium supports all languages that have Selenium client libraries like- Java, Objective-C,[JavaScript](https://www.guru99.com/interactive-javascript-tutorials.html)with node.js, PHP, Ruby, Python,[C#,](https://www.guru99.com/c-sharp-tutorial.html)etc.

## Features of Appium

* Appium does not require application source code or library.
* Appium provides a strong and active community.
* Appium has multi-platform support i.e., it can run the same test cases on multiple platforms.
* Appium allows the parallel execution of test scripts.
* In Appium, a small change does not require re-installation of the application.
* Appium supports various languages like C#, Python, Java, Ruby, PHP, JavaScript with node.js, and many others that have Selenium client library.

## Advantages of Appium

* Appium is an open-source tool, which means it is freely available. It is easy to install.
* It allows the automated testing of hybrid, native, and web applications.
* Unlike other testing tools, you do not need to include any additional agents in your app to make Appium compatible with automation. It tests the same app, which is going to upload in App Store.
* An additional feature added to Appium. Now it would support desktop application testing for windows as well along with mobile application testing.
* Appium is a cross-platform, freely available mobile testing tool, which allows us the cross-platform mobile testing. This means you can test on multiple platforms (single API for both Android and IOS platforms).

## Disadvantages of Appium

Along with some features and advantages, Appium has some drawbacks too, which are as follows-

* Lack of detailed reports.
* Since the tests depend on the remote web driver, so it is a bit slow.
* It is not a limitation, but an overhead that Appium uses **UIAutomator** for Android that only supports Android SDK, API 16, or higher. However, Appium supports older APIs, but not directly. It uses another open-source library **Selendroid** to support older APIs.
* In iOS, only one instance (iOS Script) can run on one Mac OS device, which means one test can be executed at a time per Mac. If you want to run your tests on multiple iOS devices at the same time, you need to arrange the same number of Mac machines. But it would be expensive to arrange various Mac machines.

**Solution:** This problem can be resolved if you will run your script in the **mobile cloud of Sauce Lab**. Currently, it allows scripts to run on multiple iOS simulators at the same time.

## Appium Architecture

Appium is an HTTP server that is written in **node.js**. It starts a "**test case**" on the device that gives rise to a server and listens for proxied commands from the main Appium server. Tester writes the Test scripts to execute on device or Emulator. Several webdriver sessions for different platforms like Android and IOS are created and handled by the Appium.



Test Scripts written by the tester executes on the Emulator or device by sending them as requests to the Appium server. Each vendor, such as IOS or Android, has a different method and mechanism to execute test cases on the device. So, the test case executes after listening commands from the Appium server. Appium uses JSON wire protocols to send command requests to Appium server.

## How Appium work?

* When we install the Appium, a server is also installed with it on our machine that exposes the REST API.
* It receives command and connection requests from the client and executes that command on devices like iOS or Android.
* It replies with the HTTP responses.

To execute requests, it uses a mobile test automation framework to run the user interface of the app. For Example -

**Apple** instruments used for iOS

**Selendroid** used for Android API 15 or less

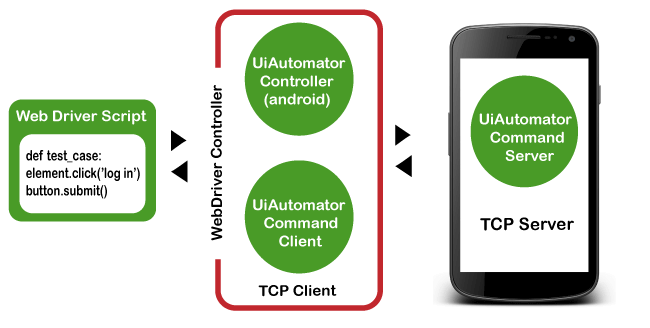
**UIAutomator** used for Android API 16 or higher

### Appium in Android:-

On Android, Appium proxies the command to a **UIAutomator** script running on the device. UIAutomator is a native UI automation framework of Android that allows you to run **Junit** test cases directly into the device using command line. Although it uses Java programming language, but Appium allows to run it from any WebDriver supported language.

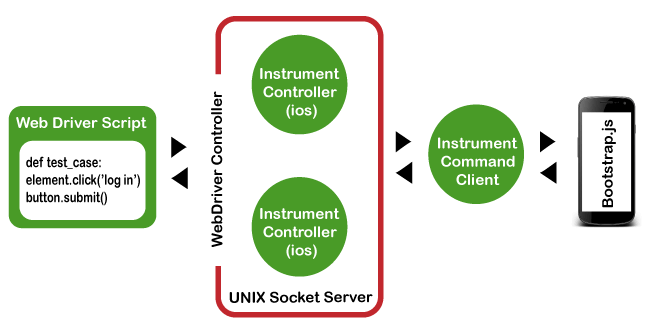
Android uses **bootstrap.jar**, which works as a TCP server. It is used to send the test commands to perform the actions on Android device using UIAutomator.

In the below figure, see the Appium architecture in respect to Android automation -



### Appium in iOS:-

### As Android uses UIAutomator, iOS uses UIAutomation. Similar to the Android, Appium proxies the command to a UIAutomation test case running on the Mac instruments environment. Apple provides this application "instrument" that performs various activities like building, profiling, and controlling iOS apps. On the other hand, it also has an automation component where you can write commands in JavaScript. It uses UIAutomation API to interact with Application UI. Appium uses same libraries to automate iOS Apps.



### Prerequisites to use Appium:-

### Install Java (JDK)

### Install Android Studio

### Install additional Android SDK tools

### Install Appium jar file

### js (Not required - It by default comes with "node.js" and "NPM", whenever the Appium server is installed. Therefore, it is not required to install node.js and NPM separately. It is already included in current version of Appium.)

### Install Appium Desktop Client

### Install Eclipse IDE for Java

### Appium Installation on windows:-

### Setting up Appium is not only the installation of a single tool or setting up a few variables. A complete setup of Appium requires the installation of many other tools. Then Appium starts testing of an application. This tutorial mainly focuses on Appium setup for Android on Windows Operating system.

### Install and setup Java

### With Appium, you need to write automation test scripts that can interact with your mobile screen and control fields (such as text boxes, buttons, etc.) in your mobile application, and also can perform actions on them. Java is used to write these automation test scripts. So, download the latest version of JDK and setup the environment variable.

### [Download the latest version of Java](https://www.oracle.com/in/java/technologies/javase-downloads.html" \t "_blank)

### Setup Java Environment Variable.

* **6.4.3 JMeter :-**

jMeter is an Open Source testing software. It is 100% pure Java application for load and performance testing.

jMeter is designed to cover various categories of tests such as load testing, functional testing, performance testing, regression testing, etc., and it requires JDK 5 or higher.

## What is JMeter?

JMeter is a software that can perform load test, performance-oriented business (functional) test, regression test, etc., on different protocols or technologies.

Stefano Mazzocchi of the Apache Software Foundation was the original developer of JMeter. He wrote it primarily to test the performance of Apache JServ (now called as Apache Tomcat project). Apache later redesigned JMeter to enhance the GUI and to add functional testing capabilities.

JMeter is a Java desktop application with a graphical interface that uses the Swing graphical API. It can therefore run on any environment / workstation that accepts a Java virtual machine, for example − Windows, Linux, Mac, etc.

Apache JMeter is pure Java-based open-source software designed to load test functional behaviour and measure performance. You can use JMeter to analyse and measure the performance of web applications or a variety of services. JMeter was originally used for testing Web Applications or FTP applications. Nowadays, it is used for functional testing, database server testing, etc.

**The protocols supported by JMeter are** −

* Web − HTTP, HTTPS sites 'web 1.0' web 2.0 (ajax, flex and flex-ws-amf)
* Web Services − SOAP / XML-RPC
* Database via JDBC drivers
* Directory − LDAP
* Messaging Oriented service via JMS
* Service − POP3, IMAP, SMTP
* FTP Service

## JMeter Features

Following are some of the features of JMeter −

* Being an open source software, it is freely available.
* It has a simple and intuitive GUI.
* JMeter can conduct load and performance test for many different server types − Web - HTTP, HTTPS, SOAP, Database via JDBC, LDAP, JMS, Mail - POP3, etc.
* It is a platform-independent tool. On Linux/Unix, JMeter can be invoked by clicking on JMeter shell script. On Windows, it can be invoked by starting the jmeter.bat file.
* It has full Swing and lightweight component support (precompiled JAR uses packages javax.swing.\* ).
* JMeter store its test plans in XML format. This means you can generate a test plan using a text editor.
* Its full multi-threading framework allows concurrent sampling by many threads and simultaneous sampling of different functions by separate thread groups.
* It is highly extensible.
* It can also be used to perform automated and functional testing of the applications.

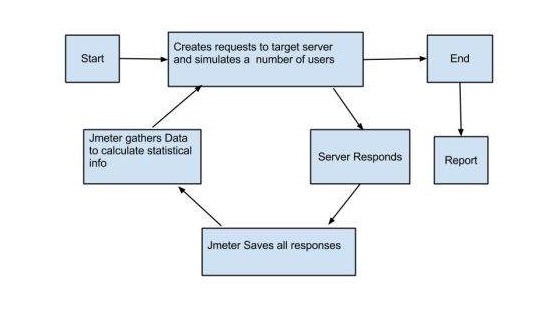
## Why Learn JMeter?

JMeter helps load test web servers, websites, and web applications by simulating real-user behaviours and testing environments. JMeter provides a user-friendly GUI, easy installation, testing strategies, simulation, and many other performance testing features. Learning JMeter helps you to ease your Performance testing tasks.

## How JMeter Works?

JMeter simulates a group of users sending requests to a target server, and returns statistics that show the performance/functionality of the target server/application via tables, graphs, etc.

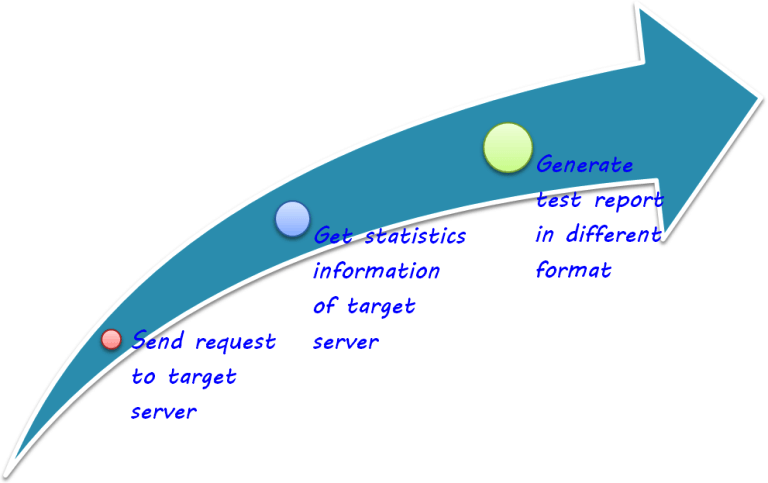
Take a look at the following figure that depicts how JMeter works −



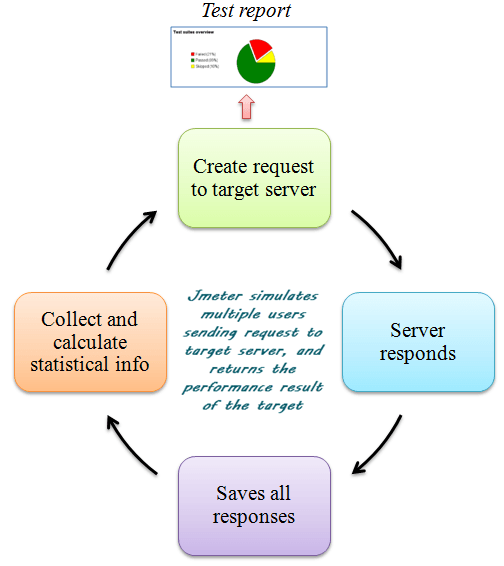
## Another Way to know how does JMeter work?

The basic workflow of JMeter as shown in the figures below

JMeter simulates a group of users sending requests to a target server, and return statistics information of target server through graphical diagrams

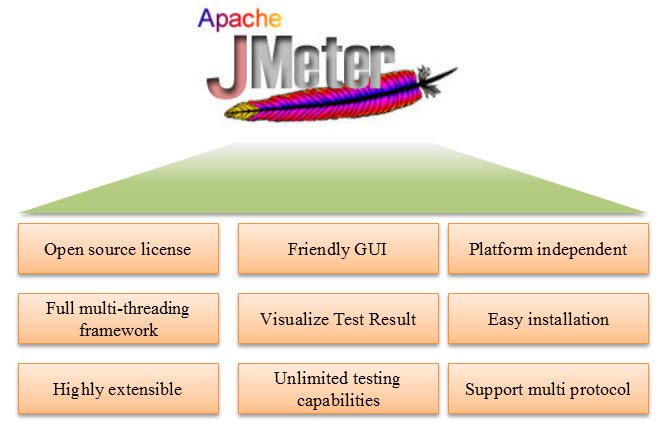


The completed workflow of JMeter as shown in the figure below



## JMeter Advantages

JMeter advantages are described as figure below



* **Open source license**: JMeter is totally free, allows developer use the source code for the development
* **Friendly GUI**: JMeter is extremely easy to use and doesn’t take time to get familiar with it
* **Platform independent**: JMeter is 100% pure Java desktop application. So it can run on multiple platforms
* **Full multithreading framework**. JMeter allows concurrent and simultaneous sampling of different functions by a separate thread group
* **Visualize Test Result:** Test result can be displayed in a different format such as chart, table, tree and log file
* **Easy installation**: You just copy and run the \*.bat file to run JMeter. No installation needed.
* **Highly Extensible**: You can write your own tests. JMeter also supports visualization plugins allow you to extend your testing
* **Multiple testing strategy**: JMeter supports many testing strategies such as [Load Testing](https://www.guru99.com/load-testing-tutorial.html), Distributed Testing, and[Functional Testing](https://www.guru99.com/functional-testing.html).
* **Simulation**: JMeter can simulate multiple users with concurrent threads, create a heavy load against web application under test
* **Support multi-protocol**: JMeter does not only support web application testing but also evaluate database server performance. All basic protocols such as HTTP, JDBC, LDAP, SOAP, JMS, and FTP are supported by JMeter
* **Record & Playback**– **Record** the user activity on the browser and simulate them in a web application using JMeter
* **Script Test**: Jmeter can be integrated with Bean Shell &[Selenium](https://www.guru99.com/selenium-tutorial.html)for automated testing.