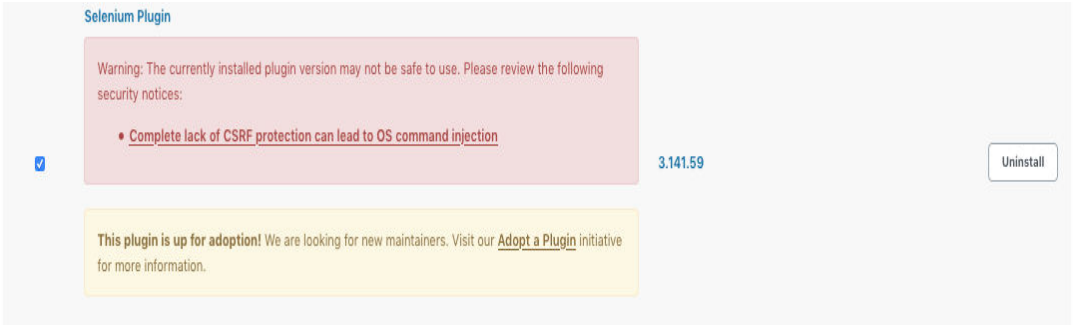
Selenium:

Selenium is a tool and a plugin used by many programming languages to interact with web interfaced apps. On Jenkins the Selenium plugin could be downloaded and referenced inside the invoked code.



In our example, Selenium is installed using the plugins. Selenium is then added to the dependency, which was a dependency after the Junit's dependency. The code that uses Selenium should import the libraries first by using the headers:

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.4.0</version>
</dependency>
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.By;
```

These headers get the abstract class Webdriver that is used to represent all web browsers, yet to deal with a specific web browser, the library should be specified as follows:

```
import org.openqa.selenium.chrome.ChromeDriver;
```

or

```
import org.openqa.selenium.firefox.*;
```

```
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.By;
import org.openqa.selenium.firefox.*;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
```

Once the libraries are imported, the code is to be written to satisfy the correct syntax and usage of the libraries and domain. First of all using the interface we instantiate the concrete class

```
WebDriver driver = new FirefoxDriver();
```

But to do that, we first have to mention the executable, or the file to be invoked for Firefox to open. This is done using a driver called: geckodriver. This is only in the case of Firefox; chromdriver is to be used instead for chrome. These driver files are to be installed according to the OS that will be hosting the web Brower.
Then the code should specify the path to this executable and type, as follows:

```
System.setProperty("webdriver.firefox.bin",
"Users/rabaa/Downloads/geckodriver");
System.setProperty("webdriver.gecko.driver",
"Users/rabaa/Downloads/geckodriver");
```

This is to be done before the actual test, meaning that the annotation is preferred to be @Before instead of including it as @Test. Then the code needed is followed according to the usage, for example:

```
driver.get("http://demo.guru99.com/test/guru99home/");
String title = driver.getTitle();
```

After the purpose of the driver is fulfilled, it is encouraged to close the driver using:

```
driver.quit();
```

This is also encouraged to be mentioned in the @after test annotation.