Different approaches to handle new window

Table of Contents:

**Introduction to Windowhandle**

**How Window handle works in handling multiple windows?**

**Sample code to handle multiple windows**

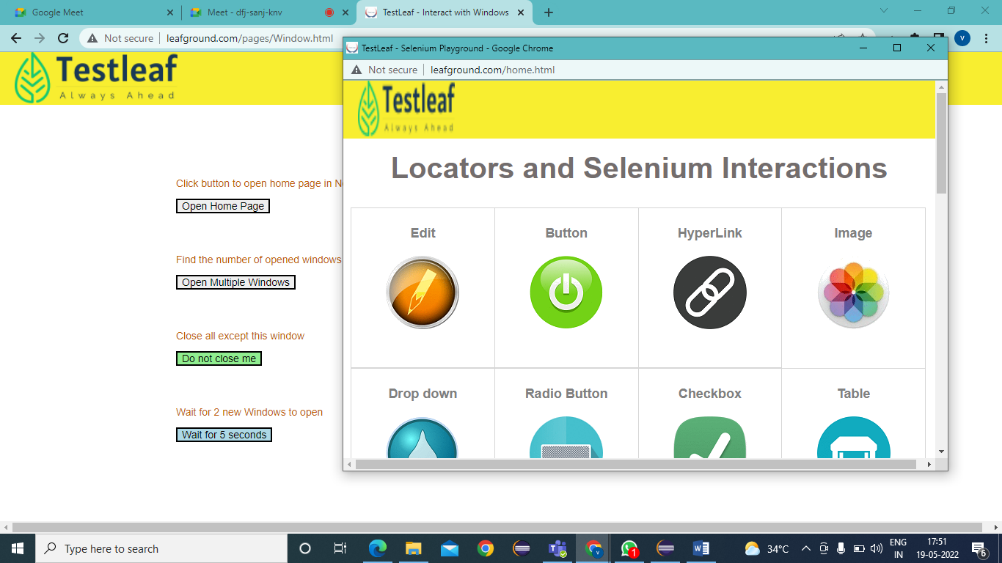
**Approaches to handle the multiple windows:**

A web application basically runs on web browser which are achieved through active driver interface. With this working architecture, web applications are automated easily with Selenium automation tool. Each applications are tested for better user interaction with the application. While automating a web page, there will be a lot of scenarios to be handled by the tester/developer. Here, we are going to know about- What are the different approaches to handle a new window and also how to switch between windows using selenium webdriver.

**Introduction to Windowhandle:**

Web applications are more dynamic in nature. Navigations are the major part here which assist the user to interact with the application. Browsing through the web apps are not restricted to access only the single page, it may leads to opening other second tabs or windows as per the user requirement in the application. In Selenium context, the so called second tab or windows are named as child window and the first loaded window is named as Parent Window.

Selenium helps to handle these multiple windows scenarios using **window handler**. Window handlers are nothing but it hold the address of the each browser windows or tab. In Selenium automation ,for each navigation to windows or tabs, a unique id(address) is created which is called as **Window handle**. It’s an API call that binds our webdriver with browser.

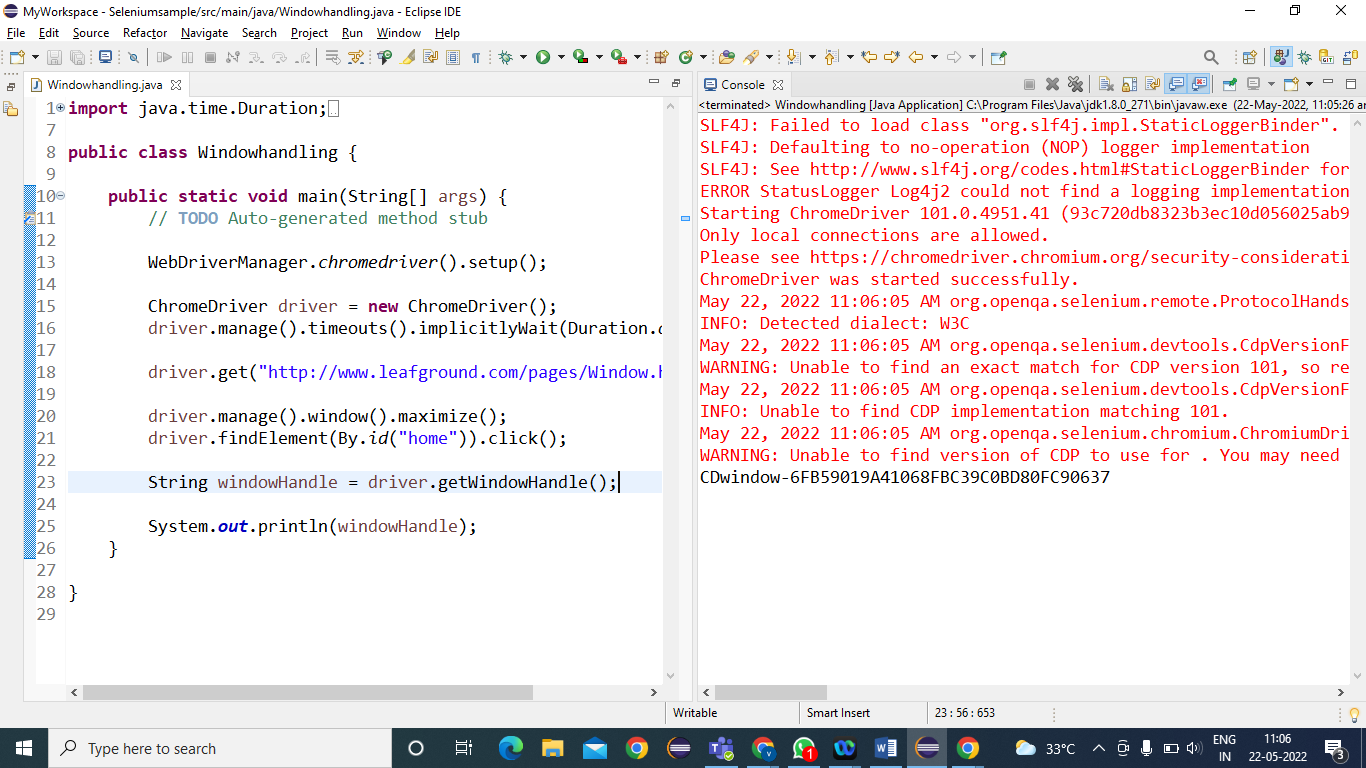
**How Window handle works in handling multiple windows?**

Window handle points to an active window, whenever we hit the button or link in a webpage. To understand this scenario using **LeafGround** application.

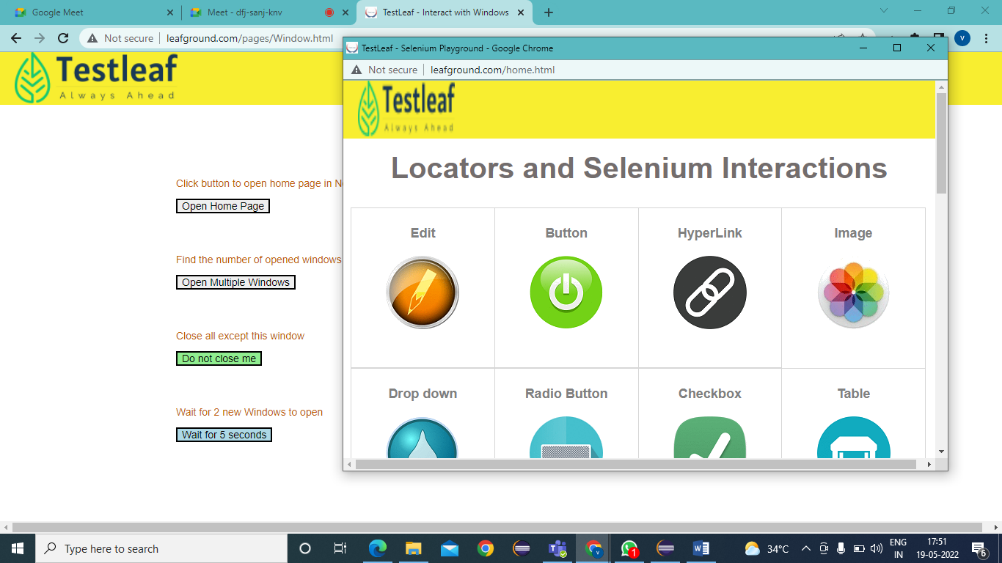
Once the button Open Home Page is clicked it will open a child window with the home page URL. While automating, the driver control will pointed to the main page. To access the second opened (child) window, we have to change the control to that window.

In this scenario, window handle come to play. Here, both the windows are treated with a unique address. Using that address, we can navigate between both the windows. To get the address of opened windows, the selenium provides two methods.

1. getWindowHandle() -> to get the current window id as a string value
2. getWindowHandles ()->returns the list of all the opened windows /tab as Set<string>

Let us look into the sample code for the LeafGround application.

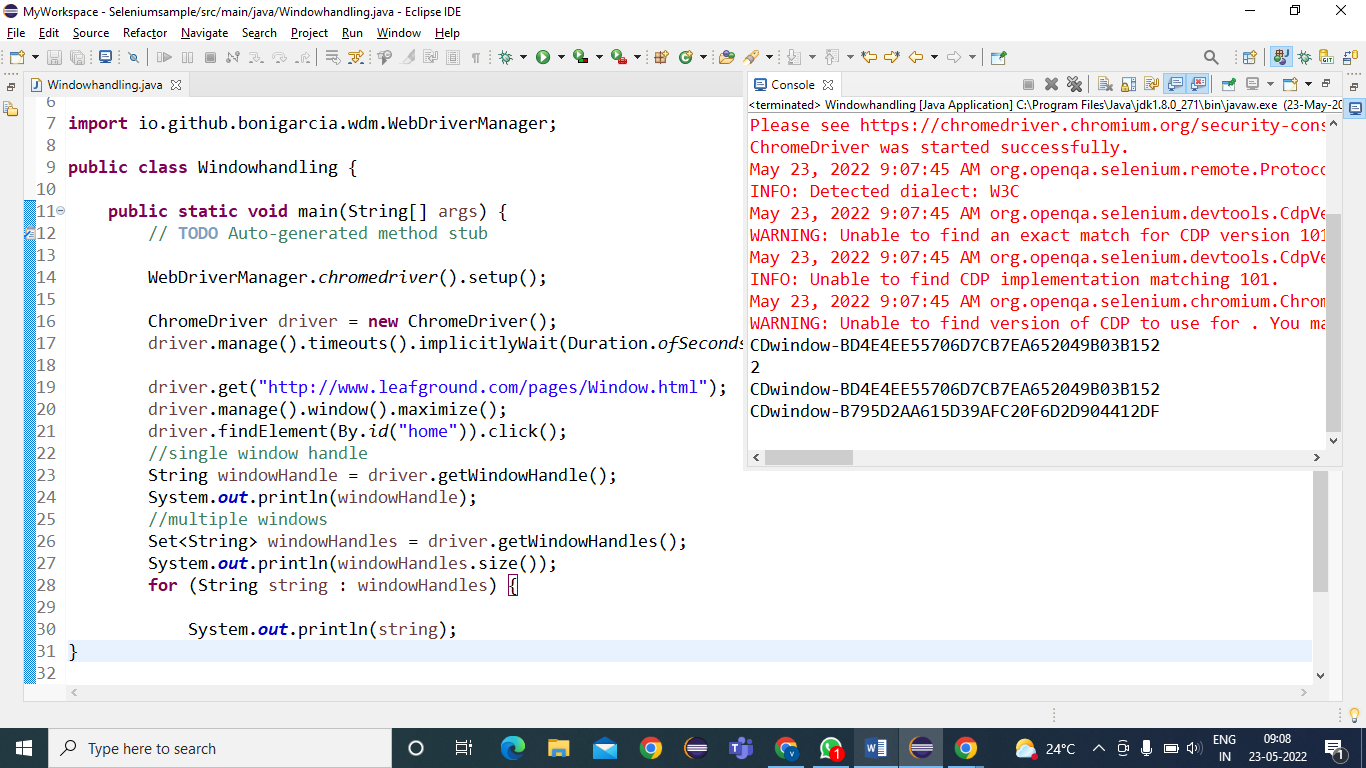
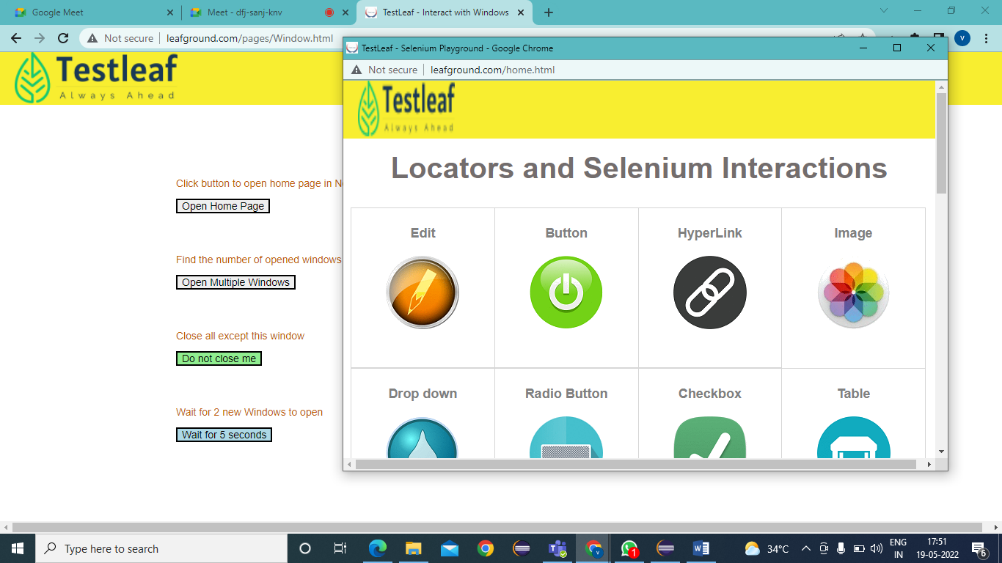
Windowhandle

****

The sample code is executed to get a single window handle where as to get the window handles of both the parent and child, the syntax should be getWindowHandles () ..

**Approaches to handle the multiple windows:**

Let’s look into the sample code to understand the concept of multiple window handles and the strategy to handle the switch between those window.

****

In the above , In console we can see the size as 2 which shows the available opened windows and the alphanumeric value

CDwindow-BD4E4EE55706D7CB7EA652049B03B152

CDwindow-B795D2AA615D39AFC20F6D2D904412DF

shows the unique handles of each window.

Now let’s write the code to switch to the second window. Selenium provides a method SwitchTo() to transfer the control from one window to other window. There are multiple strategy to handle multiple windows. Let’s code one by one,

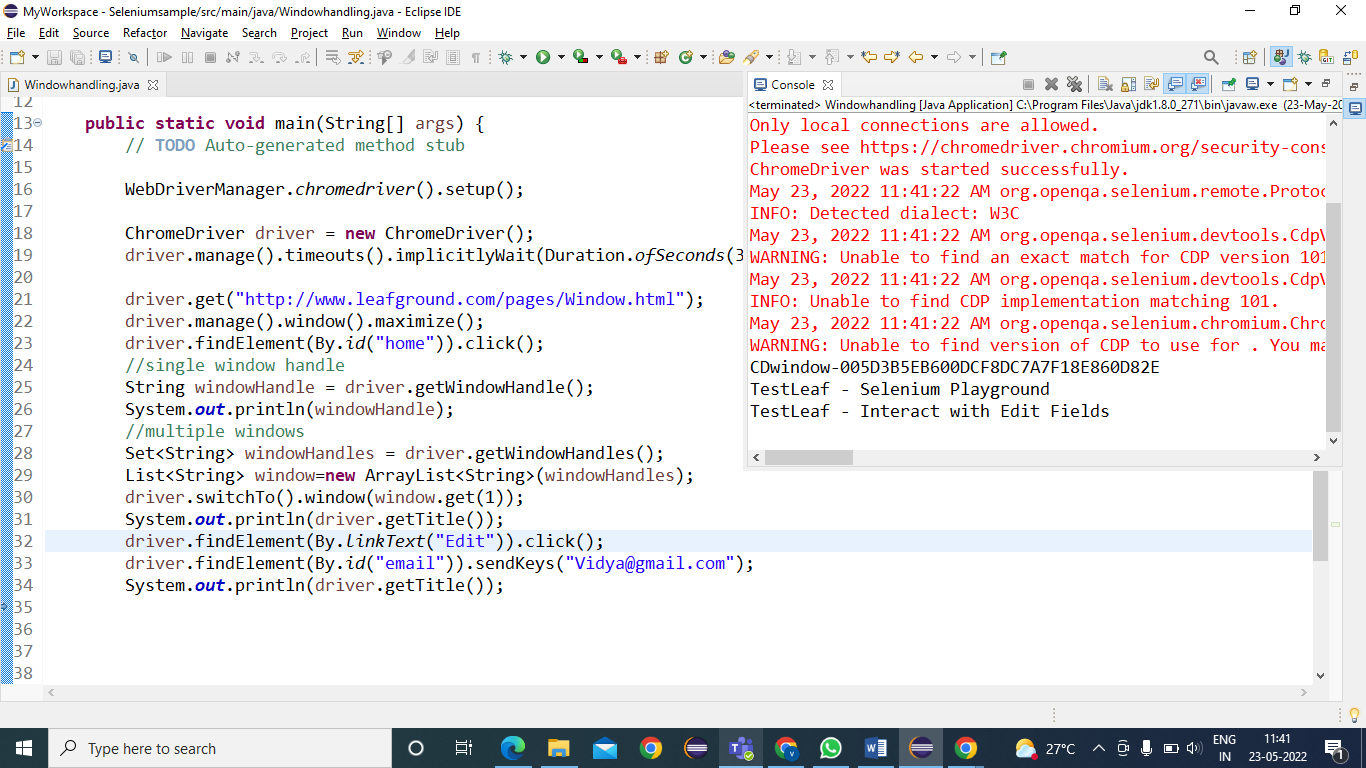
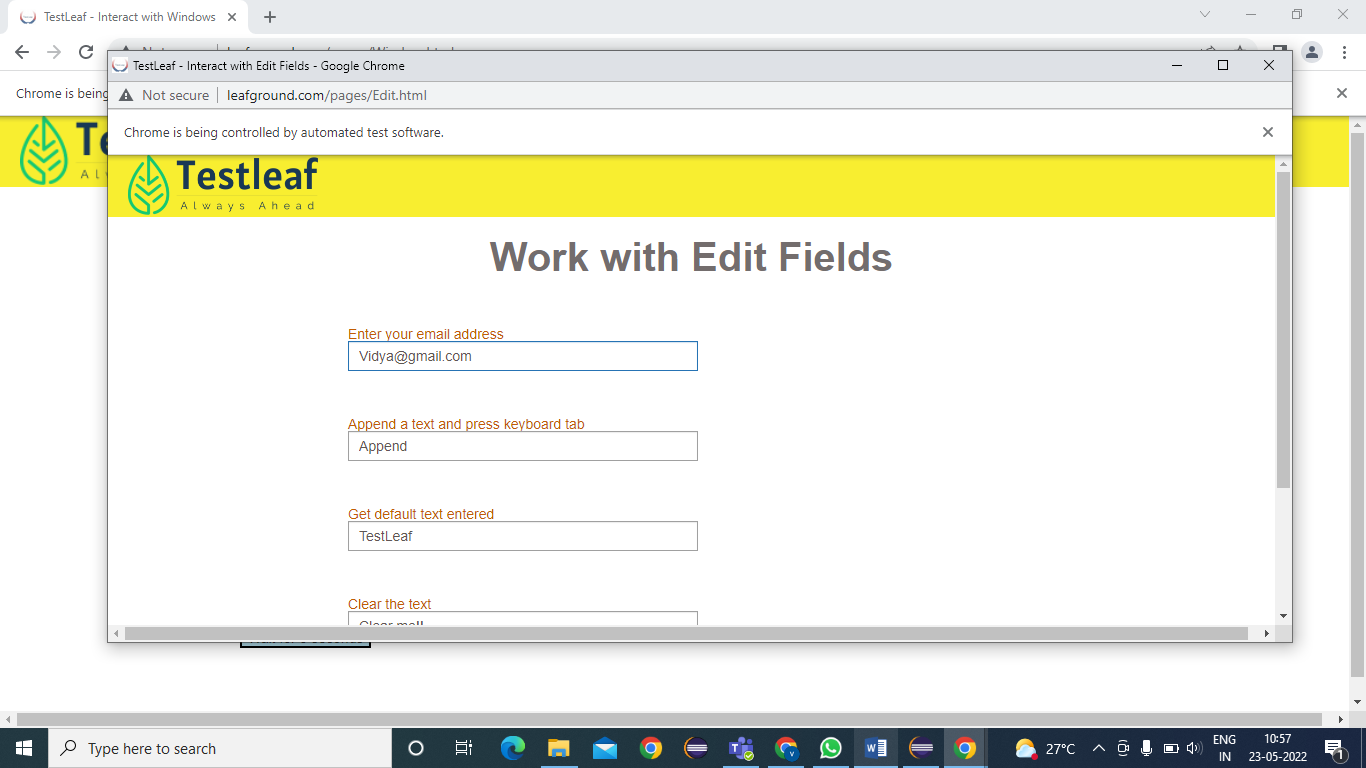
**Strategy: 1->accessing through index:**

The method getWindowHandles() returns the set of string values which denotes the window handle of each window. To access each window, the first approach to be followed is

\*Convert the Set to list  
 Set<String> windowhandles = driver.getWindowHandles();

List<String> window=new ArrayList<String>( windowhandles);

\*Using list.get(index)🡪

 driver.switchTo().window(window.get(1)) -🡪this will move the user control to first child Window.  
 driver.switchTo().window(window.get(0))🡪takes back to the parent window

**Strategy:2- using iterator**

Set<String> windowHandles = driver.getWindowHandles();

//using iterator

Iterator<String> I1 = windowHandles.iterator();

**while** (I1.hasNext()) {🡪increments the set

String childWindow = I1.next();

**if** (!windowHandle.equals(childWindow)) {

driver.switchTo().window(childWindow);🡪switch to child window using window name

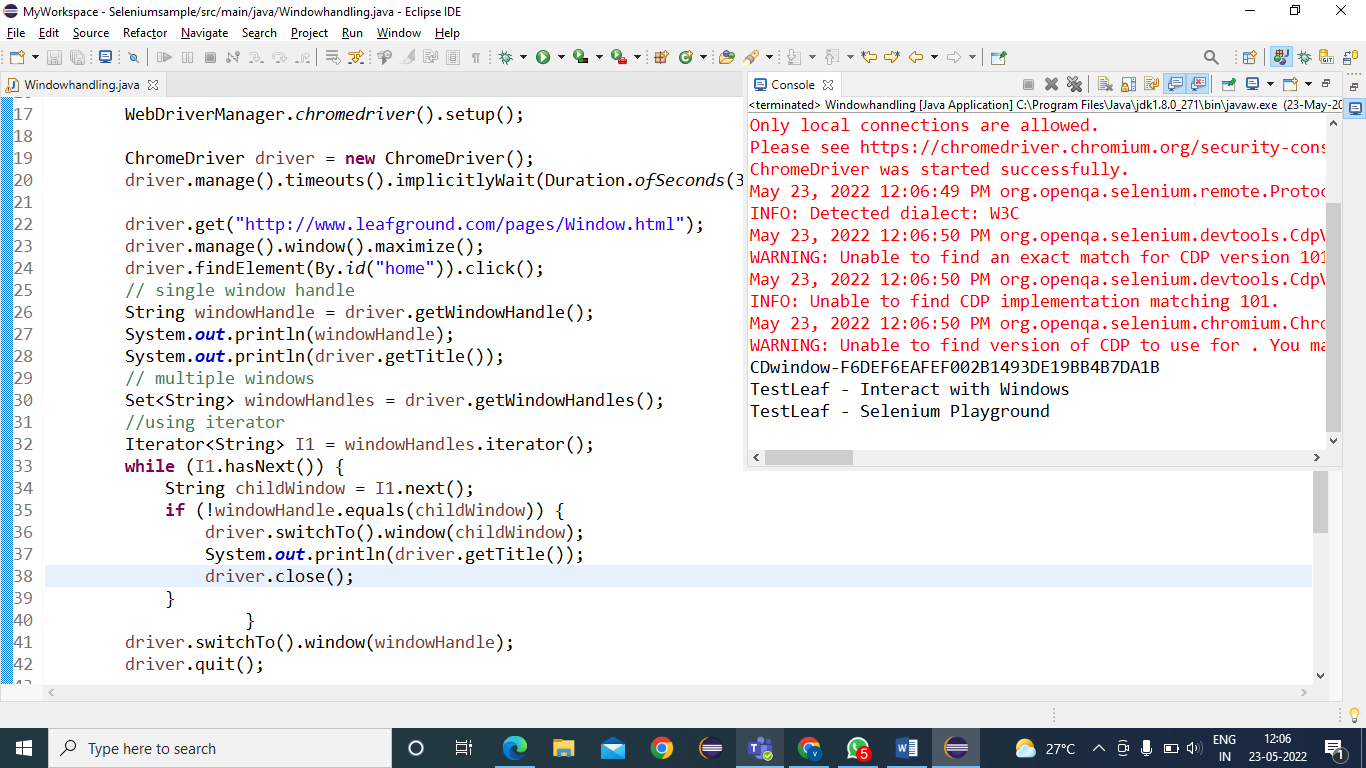
System.***out***.println(driver.getTitle());

driver.close();🡪closes the current active window

}

}

driver.switchTo().window(windowHandle);

**** driver.quit(); 🡪closes the all the automated opened window