SELENIUM

- Selenium is a tool
- It is a portable framework for testing web application
- It is an open-source tool for creating test suite
- Helps in testing web application across browsers & platforms
- It is not a single tool but suite of softwares

Benefits of Selenium

- Selenium is an open source and portable Web testing Framework
- supports various operating systems, browsers and programming languages
- supports parallel test execution that reduces time and increases the efficiency
- Selenium commands are categorized in terms of different classes

Selenium Limitations

- Selenium does not support desktop applications.
- Needed support from community forums to get the technical issues resolved
- should know at least one of the supported programming languages
- does not have any inbuilt reporting capability
- not possible to perform testing on images

Selenium WebDriver:

- ✓ test scripts can be developed using any of the supported programming languages and can be run directly in most modern web browsers.
- ✓ Languages supported by WebDriver include C#, Java, Perl, PHP, Python and Ruby
- ✓ Most of the commands used in Selenium WebDriver are simple & easy to implement
- ✓ Handles Navigation, Alerts / Dropdowns, Switching between windows, etc.

Finding Element:
It is important in a webpage; Each element has 3 important attributes: id, name & class
Will have to access HTML tags

Selenium WebDriver, uses a set of commands for performing different operations.
Commands are simply **methods** written in Java language

What should a script contain?

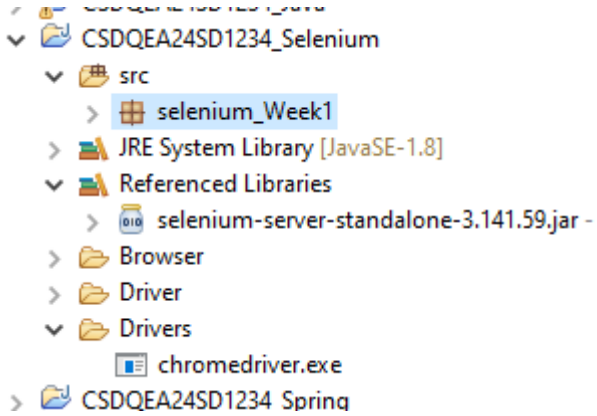
- Ensure relevant packages / classes are imported
- Invoke a browser
- Create instance of the Web Driver
- Maximize the screen window
- Give the URL of the webpage & launch the website
- Wait till the web page loads
- Find each Web Element you want to access
- Invoke an action on each of those Web Element
- Finally close the Web Driver

Locators:

1. Attribute
    a. Id

   b. Name
   c. Class
2. Text
   a. Link Text
   b. Partial Link Text
3. TagName
   a. TagName
4. Dynamic Locator - XPATH
   a. Absolute xpath
   b. Relative xpath

Folder Structure:



Example 1: Basics - Locators

```
package selenium_Week1;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Ex1_SeleniumBasics {

    public static void main(String[] args) {


    System.setProperty("webdriver.chrome.driver",
```

```
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");

    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

        System.out.println("Title of the Webpage : " + driver.getTitle());

//      WebElement email = driver.findElement(By.id("email"));
//      email.sendKeys("abc@gmail.com");
//
//      WebElement login = driver.findElement(By.name("login"));
//      login.click();

//      WebElement forgotPwd = driver.findElement(By.linkText("Forgotten password?"));
        WebElement forgotPwd = driver.findElement(By.partialLinkText("Forgotten"));
;
        forgotPwd.click();

        driver.close();
    }

}
```

Example 2: (Navigate WeSites in single tab of browser)

```
package selenium_Week1;
```

```
import java.util.concurrent.TimeUnit;

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

public class Ex2_NavWebSite {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

        System.out.println("Title of the Webpage :
" + driver.getTitle());
        System.out.println("Current URL : " +
driver.getCurrentUrl());

        Thread.sleep(3000);

    driver.navigate().to("https://www.google.com/")
;
        System.out.println("Title of the Webpage :
" + driver.getTitle());
        System.out.println("Current URL : " +
driver.getCurrentUrl());

        Thread.sleep(3000);
        driver.navigate().back();
```

```
        System.out.println("Current URL : " +
driver.getCurrentUrl());

        Thread.sleep(3000);
        driver.navigate().forward();
        System.out.println("Current URL : " +
driver.getCurrentUrl());

        Thread.sleep(3000);
        driver.navigate().refresh();
    }

}
```

Example 3: (working with Dropdown)

```
package selenium_Week1;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.support.ui.Select;

public class Ex3_UseList {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
```

```
    driver.get("https://www.roboform.com/filling-
test-all-fields");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

        WebElement month =
driver.findElement(By.name("66mm"));

//      Thread.sleep(3000);
//      Select selMonth = new Select(month);
//      selMonth.selectByIndex(2);
//
//      Thread.sleep(3000);
//      selMonth.selectByIndex(5);
//
//      Thread.sleep(3000);
//      selMonth.selectByVisibleText("Aug");

//      Thread.sleep(3000);
//      selMonth.selectByValue("11");
//
//      System.out.println("Selected value in list
is .. " +
selMonth.getFirstSelectedOption().getText());

        month.click();
        month.sendKeys("Jun");
        month.click();
    }

}
```

Exercises to work (in Eclipse):

Exercise 1:

Access all the elements in facebook page by finding element thru Id, Name, Class & text
Fill up valid details as required for each field, but don't submit;
Print tag name, some attribute value, etc for elements

Exercise 2:

Navigate front & back between different 3 different websites
Print title & URL of each navigation

Exercise 3:

Open "Facebook" from one of three browsers (Chrome / Firefox / IE).
You have to get browser name as input from user & had to open "Facebook" in that browser

Exercise 4:

https://www.roboform.com/filling-test-all-fields

a. Extract the default values from all the dropdown & print in console
b. Work with any list element; Pick a value, extract the text & print in console; do this few time & each time when you
select a value do it with index, value, visibletext, sendkeys, etc; Do this for both list with numeric & non-numeric values
c. Fill all the form elements & clear it finally

Exercise 5:

Go to this link: "https://demoqa.com/forms"
Access all the elements & fill up the form

Exercise 6:

Go to this link: https://demoqa.com/ and click on Elements.
Work on TextBox, CheckBox, RadioButton, Buttons & Links
Locate the elements & do necessary action on it
Extract values from TextBox, CheckBox & RadioButtons & print in console output

XPATH:

- It is a query language used for navigating thru XML documents to locate different elements
- It is a strategy to locate elements in Selenium
- It starts with double slash
- Uses single slash between each condition
- Uses double slash in between the xpath to by-pass parent or multiple child tags
- For attribute, use @
- While searching value, do remember that the value is case & space sensitive
- Use and, or operators to give multiple conditions

- Multiple conditions may use different attribute of same tag or same attribute with different values of a tag
- “Not” operator used find an attribute not existing in the tag
- Use “contains” and “text” functions
- Xpath index starts with 1

Syntax:

//TagName

//div

//TagName[index]

//div[2]

//TagName[@attribute]

//input[@name]

//TagName[@attribute=’AttributeValue’]

//input[@name='04lastname']

//TagName[text()]

//TagName[text()=’TextValue’]

//div[text()='Last Name']

// TagName[@attribute1=’AttributeValue’ and @attribute2=’AttributeValue’]

//input[@name='04fullname' and @type='text']

// TagName[@attribute=’AttributeValue1’ or @attribute=’AttributeValue2’]

//input[@name='04fullname' or @type='text']

//TagName[not(@attribute]

//input[not(@name)]

//TagName[not(@attribute=’AttributeValue’]

//input[not(@name='04fullname')]

//TagName[contains(@attribute, ‘AttributeValue’)]

//input[contains(@name,'04lastname')]

AXES:

- Use of relationship in xpath
- Child, parent, ancestor, descendant, preceding-sibling, following-sibling
- To refer relationship in xpath, use scope operator (::)

- While you traverse in a family tree, always travel from top to bottom
- Traverse to next nearest relationship

Syntax:

//TagName[relation::relativeTagName]

[xpath]/relation::relativeTagName

Example:

//input[@name='04lastname']/parent::div/parent::div/following-sibling::div/child::div[2]/child::input

//input[@name='04lastname']/ancestor::div/following-sibling::div/child::div[2]/child::input

Two types of XPATH:

1. Absolute XPATH
   One that is traversing from one tag to next closest tag
   Also traverses top to bottom via the tree structure
   Syntax:
   //Tagname1/TagName2/TagName3

   Example:
   //body/div[2]/form/div/div/div[4]/div[2]/input

2. Relative XPATH

   One that uses relationship

   Syntax:

   //TagName[relation::relativeTagName]

Example 1: (use of XPATH)

```
package selenium_Week2;

import java.util.concurrent.TimeUnit;

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

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

public class Ex1_useXPATH {

    public static void main(String[] args) {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

    driver.get("https://www.roboform.com/filling-
test-all-fields");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

//      WebElement lastName =
driver.findElement(By.xpath("//input[@name='04lastn
ame']"));
//      WebElement lastName =
driver.findElement(By.xpath("//body/div[2]/form/div
/div/div[4]/div[2]/input"));
        WebElement lastName =
driver.findElement(By.xpath("//input[contains(@name
,'04lastname')]"));
        lastName.sendKeys("Jacobs");

        WebElement fullName =
driver.findElement(By.xpath("//input[@name='04lastn
ame']/parent::div/parent::div/following-
sibling::div/child::div[2]/child::input"));
        fullName.sendKeys("John Jacobs");
    }
```

```
}
```

ALERTs & POPUPs

Three types of Web Page window formats:

1. Alert
2. Popup
3. Ad

Alerts:

- A small message box which appears on screen to give the user some information or notification
- It notifies the user with some specific warning or error
- At times it asks for permission to perform certain tasks

Three types of alerts:

a) Simple Alert – that displays some information & warning on the screen; will have one OK button
b) Confirmation Alert – that asks permission to do some type of operation; will have OK & Cancel button
c) Prompt Alert – that asks some input from the user and selenium webdriver can enter test using sendkeys; will have OK & Cancel button

Example for Handling Alert:

```
package selenium_Week2;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.Alert;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class Ex2_HandleAlert {
```

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


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

    driver.get("file:///E:/TrainerPlan/CorporateTra
iningMaterials/Batch35_CSDQEA24SD002-03-
04/JavaScriptExamples/Example2/input.html");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

        WebElement num =
driver.findElement(By.id("number"));
        num.sendKeys("6");

        WebElement value =
driver.findElement(By.xpath("//input[@value='Click'
]"));
        value.click();

        Thread.sleep(3000);
        Alert alert = driver.switchTo().alert();
        alert.accept();
    }

}
```

PopUps:

- It is a webpage driven content displayed in an extra window over the webpage
- Popups either partially or fully hide the content of web page
- Popup does not allow to operate on the webpage until popups are closed

WAITs:

- ✓ Most of web applications are developed using some Asysnchronous concepts
- ✓ When a page is loaded in browser, the elements may load at different time intervals
- ✓ Selenium script needs to be informed on these occasions, so script does not fail

Two types:

1. Implicit wait
2. Explicit wait

Implicit wait:

- ➢ It will tell the webdriver to wait for a certain amount of time before it throws an exception
- ➢ If the element is not located on the webpage within that time frame, it will throw the exception
- ➢ This wait is used for whole webpage to be loaded

Explicit wait:

- ➢ It will tell the webdriver to wait for certain conditions to happen or maximum time exceeded before throwing an exception
- ➢ It is a kind of intelligent wait that can be applied only on specified elements
- ➢ This gives better options during execution of script as it waits for dynamically loaded contents

Expected Conditions:

(i) alertIsPresent()
(ii) elementToBeClickable()
(iii) elementToBeSelected()
(iv) visibilityOfAllElements()
(v) visibilityOfElementLocated()
(vi) presenceOfElementLocated()
(vii) presenceOfAllElementsLocated()
(viii) textToBePresentInElement()

Example for Handling PopUp along with Explicit Wait:

```
package selenium_Week2;

import java.util.concurrent.TimeUnit;
```

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

public class Ex3_HandlePopup {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://scoop.eduncle.com/jee-
mains-preparation-tips");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

        Wait<WebDriver> winWait = new
WebDriverWait(driver, 20);
//
    winWait.until(ExpectedConditions.presenceOfAllE
lementsLocatedBy(By.id("entryPopupModal")));

    winWait.until(ExpectedConditions.presenceOfAllE
lementsLocatedBy(By.xpath("//div[@class='modal
entry-popup open']")));
```

```
        Thread.sleep(3000);
//      WebElement popUp =
driver.findElement(By.name("entryPopupButton"));
        WebElement popUp =
driver.findElement(By.xpath("//button[@class='bg-
close close-dismiss']"));
        popUp.click();
    }

}
```

JavaScript Executor

- An interface that provides a mechanism to execute JavaScript through Selenium Webdriver
- When selenium webdriver locators (like id, name, xpath, etc) do not locate an element or cannot do an operation, then JSExecutor performs that operation.
- No need an extra plugin or add-on, only to import respective package
- It provides two methods to run java script on the selected window
- JavaScript is the preferred language inside the browser to interact with HTML DOM
- Browser has JS implementation in it side & understand the JS commands quickly

1. executeScript
    a. It runs in the body of anonymous functions (a function without name)
    b. It allows to pass any data types including Boolean, long, string, etc as well as List, WebElement
    c. It blocks any other further action by browser
    d. Everything inside the script will be executed by browser & not server

2. executeAsyncScript
    a. It is one that also runs in body of anonymous function
    b. It helps to render the page quick & faster
    c. It does not block any other action by browser
    d. It helps to improve performance of the test

JS Executor functions:

Js.excuteScript(doc.getElementById.value)

.click()

.checked()

.hover()

.toString()

.scrollBy()

Example for JavaScript Executor:

```
package selenium_Week2;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.JavascriptExecutor;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class EX_JSExecutor {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
//      driver.get("https://www.facebook.com/");

    driver.get("https://www.roboform.com/filling-
test-all-fields");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

//      WebElement email =
driver.findElement(By.id("email"));
```

```
//		email.sendKeys("abcd@gmail.com");
//
//		Thread.sleep(3000);
//		WebElement login =
driver.findElement(By.name("login"));
//		JavascriptExecutor js =
(JavascriptExecutor)driver;
//		js.executeScript("arguments[0].click()",
login);
//
//		Thread.sleep(3000);
//		js.executeScript("alert('Welcome')");

		Thread.sleep(3000);
		JavascriptExecutor js =
(JavascriptExecutor)driver;
		js.executeScript("window.scrollBy(0,600)");

		WebElement comments =
driver.findElement(By.name("72__commnt"));
		comments.sendKeys("Welcome");

		Thread.sleep(3000);
		js.executeScript("window.scrollBy(0,800)");
	}

}
```

Example to handle multiple windows:

```
package selenium_Week2;

import java.util.Set;
import java.util.concurrent.TimeUnit;

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

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

public class Ex5_MultipleWindows {

    public static void main(String[] args) throws
InterruptedException {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

    driver.get("https://www.nopcommerce.com/en/show
case");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

        System.out.println("Current window title..
" + driver.getTitle());
        System.out.println("Focus is on window.. "
+ driver.getCurrentUrl());
        String  currWindow =
driver.getWindowHandle();

        WebElement image =
driver.findElement(By.xpath("//img[@title='Volvo
Cars Collection']"));
        image.click();

        Set<String> setAllWindows =
driver.getWindowHandles();
        System.out.println("Total windows opened
.." + setAllWindows.size());
```

```
        int count = 1;

        for(String window : setAllWindows){
            System.out.println("Window (" + count +
") is " + window);
            count++;

            if(currWindow.equals(window)){
                System.out.println("You are in
current window yet..");
            } else {
                driver.switchTo().window(window);
                System.out.println("New window
title.. " + driver.getTitle());
                Thread.sleep(3000);
                driver.close();
            }
        }

        driver.switchTo().window(currWindow);
        System.out.println("After switch back, prev
window title.. " + driver.getTitle());
//      System.out.println("Focus is on new
window.. " + driver.getCurrentUrl());

    }

}
```

```
Window (1) is C43E2BE30E4AD7C59437D3BC21841BE8
Window (2) is B76B72B63860C498AC760649FECB30AF


Window (1) is DDE3BAAA17154262E2C218FA33FACDA8
```

AJAX / Auto Complete:

- AJAX – stands for Asynchronous JavaScript & XML
- It is a technique used for creating fast & dynamic web pages
- This technique is asynchronous and uses a combination of JavaScript & XML
- It allows a web page to retrieve small amounts of data from the server or updates the part of web page without reloading the entire page
- Sometimes it may load in a second & sometimes it may take longer; we have no control over loading time
- AJAX sends HTTP requests from the client to server and then process the server's response without reloading the entire page
- We may not know when the AJAX call will get completed & the page has been updated
- Handling Auto Suggestion or Auto Complete in web based application is based on AJAX response
- Auto Complete will allow the browser to predict the response
- When a user starts to type in a field, the browser should display options to fill the field, based on browser history or previously typed value
- Auto Complete works with the objects like text box, search, url, email, date pickers etc.

Example for Auto-Suggestion / AutoComplete:

```
package selenium_Week2;

import java.util.List;
import java.util.concurrent.TimeUnit;

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

public class Ex6_AutoComplete {

    public static void main(String[] args) {


    System.setProperty("webdriver.chrome.driver",
```

```
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\Drivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

    driver.get("https://in.search.yahoo.com/?fr2=inr");

    driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);


    driver.findElement(By.id("yschsp")).sendKeys("Selenium");
        WebDriverWait wait = new WebDriverWait(driver, 20);


    wait.until(ExpectedConditions.visibilityOfAllElementsLocatedBy(By.xpath("//ul[@role='listbox']")));
        List<WebElement> list = driver.findElements(By.xpath("//ul[@role='listbox']/li"));

        System.out.println("Number of List values : " + list.size());

        for(int i=0; i<list.size(); i++){

    System.out.println(list.get(i).getText());

    if(list.get(i).getText().equals("selenium interview questions")){
                list.get(i).click();
                break;
            }
```

```
            }
        }
}
```

Actions:

- Perform UI actions thru the script
- Used for the user-facing API for emulating complex user gestures, instead of the Keyboard or Mouse actions directly.
- Implements the builder pattern; Builds a CompositeAction containing all actions specified by the method calls.
- perform() method called at the end of the method chain to actually perform the actions.
- Allows you to simulate user input events, such as mouse and keyboard actions.
- Clicking, double-clicking, hovering or other complex mouse actions can be scripted with an action
- Advanced user interactions such as holding a key while clicking something, or dragging and dropping an item are supported.
- These actions are performed by the Advanced User Interactions API, which consists of the Selenium Action class to perform these interactions.

Mouse Actions:

click() - clicks at the current mouse location
doubleClick() - double click at the current mouse location
contextClick() - right click at the current mouse location
dragAndDrop(WebElement source, WebElement target) - drags an element from one location to the other
moveToElement(WebElement target) - moves control to an element

Keyboard Actions:

keyUp(WebElement target, CharSequence key) - does a key release after focusing on the target element
keyDown(WebElement target, CharSequence key) - does a key press after focusing on the target element
sendKeys(WebElement target, CharSequence... keys) - types a sequence of keys.

Others:

Keys.Enter – to press “Enter” button on an object
getSize().height – to get the height of the object
getSize().width – to get the width of the object

Example for Actions & By class;

```
package selenium_Week2;

import java.util.concurrent.TimeUnit;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.interactions.Actions;

public class Ex7_Actions {

    public static void main(String[] args) {


    System.setProperty("webdriver.chrome.driver",
"E:\\Selenium\\Programs\\CSDQEA24SD1234_Selenium\\D
rivers\\chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.facebook.com/");

    driver.manage().timeouts().implicitlyWait(10,
TimeUnit.SECONDS);

        Actions faceAction = new Actions(driver);

        By byEmail = By.id("email");
        By byLogin = By.name("login");

        WebElement email =
driver.findElement(byEmail);

    faceAction.moveToElement(email).build().perform
();
        email.sendKeys("abc@gmail.com");
```

```
        WebElement login =
driver.findElement(byLogin);

    faceAction.moveToElement(login).build().perform
();
        login.click();
    }

}
```