CLASSNOTE - DAY 1
02/01/2022, Tuesday
HTML CLASS REVIEW:
- HTML : Hypertext markup language
- Is HTML a programming language?
  - No. It is a mark up language.
  - HTML does not have programming logic such as for loops, if conditions, variables.
- What makes a mark up language?
  - "MARK UP TAGS" create a mark up language.

- Is HTML the only mark up language?
  - No. We have more than 1 mark up languages.
  ex: XML : extensible markup language

- What is a mark up tag?
  - Markup tags act like containers.
  - It will change the behavior or display of the content passed into it.

- How many types of tags we have?
  - 2 types of tags.

#1- Paired tags :
  - Paireds tags have <openingTag> and </closingTag>.
  syntax: <openingTag> CONTENT </closingTag>.

  Ex: p, h, head, body, html, li, div, ul, u, strong, button, select, span, em,
  a

#2- Unpaired tags :

  - Unpaired tags have only
  - There is no closing tag.
  - That's why it is called self closing tag.

  syntax: </openingTag>
  ex: br, img, hr, input

What is an attribute:
  - Attributes provide additional information about specific web element.
  - Attributes go inside of the OPENING TAG ONLY.
  - If it is an unpaired tag/self closing tag it will go inside of the tag itself.
  - A web element can have as many attributes as needed. Number is not limited.

ex:
```
<div style="color:red;"> content </div>
```

```
<img src="cyberTruckimage.png">
```

PRACTICE:

```
<a href="https://www.etsy.com"> ETSY - SELL YOUR COOKIES </a>
```

#1- WHAT IS THE TEXT OF THIS TAG? WHAT IS THE CONTENT, WHAT IS DISPLAYED ON THE PAGE?

- "ETSY - SELL YOUR COOKIES"

#2- WHAT IS THE ATTIRBUTE OF THIS WEB ELEMENT?
- href is the attribute of this web element

#3- WHAT IS THE ATTRIBUTE VALUE OF HREF?
- https://www.etsy.com

#4- WHAT IS THE TYPE OF THIS TAG?
- Anchor tag, <a> tag, link tag

TELLING WHAT IS WHAT FROM THE COLORS:

- PURPLE	: TAG NAME
- ORANGE	: ATTRIBUTE NAME
- BLUE	: ATTRIBUTE VALUE
- BLACK	: TEXT (DISPLAYED ON THE PAGE)

- WHAT IS A WEB ELEMENT?
  - Every single thing we see on the page is a web element of its own.
  - Such as: links, buttons, input boxes, images, headers
- input tag:
  - input tags are used to let user enter some form of input.
  - the type of the input accepted from user is determined by the value we pass into "type" attribute.

  syntax:

```
<input type="text">
    --> to accept text from user
<input type="password">
    --> to accept text from user,
    --> the text will be hidded as password
<input type="button">
    --> to create button on the page
<input type="checkbox">
    --> to create checkbox on the page

<input type="radio">
```

--> to create radio button on the page

- select tag:
    - Allows us to create dropdowns in the HTML page.
    - By itself it is not enough. We have to use <option> tag to create different options displayed for user.

    syntax:
```
        <select>
            <options> TEXT 1 </options>
            <options> TEXT 2 </options>
            <options> TEXT 3 </options>
        </select>
```

- If you didn't take quiz, take it in the lunch break of after class. Today is the last day.
- Short videos explanation

- Maven project:
    - Maven is a "build" automation tool
    - "build" : repeating steps when we are creating a project
    - Maven project is different than Java project.
    - Maven is NOT a tool for testers.
    - It comes with certain type of folder structure and other files.

    - src
        - main : this is where developers write the source code of the webApp/webPage
        - test : this is where developers write their unit tests
    - pom.XML : this file is used to add and manage dependencies of our project

    - target : this file is where the maven project is storing the compiled version of the project and some other information.
    - it will only appear after you run your code.
    - everything in this folder will be refreshed every time we run our code.

Selenium methods:
    #1- What does the method do?
    #2- Does it accept any argument?
    #3- Does it have a return type?
    #4- Does it throw any certain exception?

- .get("");

#1- What does the method do?
    - It gets given URL in an opened browser.

#2- Does it accept any argument?

  - Yes it does.
  - It accepts a String argument.

#3- Does it have a return type?
  - Void return type.

- Basic navigations

  driver.navigate() --> navigate methods allows us to do simple navigations
  driver.navigate().forward () --> will take page to forward page
  driver.navigate().back() --> will take page to previous page
  driver.navigate().refresh()--> will refresh the current page
  driver.navigate().to() --> exactly same as .get() method

- .getTitle();

#1- What does the method do?
  - It gets the title of the page

#2- Does it accept any argument?
  - No. It does not accept any argument.

#3- Does it have a return type?
  - Yes. It returns a String.
  - It gets the title of the current page, and returns it as a String.

#4- Does it throw any certain exception?
  - No.

- .getCurrentUrl();

#1- What does the method do?
  - Gets the URL of the current page.

#2- Does it accept any argument?
  - No. It does not accept any argument.

#3- Does it have a return type?
  - Yes. It returns a String.
  - Gets the URL of the current page and return as String.

#4- Does it throw any certain exception?
  - No.

- driver.manage().window().maximize();
  - maximizes the currently opened browser

    - this effect (maximizing) will happen on the line we call this method
- driver.manage().window().fullscreen();
    - this one sometimes does not work for windows, so better not use it.

Why maximizing the page is important?
    - Because in different sizes the content displayed will be different.
    - This happens if the page is "responsive".
    - Responsive means : if size gets smaller, it will display less content.
        - Maybe it will store some web elements under some hamburger menu, or some other type of menu.

- driver.close();
    - will close only the currently opened browser
    - if we have more than one window open, it will only close the one that was focused on
- driver.quit();
    - it kills the current session.
    - it means if more than one window was opened, everything will be closed.
    - after using .quit() method, we cannot execute any more line of codes.
    - We will get "NoSuchSessionException"

What is a session?
    - Every time we run our Selenium code a session is created.
    - That specific session will continue until we explicitly kill it or code execution comes to end.

INTERVIEW QUESTION: Tell me about the common exceptions you are getting when you are using Selenium WebDriver?

```
========================
package com.cydeo.tests.day1_selenium_intro;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
public class BasicNavigations {
    public static void main(String[] args) throws InterruptedException {

 //1- Set up the "browser driver"
WebDriverManager.chromedriver().setup();
 //2-Create instance of the "Selenium WebDriver" Implementation will appear on the below note
 //The line below it will open my browser
//SessionID = 58DSAFASDASDF58, as user we see driver 4 selenium sees is only this
58DSAFASDASDF58 generated number
// selenium pass number around until the end when we use driver.quit the mentioned number is
deleted
```

```
        // after the number is deleted we cannot continue executed the code on the same browser
that is being executed by selenium
        // code need to be executed on the new browser
        WebDriver driver = new ChromeDriver();

        // place driver line after my ChromeDriver & it will maximize the browser size
        driver.manage().window().maximize();

        // this () below is almost the same to maximize but sometime don't work in windows
        // driver.manage().window().fullscreen();

        //3- Go to "https://www.tesla.com"
        driver.get("https://www.tesla.com");

        //Stop code execution for 3 seconds that's how we are going to be able to slow the
selenium down
        // by using Thread.sleep so when is get compiler error on sleep u need to handle the
exception by click on add exception to the method signature
        // Java run the code top 2 bottom & left 2 right so when it comes to ChromeDrive line it will
open an empty browser
        // so every time when u run specific code it generate specific id for that specific run time, it
opened a new browser.
        // get the title of the page & print URL after the titled
        String currentTitle = driver.getTitle();
        System.out.println("currentTitle = " + currentTitle);
        String currentUrl = driver.getCurrentUrl();
        System.out.println("currentURL = " + currentUrl);

        String currentURL = driver.getCurrentUrl();
        System.out.println("currentURL = " + currentURL);

        //Stop code for 3 seconds.
        Thread.sleep(3000);
        // use selenium to navigate back
        driver.navigate().back();

        //Stop code execution for 3 seconds
        Thread.sleep(3000);

        // use selenium to navigate forward
        //selenium session is created everytime when u run a selenium code
        driver.navigate().forward();

        // stop code execution for 3seconds in each step
        Thread.sleep(3000);

        // use selenium to navigate refresh
        driver.navigate().refresh();
        // stop code execution for 3seconds in each step
        Thread.sleep(3000);

        // use navigate().to(); I need to call my title again by rea-signed currentTitle equal
diver.getTitle
        //the Selenium check the title & return the value, if I am in GooglePage GetGoogleTitle,
return & print
        driver.navigate().to("https://www.google.com");

        // To get the two title I should print 2x the currentTitle after getting the Google page
        currentTitle = driver.getTitle();
        //System.out.println("driver.getTitle() = " + driver.getTitle());
        System.out.println("currentTitle = " + currentTitle);

        // Get the current URL using Selenium 4 me 2 get 2 different URL Tesla & Google URL
        //Summary try to get title & URL from 2 different pages  & You can inspect the web page to
see the changes
        // create one string call one method 2 print one value
        currentURL = driver.getCurrentUrl();

        System.out.println("currentURL = " + currentUrl);

        // To avoid get Google driver on my computer Dock after write ur code add driver.close(); 2
close current window
        driver.close();

        // this will close all of the opened windows
        driver.quit();
    }
}
======================
package com.cydeo.tests.day1_selenium_intro;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class SeleniumTest {
    public static void main(String[] args) {
        //1-Setting up the web driver manager once u typing to import type & press enter to import
        // 4 class purpose use chromedriver & web browser line bellow
        // We create our "Browser Driver".
        WebDriverManager.chromedriver().setup();

        //2-Create instance of the chrome driver object in line bellow
        WebDriver driver = new ChromeDriver();

        //3-Test if driver is working as expected, once you run the class it open the Google browser
        // on the computer & go straight to the web now ur programme is all set and ready to go
        // if you see the red message even see failed it is normal its just selenium default line that
is print on the console
        driver.get("https://www.facebook.com");
    }
}
====================
package com.cydeo.tests.day1_selenium_intro;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class Task1_YahooTitleVerification {
    public static void main(String[] args) {

        //TC #1: Yahoo Title Verification
        //1. Open Chrome browser.
        // do setup for browser driver
        WebDriverManager.chromedriver().setup();

        //1. Open Chrome browser
        WebDriver driver = new ChromeDriver();
        //2. Go to https://www.yahoo.com
        driver.get("https://www.yahoo.com");

        //3. Verification title:
        // Expected: Yahoo | Mail,Search,Politics,News,Finance,Sport & Videos
        String expectedTitle = "Yahoo | Mail,Search,Politics,News,Finance,Sport & Videos";

        //Expected: Yahoo actual title comes from the browser
        // We are going 2 create a simple if condition 2 do the verification2 check if
actualTitle.equals(expectedTitle
        // if passes
        String actualTitle = driver.getTitle();
        if (actualTitle.equals(expectedTitle)){
            // then print out
            System.out.println("Title is as expected. Verification PASSED");
        }else{
            System.out.println("Title is NOT as expected. Verification FAILED!");
        }
    }
}
===================
REVIEW WITH OSCAR——-<!DOCTYPE html>
<!-- #1. Version of the HTML and it has to go before everything-->
<html>
<!-- #2. Root of everything,
            Inside HTML tags: head and body
-->
<head>
            <!-- #3. Title, some links, stuff can not be seen on the page
            -->
            <meta charset="utf-8"> <!--meta tags are used for Meta Data-->
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <meta name="author" content="Oscar">
            <title>EU8 is the Best</title>
            <!--title : comes in the search engines, title is NOT visible inside the page
            getTitle(): to get with Selenium -->
</head>
<body>
<!-- #4. The main part, whatever we see on the page goes here
-->
<h1>What will be in this review Page?</h1>
<h2>What will be in this review Page?</h2>
<h3>What will be in this review Page?</h3>
<h4>What will be in this review Page?</h4>
<h5>What will be in this review Page?</h5>
<hr>  <!-- horizontal line-->
<br>  <!-- break -->
<ol>
            <li>What is HTML, CSS, Javascript?</li>
            <ul>
                        <li>HTML: HyperText Markup Language, it is not a programming language, no if
statements, no loops</li>
                        <li>CSS: Cascading style sheet, it used for existing web elements, it will not work
without HTML</li>
                        <li>JavaScript: JS gives actual dynamisim, functionality to HTML and CSS</li>
            </ul>
            <br>
            <img src="comparison.jpg"; height="300px">
            <li>What is WebElement, Tags, Attribute, Value?</li>
        <ul>
            <li>WebElements: Everything we see on a web page, links, pictures,
texts, inputBoxes, radio buttons, checkboxes</li>
            <li>Tags: They are like contaniers, whatever you put in them, they will be
displayed accordingly, We have paired and unpaired Tags. Paired ones come with opening and
closing tags. </li>
            <li>
                Attribute: gives extra information about the Tag
            </li>
            <li>
                Value: Value or content of the attribute
            </li>
            <hr>
            <img src="example.png" height="100px">
            <li>Paired Tag</li>
            <li>TagName: h1</li>
            <li>One Attribute: "style" </li>
            <li>Value of Style Attribute: "color:blue"</li>
            <li>Text of the webElement: My Header</li>
        </ul>
    <li>What are common tags that you will deal with as a Tester?</li>
</ol>
<div style="background-color: skyblue; color:darkred;"> <!--a container to style group of web
elements-->
    <h3>Why are we learning HTML?</h3>
    <p>
        Selenium library works on HTML, we need Selenium to automate HTML pages.
        We need to apply CSS to make our div element apart from other webelements in
the page. CSS goes to opening tag.
    </p>
    <a href="https://github.com/Cydeo-EU8/EU8HTMLReview">Click Me!</a>
    <!--
        anchor Tag: very important, we will use some special locator called
LinkText/Partial Link Text
        paired tags
        href: hypertext reference to make it functional
    -->
</div>
</body>
</html>
<!--
    1. Open Sublime Text, save your file as .html extension
What is HTML, CSS, Javascript?
What is WebElement, Tags, Attribute, Value
```

What are common tags that you will deal with as a Tester?
Why are we learning HTML?
Selenium library works on HTML, we need Selenium to automate HTML pages.
What is Front End?
What is Back End?
→
===
CLASSNOTES: DAY2—FEB 4TH, FRIDAY
TODAY'S SCHEDULE:
- REVIEW, - TASK SOLUTION, - findElement(), - Locators, - linkText,- partialLinkText
- name, - id,- className, - tagName, - getText(), - getAttribute()

- What is Selenium?
  - Basically Selenium is a bunch of jar files that allows us to automate browsers.
  - In simple terms, selenium is bunch of jar files (libraries)
  - We use these jar files to automate browsers
- What is Maven?
  - Maven is a build automation tool.
  - Maven projects come with certain folder structure and pom.xml file

- What is a build?
  - Build is repeating steps when creating the projects
  - Build also includes the repeating steps when managing the project as well.

```
  -src
     - main
     - test
  - pom.xml
```

- What is pom.xml file?
  - It is short for "project object model".
  - pom.xml file is the most important file in a maven project.
  - We manage (add, remove, change version of) our dependencies.

- Is Maven for tester?
  - No. Maven is a tool created for developers.
  - As testers (SDETs) we will take advantage of some of its functionality.

- What version of Selenium we are using?
  - 3.141.59
- What version of WebDriverManager we are using?
  - 5.0.3

- If the developers are using both main and test folders, which folder the testers use?

- Testers and developers are not working on the same project.
    - As SDET you will not have access to the source code.

- What is a dependency?
    - Dependencies are just jar files.
    - We add to pom.xml file and maven automatically downloads the version of the dependency we provide.
- Where do we get our expected data?
    - Requirement documents.
- Where do we get the actual data?
    - Comes from the browser. When we do our testing, we get actual data from browser.

- To be able to do any action on any web element, there are certain steps we have to follow.
    #1- We have to locate the web element we want to work on.
    #2- To locate we have to use a set of methods coming from Selenium library.
        - findElement()
        - locator methods
    #3- We decide what action we want to apply, find the method, and use it on the already located web element.
        - click

Selenium methods:
    #1- What does the method do?
    #2- Does it accept any argument?
    #3- Does it have a return type?
    #4- Does it throw any certain exception?

- .findElement(LOCATOR) method:
    1- This method finds and returns SINGLE web element.
    2- It accepts a locator argument, and finds the requested web element using locator.
    3- Return type: WebElement type (comes from Selenium library)
    4- Yes. It will throw "NoSuchElementException"
        - if you provide a wrong locator
        - if your page does not load as fast as it should

.sendKeys("string to be sent");
    - this method is used to send text using Selenium WebDriver
    - it accepts a string argument
    - takes the String, and passes it into the web element as a String.

- What is a locator?
    - Locators help us locate web elements using Selenium WebDriver.
    - Locators are basically 8 methods coming from Selenium library.

    - There are total of 8 locators.

- We will learn first 6 today.

#1- linkText:

- linkText locator will look through the TEXTS OF all of the links in the page, and return the matching link.
- It accepts a string as the text of the link we are trying to locate
- This locator will ONLY ONLY ONLY works on links. <a> tags only.
- This method looks for EXACT text match.
- It is similar to .equals method coming from JAVA.
- It will search through HTML code and return first matching result.

ex: <a href="https://something.com"> GOOGLE SEARCH </a>
syntax: driver.findElement(By.linkText("GOOGLE SEARCH"));

#2- partialLinkText:

- linkText locator will look through the TEXTS OF all of the links in the page, and return the PARTIAL matching link.
- It accepts a string as the text of the link we are trying to locate
- This locator will ONLY ONLY ONLY works on links. <a> tags only.
- This method looks for PARTIAL text match.
- It is similar to .CONTAINS method coming from JAVA.
- It will search through HTML code and return first matching result.

ex: <a href="https://something.com"> GOOGLE SEARCH </a>

syntax1: driver.findElement(By.partialLinkText("GOOGLE"));
syntax2: driver.findElement(By.partialLinkText("SEARCH"));
syntax3: driver.findElement(By.partialLinkText("GOOGLE SEARCH"));

All 3 syntaxes returns same web element.

#3 name:

- it uses "name" attribute's value to locate web elements.
- it will go through HTML code and find matching "name" attribute value, and return the first matching web element
- name does not have to be unique. So use whenever possible.

<div name="hu58"> </div>

syntax: driver.findElement(By.name("hu58"));

#4 id:

- it uses "id" attribute's value to locate web elements.
- it will go through HTML code and find matching "id" attribute value, and return the web element

- id HAS TO BE UNIQUE.
- id attribute's value will always be unique.

<div name="hu58" id="asdf57"> </div>

syntax: driver.findElement(By.id("asdf57"));

#5   className:
- it uses "class" attribute's value to locate web elements.
- it will go through HTML code and find matching "class" attribute value, and return the first matching web element.
- class attribute's value does not have to be unique.

<div name="hu58" id="asdf57" class="cde45"> </div>

syntax: driver.findElement(By.className("cde45"));

#6 tagName:
- it will find and return the first matching tag provided.
- this locator does not have many use cases unless either the page is very simple page, or we are trying to get a group of web elements using findElementS method

<div name="hu58" id="asdf57" class="cde45"> </div>

syntax: driver.findElement(By.tagName("div"));

- getText():
  - .getText() method will get the text from in between the opening tag and closing tag
  - it does not accept any argument
  - Return type: String

ex : <a href="https://something.com"> GOOGLE SEARCH </a>

driver.findElement(By.linkText("GOOGLE SEARCH"));
    -> returns me the link tag as a web element
    -> return type is WebElement.

driver.findElement(By.linkText("GOOGLE SEARCH")).getText();
    -> this line returns me a String

- getAttribute("attributeName");
  - .getAttribute() method will return the given attribute's value
  - it accepts a String argument.
  - return type: String

```
    - it will find the matching attribute, and return its value

    ex: <div name="hu58" id="asdf57" class="cde45"> TEXT </div>

    syntax: driver.findElement(By.id("asdf57")); --> this line returns the web element

- driver.findElement(By.id("asdf57")).getAttribute("class") --> cde45
- driver.findElement(By.id("asdf57")).getAttribute("name") --> hu58
=============
package com.cydeo.tests.day2_locators_getText_getAttribute;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;

public class T1_CydeoVerifications {
    public static void main(String[] args) {

        // TC #1: Cydeo practice tool verifications
        //1. Open Chrome browser
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();

        //2. Go to https://practice.cydeo.com
        // method we need to use to open our page is on the sentence below
        driver.navigate().to("https://practice.cydeo.com");

        //3. Verify URL contains, and we use our driver and getCurrent URL &  create a simple if
condition bellow
        //Expected: cydeo
        String expectedURL = "cydeo";
        String actualURL = driver.getCurrentUrl();

        if (actualURL.contains(expectedURL)) {
            System.out.println("URL verification PASSED");
        }else{
            System.out.println("URL Verification FAILED");
        }
         //4. Verify title:
         //Expected: Practice
         // To get my  actual title driver getTitle & driver will locate & find the specific element
         String expectedTitle = "Practice";
         String actualTitle = driver.getTitle();
        // next check is the actual titled is match expected title practice say equals(expectedTitle
        if (actualTitle.equals(expectedTitle)){
           System.out.println("Title verification PASSED");
        }else {
           System.out.println("Title Verification FAILED");
        }
         // I am comparing two different strings on this project
         // is very important to comment & explain what you are doing with the codes to
understand
        driver.close();

        // break 15mins: 10. 10am cst
        //break 15mins: 11.10 EST
         // To do any action on webElement we need to locate webElement 2 work on, then learn
some methods
         // from selenium livraria 1st findElement(), locator methods, so once u decide which
action
         // U need 2 find & apply the method on your located webElement.
    }
}
==================
package com.cydeo.tests.day2_locators_getText_getAttribute;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class T2_LinkTextPractice {
    public static void main(String[] args) {
      //TC #3: Back and forth navigation
      // 1- Open a Chrome browser, doing setUp & Maximizing my page
      WebDriverManager.chromedriver().setup();
      WebDriver driver = new ChromeDriver();
      driver.manage().window().maximize();

      //2- Go to: https://practice.cydeo.com
      driver.get("https://practice.cydeo.com");

      //3- Click to A/B Testing from top of the list.
      //Next we need 2 use findElement method code below doNot select elements() coz U
getDifferent result
      // The By.linkText() then go 2 ur code & click on ur link 2 inspect select A/BTesting, inspect
doubleClick A/BTesting text
```

```
    //We use findElement(),then apply linkText() 2 findLocation 2 GetText on the TestAutomation
webPage
    // the line below go to the HTML code find the link & return 2 us, so when I run create a
string
    //Thread.sleep(2000);
    //driver.findElement(By.linkText("A/B Testing")).click();
     WebElement abTestLink = driver.findElement(By.linkText("A/B Testing"));
     abTestLink.click();

      //4- Verify title is:
      //Expected: No A/B Test
      String expectedTitle = "No A/B Test";
      String actualTitle = driver.getTitle();

      if (actualTitle.equals(expectedTitle)){
         System.out.println("Title verification PASS!");
      }else {
         System.out.println("Title verification FAILED!!!");
         // on the above block we getTitle & reassigned to the user
      }

      //5- Go back to home page by using the .back();
      driver.navigate().back();
      //6- Verify title equals:
      //Expected: Practice
      expectedTitle= "Practice";
      actualTitle = driver.getTitle();

      if (actualTitle.equals(expectedTitle)){
         System.out.println("Title verification PASSED!");
      }else{
         System.out.println("Title verification FAILED!!!");
      }
      //BREAK UNTIL 1.01PM CST
      //BREAK UNTIL 2.01PM EST
   }
}
```
====================
```
package com.cydeo.tests.day2_locators_getText_getAttribute;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.Keys;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;
public class T3_GoogleSearch {
    public static void main(String[] args) {
        //TC#3: Google search
        //1- Open a Chrome browser
        WebDriverManager.chromedriver().setup(); // SIDE SUMMARIES set the browser
        WebDriver driver = new ChromeDriver(); //get browser
        driver.manage().window().maximize(); // maximise browser

        //2- Go to: https://google.com
        driver.get("https://www.google.com"); // get Goggle.com & next to write apple in the search
box

        //3- Write “apple” in search box
        //Expected: Title should start with “apple” word so for me to be able to generate some
actions
        //I need to locate the element first by our generate web & said find element by the letter q
        // Then it goes to the web element page check name & attribute find the 1st matching result
& return
        // even if we have multi locator by always find the 1st matching element & returnBack to us
        //4- PRESS ENTER using Keys.ENTER
        //to locate WebElement on googleSearchBox = driver.findElement(By.name("q"));. Then
find attribute value q
        // Q was found googlePage we use googleSendKeys then enter apple, NEXT command
Keys.ENTER) is to...
        //imitate user press enter
        WebElement googleSearchBox = driver.findElement(By.name("q"));//2 write webElement in
searchBox,WeNeed to locate
        googleSearchBox.sendKeys("apple" + Keys.ENTER);//the code type apple then press
ENTER

        //5- Verify title:
        //Expected:Title should start with "apple" word 4 checking the title
        // 1st verify the title otherwise u cannot move forward with ur project
        String expectedInTitle = "apple";
        String actualTitle = driver.getTitle();//string is 2 get the title

        if (actualTitle.startsWith(expectedInTitle)){
            System.out.println("Title verification PASSED!");
        }else{
            System.out.println("Title verification FAILED!!!");
        }
    }
}
```

```
=====================
package com.cydeo.tests.day2_locators_getText_getAttribute;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class T4_LibraryLoginPage {

    public static void main(String[] args) {
        //TC #4: Library verifications
        //1. Open Chrome browser
        WebDriverManager.chromedriver().setup();
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        //2. Go to https://library2.cybertekschool.com/login.html
        driver.get("https://library2.cybertekschool.com/login.html");

        //3. Enter username: “incorrect@email.com”
        // The webElement on your inspect from the web that accept input from the user are?
INPUT TAG
        //From the WebElement inspect page that accept value from the user both email &
password
        //They both have type,ID & class attribute & to log in on libraryWebpage, using
"form-control"input 4 the email
        //It's find & return 2 me the 1st matching one result,I need to add (By.className plus
("form-control"));
        // to locate the username. Next type usernameInput.sendKeys("incorrect@email.com");
        WebElement usernameInput = driver.findElement(By.className("form-control"));
        usernameInput.sendKeys("incorrect@email.com");

        //4. Enter password: “incorrect password”
        //Use "inputPassword" for the password
        //Locate input if u see the label, go straight 2 input box, find ID & copy paste after
(By.id("inputPassword"))
        //To do the negative test adding the passwordInput.sendKeys("incorrect password");
        WebElement passwordInput = driver.findElement(By.id("inputPassword"));
        passwordInput.sendKeys("incorrect password");

        //5. Click to Sign in button
        WebElement signInButton = driver.findElement(By.tagName("button"));
        signInButton.click();
        //6. Verify: visually “Sorry, Wrong Email or Password”
        //displayed.
    }
}
=================
package com.cydeo.tests.day2_locators_getText_getAttribute;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class T5_getText_getAttribute {
    public static void main(String[] args) {

        //TC #5: getText() and getAttribute() method practice
        //1- Open a Chrome browser.
        WebDriverManager.chromedriver().setup(); //set browser driver
        WebDriver driver = new ChromeDriver(); //get new browser
        driver.manage().window().maximize(); //maximizing
        //2- Go to: https://practice.cydeo.com/registration_form
        driver.get("https://practice.cydeo.com/registration_form"); //get ourPage

        //3- Verify header text is as expected:
        //If U don't C any attribute,use the TAG name locator ("h2")) to locate our header add
(By.tagName("h2"))
        //Since return the webElement ("h2")) I need to  store inside the WebElement not a String
        //First We need to locate the WebElement more details on the 1st link below
        WebElement headerText = driver.findElement(By.tagName("h2"));//located ourHeader &
storedInside HeaderText below
        //Expected: Registration form from the Google test WebSite. Once U click on your
"Registration form" header
        //Once locate U can add String to your header
        String expectedHeaderText = "Registration form"; //Create Expected HeaderText
        String actualHeaderText = headerText.getText(); //We use getText, 2 find & return
"Registration form" as String

        if (actualHeaderText.equals(expectedHeaderText)){ //Do verification & print
            System.out.println("Header text verification PASSED!");
        }else{
            System.out.println("Header text verification FAILED!!!");
        }

        //4- Locate “First name” input box
        // we are locating the web element using "name" locator
        // name attribute has "firstname" value
        //WebElement firstNameInput = driver.findElement I use className to find the attribute to
locate WebElement
        WebElement firstNameInput = driver.findElement(By.className("form-control")); //Located
the webElement

        //5- Verify placeholder attribute's value is as expected:
        // Expected: first name from our GoogleWebPage Registration form 2 verify placeholder
attribute value below
        //GetPlaceHolder store insideNewString
        String expectedPlaceHolder = "first name";
        String actualPlaceHolder = firstNameInput.getAttribute("placeholder");//Get the Attribute
Value

        if (actualPlaceHolder.equals(expectedPlaceHolder)){//Do verification & Print
            System.out.println("Placeholder text verification PASSED!");
        }else{
            System.out.println("Placeholder text verification FAILED!!!");
        }
        driver.close();
    }
}
```
-===

CLASSNOTES: DAY3

Today's schedule: #1- Review, #2- Task1,#3- Create utility class, #4- Task2
   #5- .isDisplayed(), #6- cssSelector, #7- xpath

- What is maven?
- Maven is build automation tool.
- What is build?
- Repeated steps when creating the project and also managing the project.
- such as; creating the folder structure, adding dependencies, managing dependencies, compiling, deploying...

- Is maven for testers? Was it created for testers?
- No. It is a developer tool. We use it to take advantage of some of its functionalities.

- Is maven the only "build automation tool" out there?
- No. Ant, Gradle

- .findElement(By.locator("STRING"))

- What does it do?
- Finds and returns a single web element’

- What does it accept as argument?
- It accepts locator parameter as String.

- What is the return type?
- WebElement type: comes from Selenium library.

- What kind of exception does it throw?
- It will throw NoSuchElementException if it cannot somehow find given web element.

- What happens when exception is thrown in Selenium code execution? Is it going to execute the rest of the lines?
- Once it throws the exception, the rest of the lines will not be executed UNLESS we handle it.

- When do we have NoSuchElementException?
#1- Wrong locator provided
#2- Synchronization: when browser driver and browser are not on the same page, driver will try to find a web element that is not loaded yet. If this happens, it will throw NoSuchElementException.

- .click();
- What does it do?
- It clicks to the given web element.
- Does it accept argument?
- No.

- Does it work just by itself --> driver.click();
- driver.findElement(locator).click();

- .sendKeys();
- What does it do?
- It will pass the provided string into given WebElement.

- Does it accept argument?
- Yes. It accepts String argument.
- We can pass Keys. commands into sendKeys() method as well.

- Keys.ENTER:
- This piece of code will imitate user pressing ENTER from keyboard.

LOCATORS:
- How many total locators we have?

- Selenium has total of 8 locators.

- id
- name
- className
- linkText
- partialLinkText
- tagName

- cssSelector
- xpath

<a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>

1- id:
- It will find and return the web element that has matching id attribute value
- "id" is always unique.

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>

syntax: driver.findElement(By.id("bb22"))

2- name:
- It will find and return the web element that has matching name attribute value
- "name" is NOT always unique

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>
syntax: driver.findElement(By.name("uh68"))

3- className:
- It will find and return the web element that has matching class attribute value
- "class" is NOT always unique

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>
syntax: driver.findElement(By.className("ff58"))

4- linkText:
- It will look through all of the LINKS in the HTML page and return FIRST matching result.
- It ONLY ONLY ONLY works with links (<a> tags)
- It works similar to .equals method in java.
- It is looking for exact String match.

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>
syntax: driver.findElement(By.linkText("TESLA CYBERTRUCK"));

5- partialLinkText
- It will look through all of the LINKS in the HTML page and return FIRST matching result.
- It ONLY ONLY ONLY works with links (<a> tags)
- It works similar to .contains method in java.
- It is looking for PARTIAL String match.

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>
syntax1: driver.findElement(By.partialLinkText("TESLA CYBERTRUCK"));
syntax2: driver.findElement(By.partialLinkText("CYBERTRUCK"));
syntax3: driver.findElement(By.partialLinkText("TESLA"));
syntax4: driver.findElement(By.partialLinkText("ESLA"));

6- tagName:
- This locator locates using given TAG NAME.
- We literally pass the TAG name as String parameter.
- It will find and return first matching result.

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>
syntax1: driver.findElement(By.tagName("a"));

- What happens if we provide name,class attribute value and there is more than 1 result?
- It will find and return the first matching result.

- .getText();
- What does it do?
- It will get the content from in between the opening tag and closing tag

- What is the return type?
- String
- Does it accept any argument?
- No.

syntax: We cannot say driver.getText();
- driver.findElement(locator).getText(); --> it will return the text of given web element

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>

syntax: driver.findElement(By.name("uh68")).getText() --> TESLA CYBERTRUCK

- .getAttribute();
- What does it do?
- It will accept an attribute and return its value.
- It accepts a String argument
- Return type is String

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>

```
syntax: driver.findElement(By.name("uh68")).getAttribute("href")     --> https://tesla.com
syntax: driver.findElement(By.name("uh68")).getAttribute("name")     --> uh68
syntax: driver.findElement(By.name("uh68")).getAttribute("class")    --> ff58
syntax: driver.findElement(By.name("uh68")).getAttribute("id")       --> bb22
```

- .isDisplayed():
- What does it do?
- It returns boolean value on a given web element.
- If web element is displayed, it will return "true"
- If web element is not displayed, it will return "false"
- It does not accept any argument.
- syntax: driver.findElement(locator).isDisplayed(); --> true, if displayed
- syntax: driver.findElement(locator).isDisplayed(); --> false, if not displayed

----
- cssSelector locator:
  - cssSelector is one of 8 locators from Selenium library.
  - cssSelector allows us to create custom locators.
  - we are able to locate web elements with any attribute value.
  - we are not just limited to name, id, className
  - we can use any custom attribute we see on a web element

  syntax#1: tagName[attribute='value']
  syntax#1: tagName[attribute="value"]

ex: <label class="login-item-checkbox-label" for="USER_REMEMBER">Remember me on this computer</label>

cssSelector practice #1- Locate "label" using "class" attribute

  tagName[attribute='value']

  label[class='login-item-checkbox-label'] cssSelector practice #2- Locate "label" using "for" attribute

    tagName[attribute='value']

    label[for='USER_REMEMBER']

ex#2: ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>

#1- locate above link using cssSelector, using name:

    tagName[attribute='value']
    a[name='uh68']

#2- locate above link using cssSelector, using id:

    a[id='bb22']

#3- locate above link using cssSelector, using class:

    a[class='ff58']

#4- locate above link using cssSelector, using href:

    a[href='https://tesla.com']

- cssSelector has another syntax, but it works with id and class attributes only.

- . stands for class attribute
- # stands for id attribute

syntax#2:

    tagName.classValue
    tagName#idValue

ex#3: ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>

    tagName.classValue ---> a.ff58

    tagName#idValue --> a#bb22

- XPATH LOCATORS
    - xpath is one of 8 locators of Selenium
    - xpath allows us to create custom locators using provided attributes and their values
    - we can also use the text of the provided web element to create locators

    - XPATH has 2 different types

Interview question: What is the difference between absolute xpath and relative xpath?

#1- ABSOLUTE XPATH:
    - Absolute xpath starts with single slash "/"
    - It starts looking in html from the root/parent element : html element
    - It starts from html tag, and it goes down 1 by 1 until we reach to the web element we are looking for
    - This is not good way of locating a web element.
    - It will break with any minimal change in the html code.

```
/html/body/table/tbody/tr[2]/td/div/div/form/div[4]/button
```

#2- RELATIVE XPATH
    - Relative xpath starts with double slash "//"
    - "//" means you can start from anywhere in the HTML code
    - Since we are allowed to start from anywhere in the HTML code, relative xpath is very dependable
    - We will use relative xpath, not absolute xpath
    - The only time your relative xpath is breaking (not working) is when/if the developer is specifically changing the attribute value we used

--> MAIN SYNTAX: //tagName[@attribute='value']

ex: <a href="https://tesla.com" name="uh68" class="ff58" id="bb22"> TESLA CYBERTRUCK </a>
-->   //tagName[@attribute='value']
- locate above <a> tag using relative xpath locator with different attributes.

    - using name attribute   : //a[@name='uh68']
    - using class attribute   : //a[@class='ff58']
    - using id attribute   : //a[@id='bb22']

COMMONLY USED XPATH SYNTAXES:

#1- //tagName[@attribute='value']
#2- //tagName[contains(@attribute, 'value')]
#3- //tagName[.='text'] same as //tagName[text()='text']

#4- //*[@attribute='value']

EXPLANATIONS:
#1- //tagName[@attribute='value']

We are saying, get me the given tag with provided attribute and value

#2- //tagName[contains(@attribute, 'value')]

We are saying, get me the given tag that has the attribute which contains the value in the locator
Looks for the tagName that has matching or containing attribute value

#3- //tagName[.='text']

This locator will return the web element with given text

#4- //*[@attribute='value']

We are saying, we do not care about which tagName, return us the web element with matching attribute and value result
chro path'
chrome developer tool : right click > copy > copy selector --> cssSelector
chrome developer tool : right click > copy > copy xpath --> relative xpath
chrome developer tool : right click > copy > copy full xpath --> absolute xpath
- I do not suggest using any tools unless for just seeing example purpose.
======================

```
 package com.cydeo.tests.day3_CssSellector_xpath;
import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.chrome.ChromeDriver;

public class T1_locators_getText {

   public static void main(String[] args) {
      //TC #1: NextBaseCRM, locators and getText() practice
      //1- Open a Chrome browser
      //WebDriverManager.chromedriver().setup();
      //WebDriver driver = new ChromeDriver();
      WebDriverManager.chromedriver().setup();//Set up browser
      WebDriver driver = new ChromeDriver();//get ChromeBrowser
      driver.manage().window().maximize();//maximize browser
```

```
        //2- Go to: https://login1.nextbasecrm.com/
        driver.get("https://login1.nextbasecrm.com/");

        //Next 2 locate WebElement we can say driver findElement 4 the 1st we can say
className below
        //3- Enter incorrect username: “incorrect”
        //driver.findElement(By.className("login-inp")).sendKeys("incorrect");
        WebElement inputUsername = driver.findElement(By.className("login-inp"));
        WebElement user_password = driver.findElement(By.name("USER_PASSWORD"));
        inputUsername.sendKeys("incorrect"); //U can say inputUsername.sendKeys "incorrect"

        //4- Enter incorrect password: “incorrect”
        //To auto generate local variable, once U get "USER_PASSWORD" from our Chrome
WebPage, I want 2 use my name
        //once U get userPassword U can say river.findElement get the name & pass
USER_PASSWORD 2 create variableBelow
        //When U use ShortCut will automaticGenerate WebElement & guess WhateverLine & type
of variable U try 2 return.
        //Mac: option + enter
        //Windows: alt + enter
        WebElement inputPassword= driver.findElement(By.name("USER_PASSWORD"));
        inputPassword.sendKeys("incorrect");

        //5- Click to log in button.
        WebElement loginButton =
driver.findElement(By.className("login-btn"));//locateLoginButton on WebPage
        loginButton.click();

        //6- Verify error message text is as expected:
        //Expected: Incorrect login or password
        //Need to locate ErrorMessage on my workingChromeWebPage inspect & locate the
errortext
        //The LineBelow I need to use errorMessage coz I got my actualValueFromTheBrowser &
errorMessage is WebElement
        //We store errorMessageInsideWebElement so I create a string & try 2 store string inside
the WebElement
        //I can say getText  2 GetWebElement&ReturnAs a String
        WebElement errorMessage = driver.findElement(By.className("errortext"));

        String expectedErrorMessage = "Incorrect login or password";
        String actualErrorMessage = errorMessage.getText();//create a string try 2 store
WebElementInsideTheString
```

```
        if (actualErrorMessage.equals(expectedErrorMessage)){
            System.out.println("Error message verification PASSED!");
        }else{
            System.out.println("Error message verification FAILED!!!");
        }
    }
}
====================
package com.cydeo.tests.day3_CssSellector_xpath;
import com.cydeo.tests.utilities.WebDriverFactory;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
public class T2_getText_getAttribute {
    public static void main(String[] args) {
        //TC #2: NextBaseCRM, locators, getText(), getAttribute() practice
        //1- Open a Chrome browser
        //U must do manual testing 1st 2 make sure everything is fine 4 U to be able to automate ur
Test On the WebPage
        WebDriver driver = WebDriverFactory.getDriver("chrome");
        driver.manage().window().maximize();

        //2- Go to: https://login1.nextbasecrm.com/
        driver.get("https://login1.nextbasecrm.com/");

        //3- Verify “remember me” label text is as expected:
        //Expected: Remember me on this computer
        WebElement rememberMeLabel =
driver.findElement(By.className("login-item-checkbox-label"));

        String expectedRememberMeLabel= "Remember me on this computer";//To VerifyApply
regular label verification
        String actualRememberMeLabel = rememberMeLabel.getText();//2rd Verification

        if (actualRememberMeLabel.equals(expectedRememberMeLabel)){
            System.out.println("Label verification PASSED!");
        }else{
            System.out.println("Label verification FAILED!!!");
        }
        //4- Verify “forgot password” link text is as expected:
        //Expected: Forgot your password?
        WebElement forgotPasswordLink =
driver.findElement(By.className("login-link-forgot-pass"));
```

```
      String expectedForgotPasswordLinkText = "FORGOT YOUR PASSWORD?";//Remember
ifFail COZ WeJust compareValue 2gether
      String actualForgotPasswordLinkText = forgotPasswordLink.getText();//If Fail COZ WeJust
compareValue together

        if (actualForgotPasswordLinkText.equals(expectedForgotPasswordLinkText)){
           System.out.println("Forgot password link verification PASSED!");
        }else {
           System.out.println("actualForgotPasswordLinkText = " + actualForgotPasswordLinkText);
           System.out.println("expectedForgotPasswordLinkText = " +
expectedForgotPasswordLinkText);

           System.out.println("Forgot password link verification FAILED!!!");
        }
        //5- Verify “forgot password” href attribute’s value contains expected:
        //Expected: forgot_password=yes
        String expectedInHref = "forgot_password=yes";
        String actualHrefAttributeValue = forgotPasswordLink.getAttribute("href");
        System.out.println("actualHrefAttributeValue = " + actualHrefAttributeValue);

        if (actualHrefAttributeValue.contains(expectedInHref)){
           System.out.println("HREF attribute value verification PASSED!");
        }else{
           System.out.println("HREF attribute value verification FAILED!!!");
        }
    }
}
====================
package com.cydeo.tests.day3_CssSellector_xpath;
import com.cydeo.tests.utilities.WebDriverFactory;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
public class T3_getAttribute_cssSelector {
    public static void main(String[] args) {
        //TC #3: NextBaseCRM, locators, getText(), getAttribute() practice
        //1- Open a Chrome browser
        WebDriver driver = WebDriverFactory.getDriver("Chrome");
        driver.manage().window().maximize();

        //2- Go to: https://login1.nextbasecrm.com/
        driver.get("https://login1.nextbasecrm.com/");

        //3- Verify “Log in” button text is as expected:
```

```
        //Expected: Log In
      //WebElement
signInButton=driver.findElement(By.className("login-btn"));//oldWay.InsteadOfClasName use
cssSelector
        //LOCATING THE SAME WEB ELEMENT USING DIFFERENT ATTRIBUTE VALUES.
        //  REMEMBER                  tagName[attribute='value']
        //  REMEMBER                  input[class='login-btn']
        // Above statement CustomGenerator tagName toGive moreWay to locate the webElement
        // LOCATED USING CLASS ATTRIBUTE
        //WebElement signInButton = driver.findElement(By.cssSelector("input[class='login-btn']"));
        // LOCATED USING TYPE ATTRIBUTE
        //WebElement signInButton = driver.findElement(By.cssSelector("input[type='submit']"));
        // LOCATED USING VALUE ATTRIBUTE
        //lineBelowDifference is InsteadOfClasName use cssSelector & insteadOf passTestDirect
weFollowsThe CSS syntest
        //B4 Run my code I canPaste This input[class='login-btn'] to see ifLocate on myWeb to see
if Can find anything
        //If is not find on the WebPage inspect finding search Bar is not going to work on selenium
        //this version we can locate whateverWe want to locate getInput fromWeb &
      WebElement signInButton = driver.findElement(By.cssSelector("input[value='Log
In']"));//cssSelector & CSS syntest

        String expectedButtonText = "Log In";

        //Getting the name value of the attribute "value"
        String actualButtonText = signInButton.getAttribute("value");
        System.out.println("actualButtonText = " + actualButtonText)
        if (actualButtonText.equals(expectedButtonText)){
            System.out.println("Log In button text verification passed!");
        }else{
            System.out.println("Log In button text verification failed!");
        }
    }
}
=================
package com.cydeo.tests.day3_CssSellector_xpath;
import com.cydeo.tests.utilities.WebDriverFactory;
import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
public class T4_cssSelector_getText {

    public static void main(String[] args) {
        //TC #4: NextBaseCRM, locators, getText(), getAttribute() practice
        //1- Open a Chrome browser
        WebDriver driver = WebDriverFactory.getDriver("chrome");

        driver.manage().window().maximize();
        //2- Go to: https://login1.nextbasecrm.com/?forgot_password=yes
        driver.get("https://login1.nextbasecrm.com/?forgot_password=yes");

        //3- Verify "Reset password" button text is as expected:
        //Expected: Reset password
        //          tagName[attribute='value']

        //locating reset password button using class attribute and its value
        //WebElement resetPasswordButton =
driver.findElement(By.cssSelector("button[class='login-btn']"));

        //locating reset password button using VALUE attribute and its value
        WebElement resetPasswordButton =
driver.findElement(By.cssSelector("button[value='Reset password']"));

        String expectedResetPasswordButtonText = "Reset password";
        String actualResetPasswordButtonText = resetPasswordButton.getText();//Use
getTextBetween OpenTag &CloseTag

        if (actualResetPasswordButtonText.equals(expectedResetPasswordButtonText)){
            System.out.println("Button text verification PASSED!");
        }else{
            System.out.println("Button text verification FAILED!");
        }
    }
}
=================
package com.cydeo.tests.utilities;
// TASK: NEW METHOD CREATION
// Method name : getDriver
// Static method
// Accepts String arg: browserType
//   - This arg will determine what type of browser is opened
//   - if "chrome" passed --> it will open chrome browser
//   - if "firefox" passed --> it will open firefox browser
// RETURN TYPE: "WebDriver"

import io.github.bonigarcia.wdm.WebDriverManager;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.chrome.ChromeDriver;
import org.openqa.selenium.firefox.FirefoxDriver;
public class WebDriverFactory {

    //static method IDon't need clas object we say methodName WebDriver returnType getDriver
pass BrowserType String
    public static WebDriver getDriver(String browserType) {

        //If the String is Chrome I must set up and open ChromeBrowser below is how U setUp
WebBrowser
        // Plus use equalsIgnoreCase coz accepts all the matching keywords
        if (browserType.equalsIgnoreCase("chrome")) {//

            WebDriverManager.chromedriver().setup(); // do Chrome browser Setup
            return new ChromeDriver();//Just pass return a Chrome driver

        } else if (browserType.equalsIgnoreCase("firefox")) {
            WebDriverManager.firefoxdriver().setup();
            return new FirefoxDriver();
        } else {
            //U can give 1 more supporting argument inside the statement below
            System.out.println("Given browser type does not exist/or is not currently supported");
            System.out.println("Driver = null");// means variable does not refer to any object or array
            return null;
        }
    }
}
```