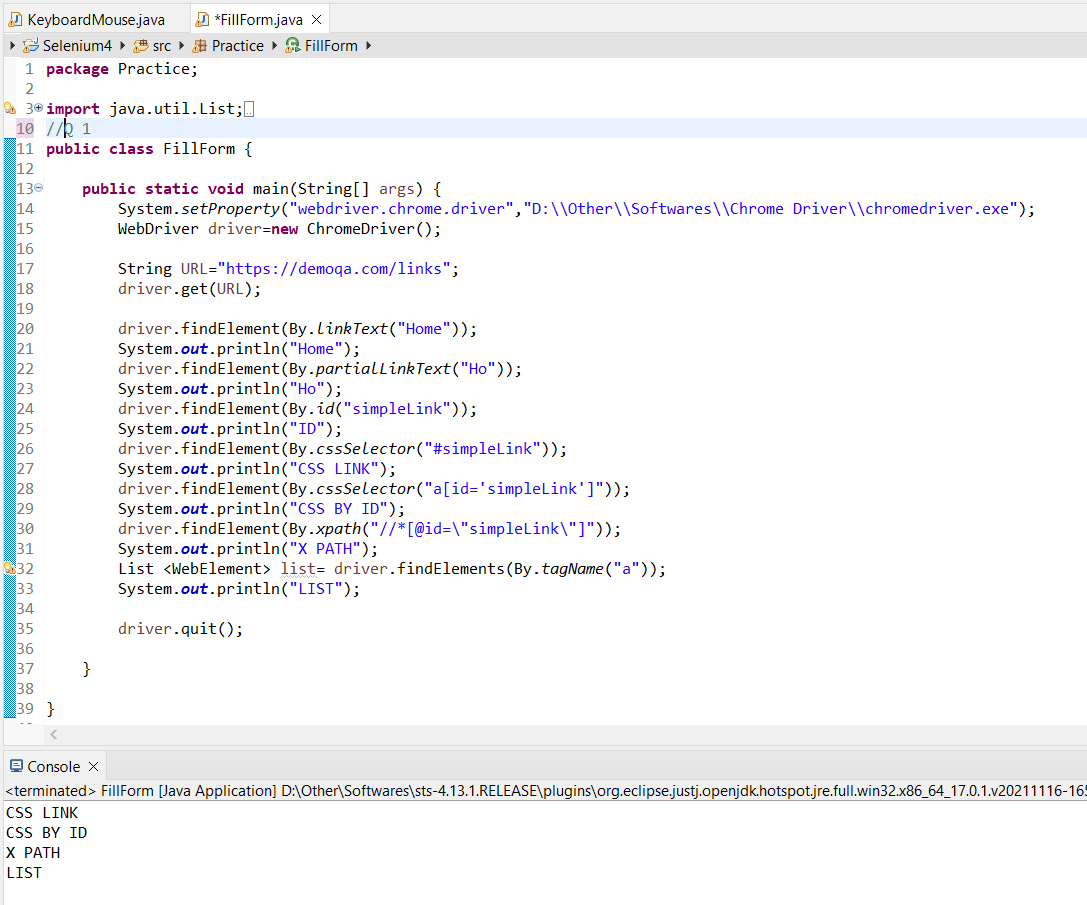
================================================================**Assignment 8**

================================================================

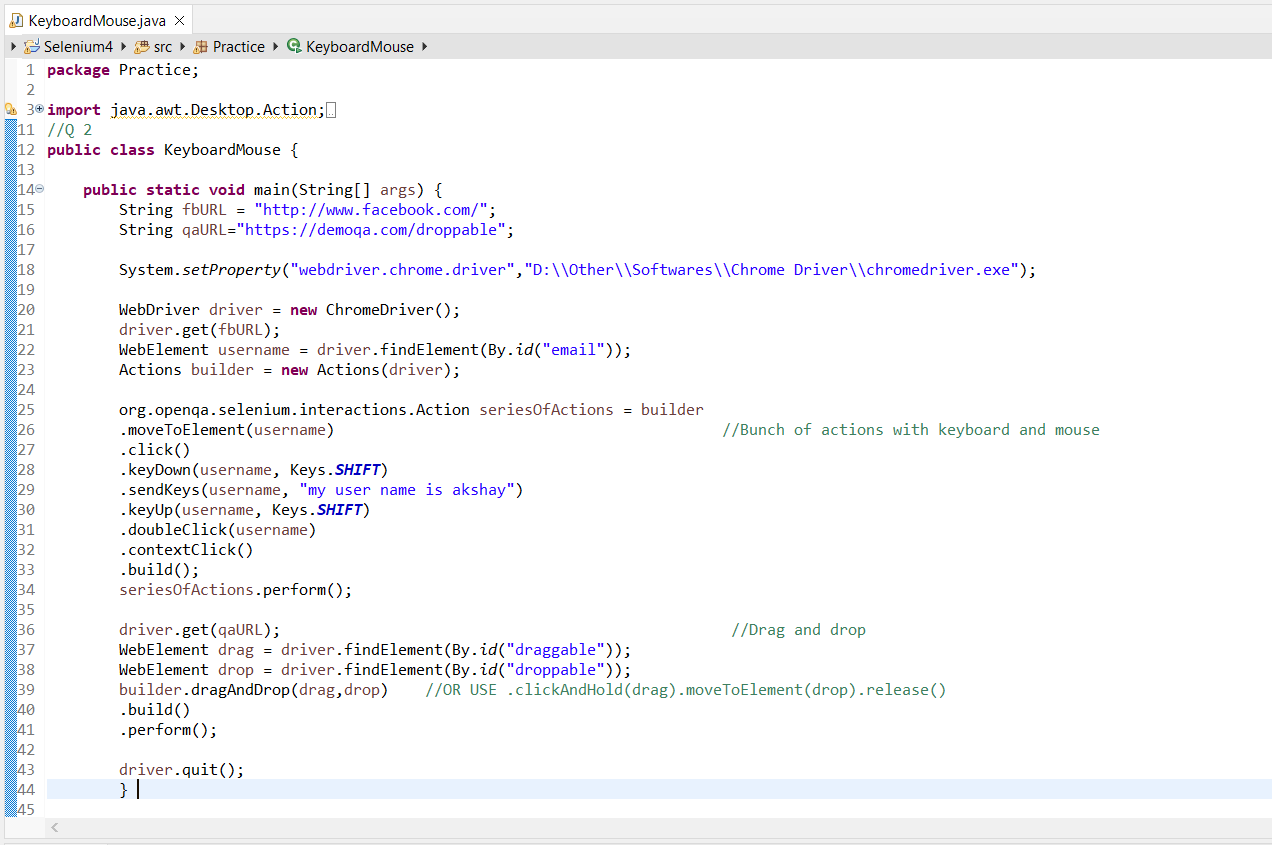
**Q 1. Test any website and write the script for finding elements by including all locators. By.className , By.cssSelector , By.id , By.linkText , By.name , By.partialLinkText , By.tagName , By.xpath**

****

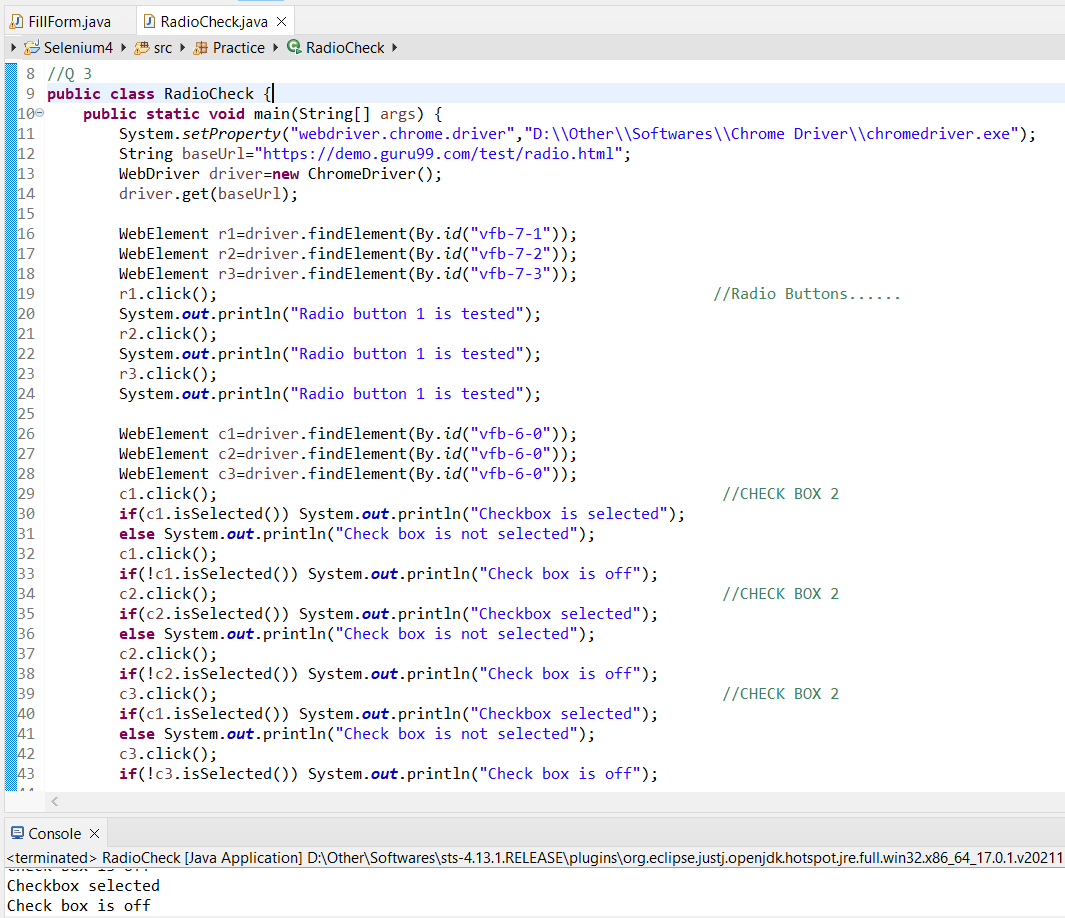
**Q 2. Test any website and write the script for that involves keyboard actions dragAndDrop(source,**

**target), keyDown(modifier\_key), keyUp(modifier\_key), moveToElement(toElement),**

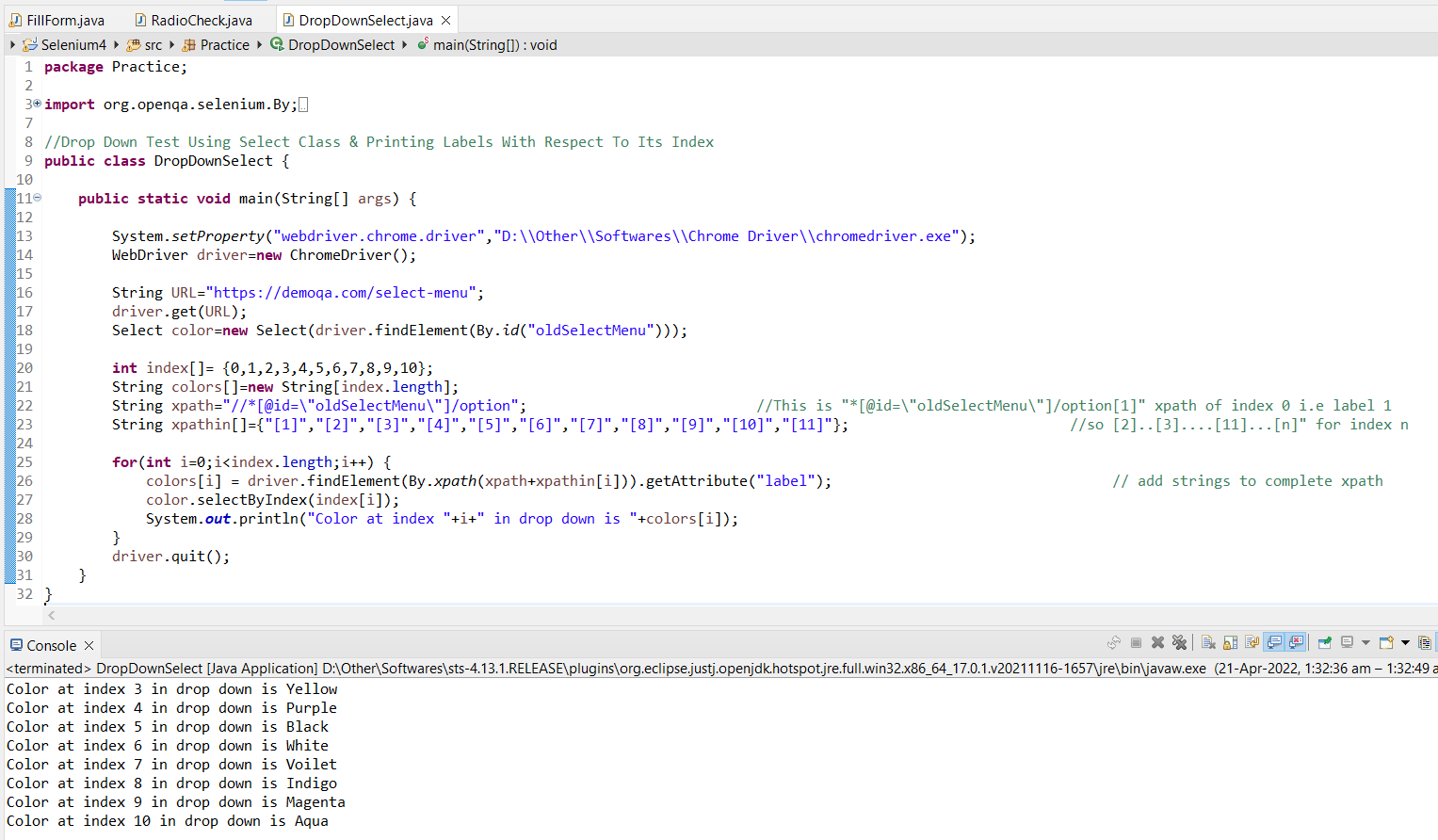
**release(),sendKeys(onElement, charsequence).**

****

**Q 3. Test any website and write the script for testing textbox , dropdown , checkbox and radio button for the registration form.**

****

1. RADIO AND CHECK BOX

****

B . Select & Print Drop Down Labels

**Q 4. What is meant by XPath in Selenium. Explain XPath Absolute and XPath Relative.**

XPath is a technique in Selenium to navigate through the HTML structure of a page. XPath enables testers to navigate through the XML structure of any document, and this can be used on both HTML and XML documents.

**A . Absolute XPath**

The simplest XPath locator example in Selenium is to provide the absolute path of an element in the DOM structure. For instance, consider the HTML below:



The syntax to select the business email field is as follows:

html/body/form/input[3]

This searches for the first form tag in the body of the page and selects the third input field in the form. This format, though simple, is also the most vulnerable to minor changes in the structure of the page. This method is also known as a single slash search.

**B . Relative XPath**

A relative path, or a double slash search, begins with double slashes. The double slashes signify a break in the absolute path. Here is how to select the same business email field using a relative path.

//form/input[3]

If there are multiple forms on the page, one may need to provide an extra identifier for the form field.

**Note :-**

**How to handle dynamic elements in Selenium using XPath?**

1. Using attributes

While the example shown above is feasible if there is only a single form on the page, one can make the search patterns more robust by using attributes.

//form[@id='loginForm']/input[3]

In place of id, one can use any attribute and its corresponding value to locate an element with Selenium.

While this example shows the use of a single attribute, one can also use multiple attributes of the same tag to locate it on the page. For instance, to select the Last Name field, one can use the following XPath syntax in Selenium:

//input[@name='name'][@value='Last Name']

### 2. Logical operators in selections

While attributes may be sufficient to locate elements in most cases, testers may feel the need of using logical operators too. For instance, if the HTML structure has name or id attributes populated by the value “name”, one may use the following syntax to select them.

//input[@id='name' or @name='name']

Similarly, one can replace the or keyword with and to only select an element that satisfies all conditions.

### 3. Using text

One may search for an element using the text that it contains too. For instance, to select a link that says “Click Me”, one can use the following search:

//a[text()='Click Me']

This snippet searches for any hyperlink that contains the text “Click Me” within it. Replace the tag with a wildcard \* to search for any element that contains the text “Click Me”.

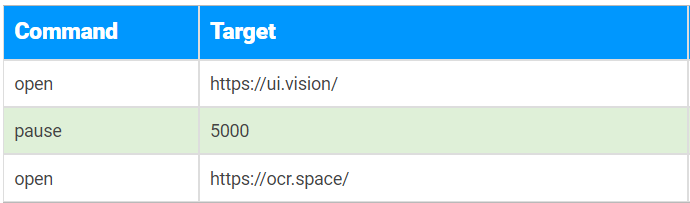
//\*[text()='Click Me']

**Q 5. Explain the pause feature in Selenium IDE.**

**pause (time in milliseconds)**

The pause command is a simple wait command and useful to delay the execution of the automated testing for the specified time. Note that the wait time is in milliseconds. So if you want to wait for 3 seconds, enter 3000.

In this example the Selenium IDE will wait for five seconds (5000ms).



**Q 6.** **With the help of code snippets, explain how we can create right-click and mouse hover actions in Selenium.**

A mouse hover is also called as hover. Mouse hover action is basically an action where a user places a mouse over a designated area like a hyperlink. It can cause some event to get triggered.

Actions actions = new Actions(driver);

As seen above, the moveToElement() method has argument WebElement target. So, it is required to pass a WebElement object to the method. This WebElement target is any web element where the mouse needs to be moved. To find the target element use the below command:

WebElement target = driver.findElement(Any By strategy & locator);

Here, you can use any **By strategy to locate the WebElement**like find element by its id, name attribute, etc.

Now, when we have got the actions class object and the element as well, just invoke **perform()** method to make a move to the element:

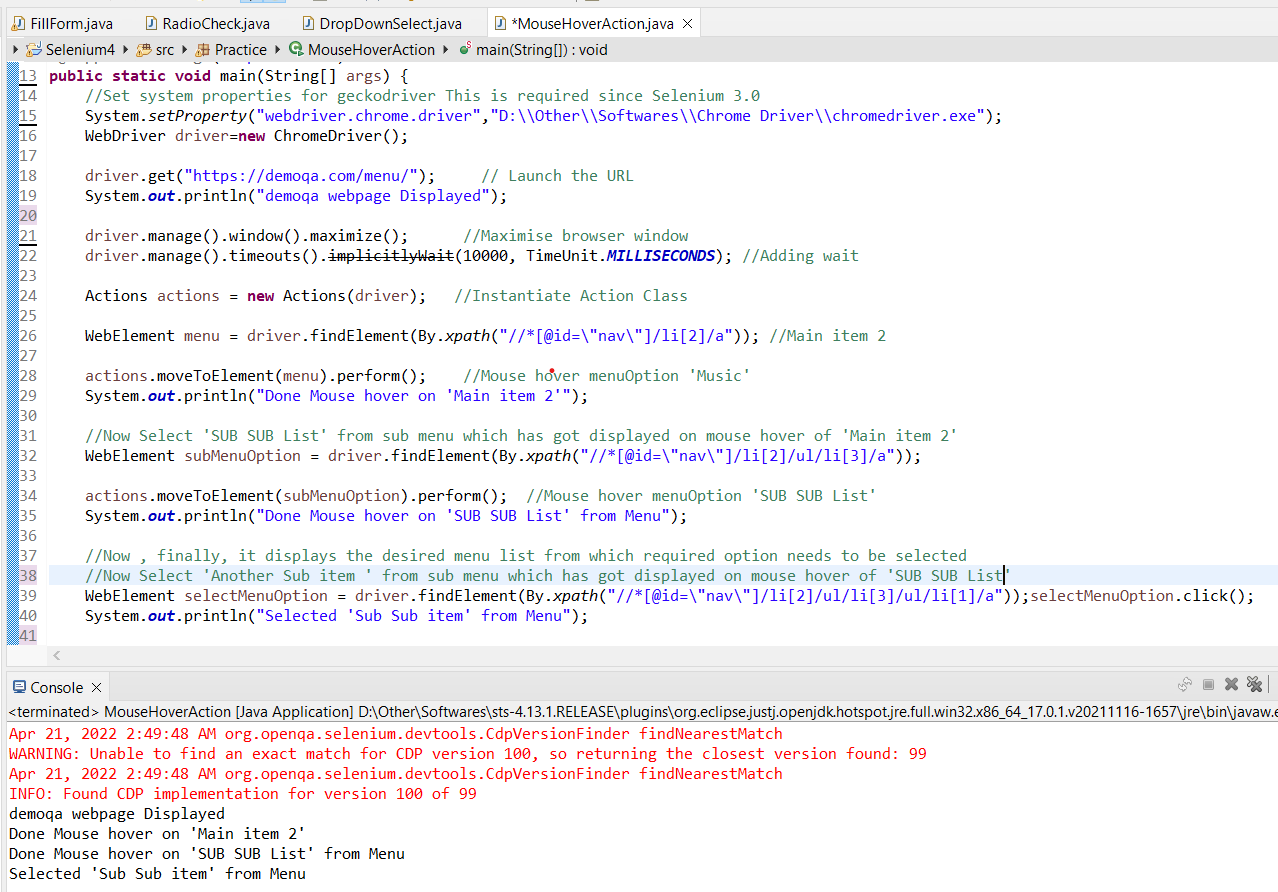
actions.moveToElement(target).perform();

Let’s see what happens internally when invoke the perform() method above:

* **Move Mouse Action :** Mouse gets moved to the middle of the element.
* Here, the element is scrolled into view and its location gets calculated using getBoundingClientRect.
* **Build :** build() method is used to generate a composite action containing all actions.
* In this case, it’s just a single move mouse action. If you observe, we have not invoked it in our above command. This is because the build is executed in the perform method internally.
* **Perform :** perform() method performs the actions we have specified. But before that, it internally invokes build() method first. After the build, the action is performed.

#### **Practice Exercise to Perform Mouse Hover using Actions Class in Selenium**

Let’s consider an example from an already available demo page on Toolsqa as [**https://demoqa.com/menu/**](https://demoqa.com/menu/)

****

Now it is the time to write a selenium automation script to mouse hover the object using moveToElement() method.

Find below the steps of the scenario to be automated:

1. Launch the web browser and launch our practice page https://demoqa.com/menu/
2. Find the required element i.e. Main item 2 object in our sample
3. Now move the mouse on Main item 2 option which will display the submenu list
4. Find the required element i.e. SUB SUB List object in the sub menu list
5. Move mouse on SUB SUB List option which will display one more submenu list
6. Find the required element i.e. ‘Alternative’ object in submenu list
7. Click on Sub Sub item

**Q 7. Can we handle a windows-based pop-up in Selenium, and if not, then what are the alternatives?**

Yes, it is possible to handle Windows based pop-ups in Selenium webdriver. Sometimes on clicking a link or a button, another window gets opened. It can be a pop up with information or an advertisement.

The methods getWindowHandles and getWindowHandle are used to handle child windows. The getWindowHandles method stores all the handle ids of the opened windows in the form of Set data structure.

The getWindowHandle method stores the handle id of the window in focus. Since the getWindowHandles method holds all the opened window handle ids, we can iterate through these handle ids with the iterator and next methods.

To switch to a specific window, switchTo.().window() method can be used. The handle id of the window where we want to switch is passed as a parameter to this method.

