A web page consists of various web elements ranging from text fields, buttons, checkboxes, drop-downs, etc. With the wide use of forms on the websites these days, we come across ***DropDowns*** now and then. There are various types of *dropdowns* available these days, majorly of which are categorized as a ***single-select*** *(which allows selecting only one value)* or ***multi-select*** *(which allows selecting multiple values).* ***Selenium WebDriver*** provides a class named ***"Select"***, which provides various methods to handle the *dropdowns,* be it single-select or multi-select *dropdowns.* In this article, we will understand the intricacies of the ***"Select "*** class of *Selenium WebDriver* and will understand how we can handle *dropdown in Selenium* by covering the details under the following topics:

What is Select class in Selenium?
How to select a value from a dropdown in Selenium?
How to select multiple values from a dropdown in Selenium?
Also, how to get options from a dropdown in Selenium?
How to deselect a value from a dropdown Selenium?
Examples illustrating Select class usage in Selenium.
Example 1 - Handling dropdown using Selenium WebDriver.
Example 2 - Handling multi-select using Selenium WebDriver.

What is Select Class in Selenium?

In *HTML,* the *dropdowns* are generally implemented either using the `<select>` tag or the `<input>` tag. To perform certain operations on the *dropdowns,* which are declared using the `<select>` **HTML tag,** *Selenium WebDrivers* provides a class called ***"Select "*** class. As soon as you start typing the ***"Select "*** in your ***IDE,*** it will show the details as shown below:

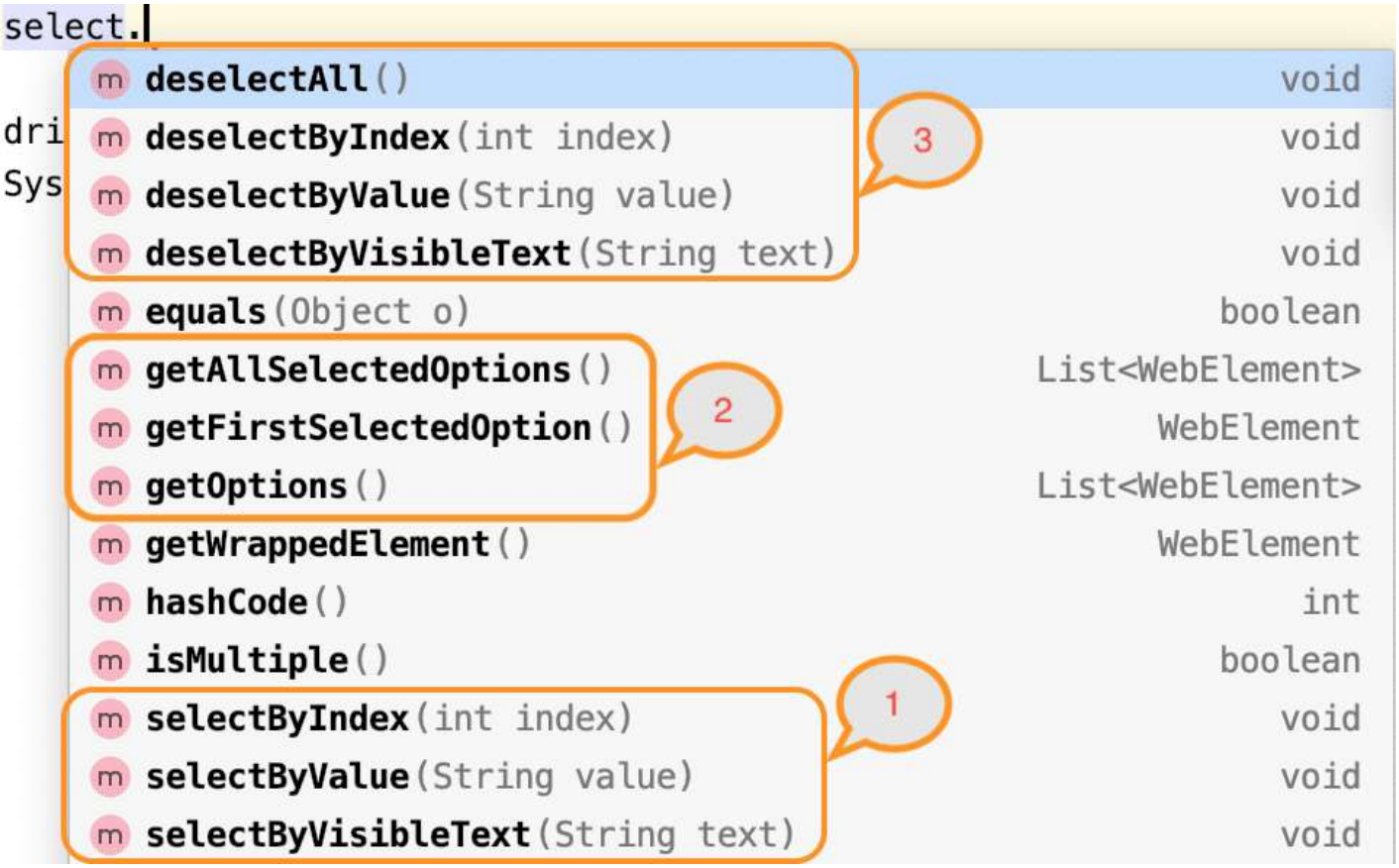
As we can see from the above screenshot, the ***"Select "*** class is provided by the ***"org.openqa.selenium.support.ui "*** package of *Selenium WebDriver.* You can create an object of the Select class, by-passing the object of the ***"WebElement"*** class, which shows the object returned by the corresponding ***locator*** of the *WebElement.*

So, you can create the object of the ***Select*** class by using the following syntax:

```
Select select = new Select(WebElement webelement);
```

The *Select class constructor* accepts one parameter: the WebElement object returned by the locators of the specified element.

The ***"Select "*** class provides various methods for handling the *dropdown* operations. The following figures list a few of them:



Let's now understand the syntax and usage of various select and deselect methods provided by the ***"Select "*** class in *Selenium WebDriver.*

How to select a value from a dropdown in Selenium?

As highlighted in the above figure, the *Select class* of *Selenium WebDriver* provides the following methods to select an option/value from a drop-down *(as highlighted by marker 1 in the above image):*

selectByIndex
selectByValue
selectByVisibleText

Let's understand the syntax and usage of all these methods:

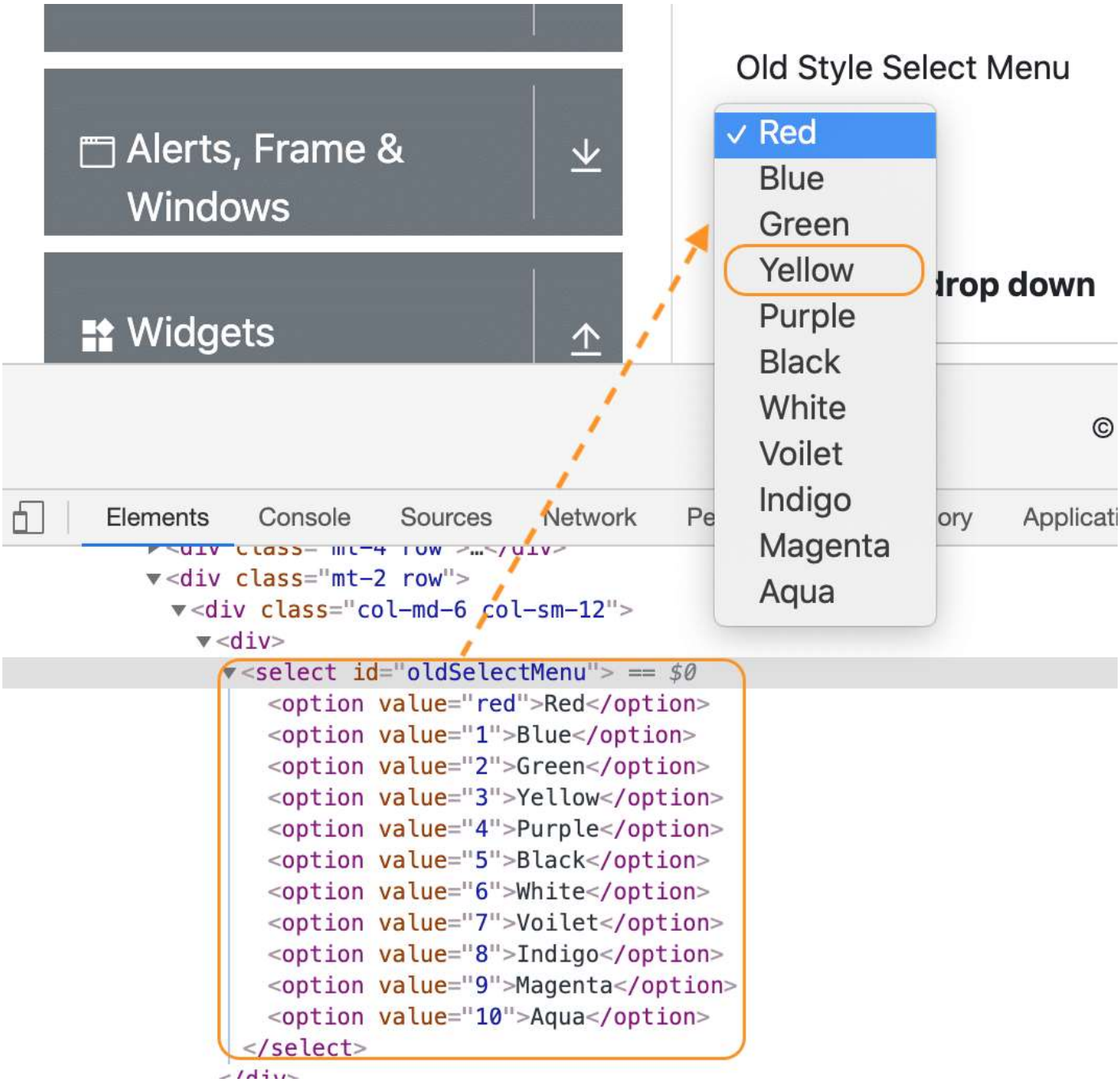
selectByIndex:

This method selects the *dropdown* option by its *index number.* We provide an integer value as the index number as an argument. It possesses the following syntax:

```
selectByIndex(int arg0) : void
```

i.e., it accepts the index of the *dropdown* value, which needs to be selected. The index starts at 0.

Suppose on the web page ***"https://demoqa.com/select-menu"*** we have the select the 4th value of the *dropdown* as highlighted below:



As we can see, the above *dropdown* is being implemented using the `<select>` *HTML* tag, so we can use the ***"Select "*** class of *Selenium WebDriver* to select the option ***"Yellow "*** using

the *index* as shown below:

```
// Create object of the Select class
Select se = new Select(driver.findElement(By.xpath("//*[@id='oldSelectMenu']")));

// Select the option by index
se.selectByIndex(3);
```

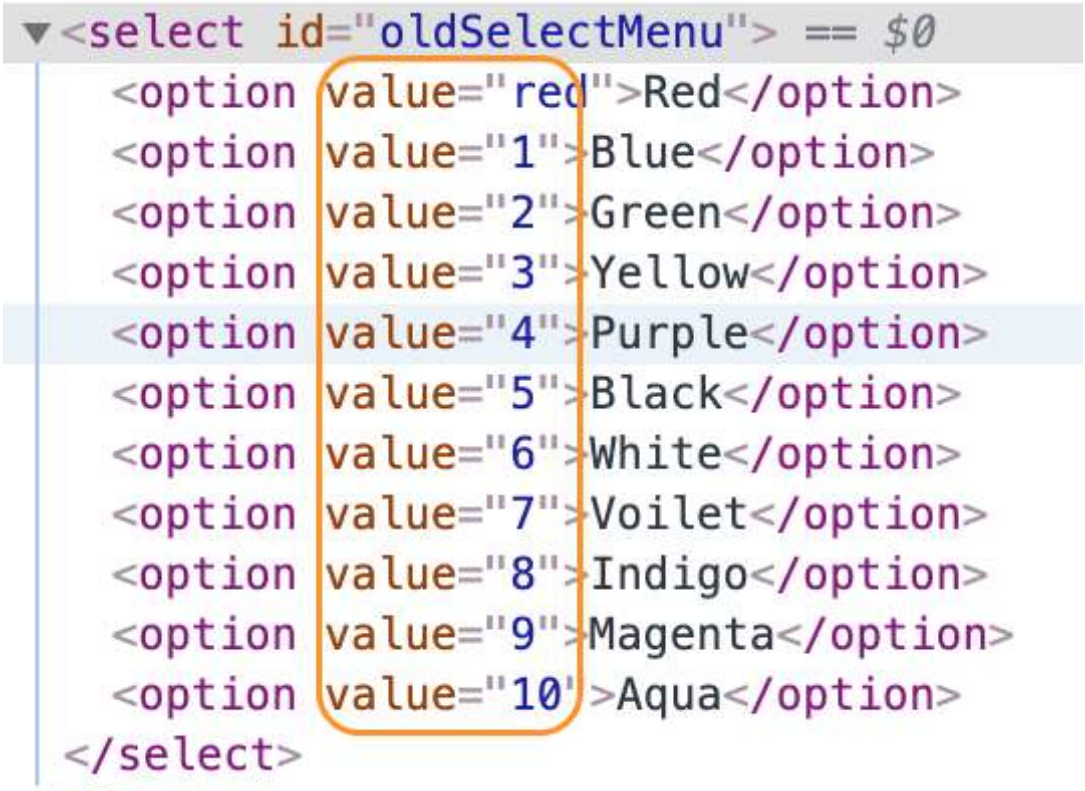
As we mentioned, the indices of *dropdown* start at 3, so the value ***"Yellow "*** can be selected using index 3.

selectByValue

This method selects the *dropdown* option by its *value.* We provide a string value as the value as an argument. It possesses the following syntax:

```
selectByValue(String arg0) : void
```

If we consider the same *dropdown* on page ***"https://demoqa.com/select-menu"***, as in the previous section, we can see that each of the options of the *dropdown* has an assigned value as shown below:



Now, if we have to select the option ***"White"***, we can use its value ***"6 ",*** as shown in the following code snippet:

```
// Create object of the Select class
Select se = new Select(driver.findElement(By.xpath("//*[@id='oldSelectMenu']")));

// Select the option with value "6"
se.selectByValue("6");
```

As the value ***"6 "*** corresponds to the option ***"White,"*** so it will select the value **"White"** from the *dropdown.*

selectByVisibleText

This method enables one to select one option from the *dropdown* or multi-select *dropdown* based on the *dropdown text.* You need to pass the *String value* of the `<select>` element as an argument. It possesses the following syntax:

```
selectByVisibleText(String arg0): void
```

If we consider the same *dropdown* on page ***"https://demoqa.com/select-menu"***, as in the previous section, we can see that each of the options of the *dropdown* will have a text value, which is displayed on the web page also, so we can use that text to select the corresponding option, as shown below:

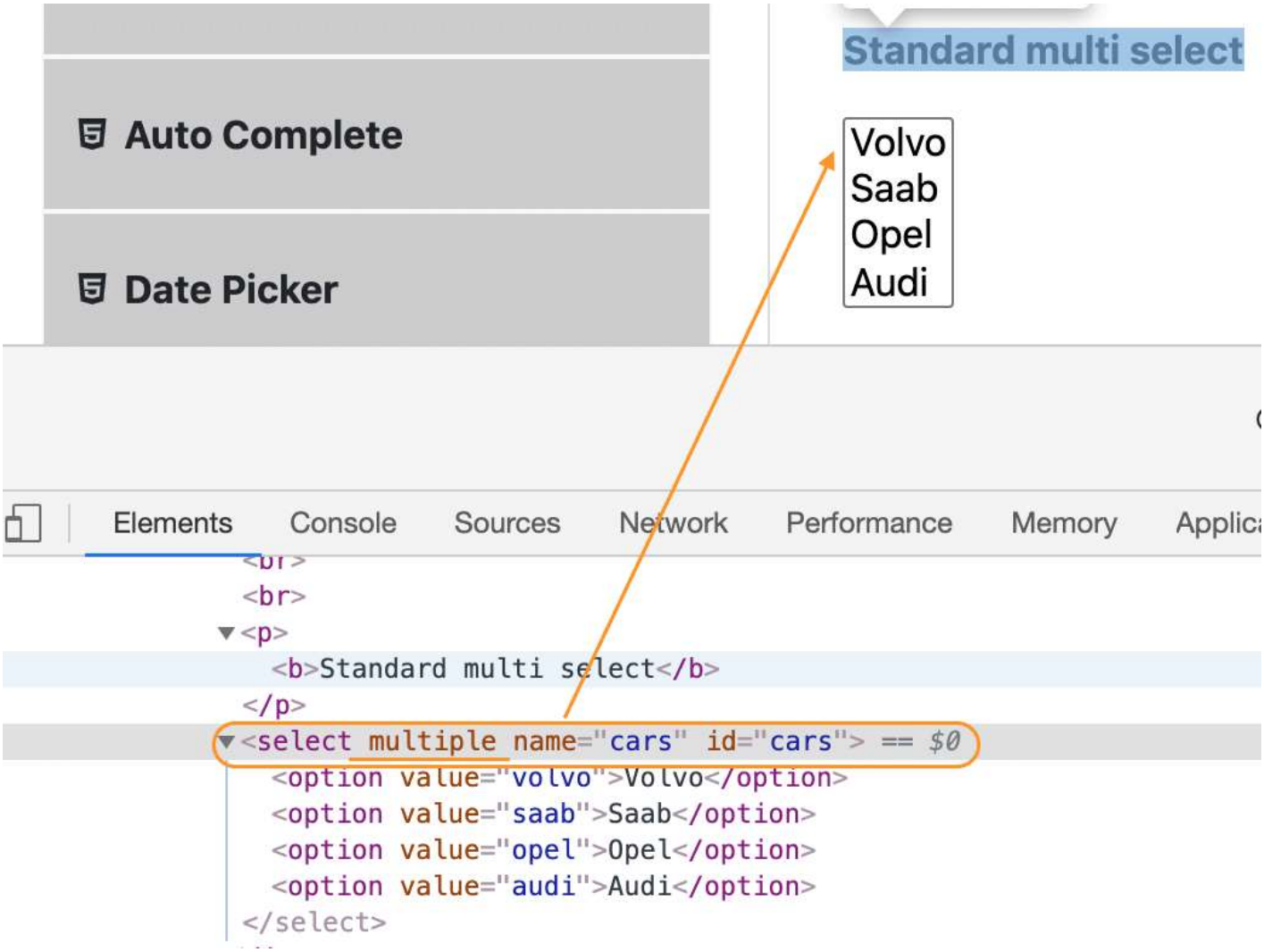
```
// Create the object of the Select class
Select se = new Select(driver.findElement(By.xpath("//*[@id='oldSelectMenu']")));

// Select the option using the visible text
se.selectByVisibleText("White");
```

As the *dropdown* has one of the options having the text as ***"White"***, the same will be selected using the above code.

Apart from the *dropdown* types briefed above, the *HTML* `<select>` tag also provides ways to define *dropdowns,* which allows selecting multiple values. Let's see how a *Multi-Select dropdown* will be declared and how we can select multiple options in a *dropdown* using the ***"Select "*** class of *Selenium WebDriver.*

How to select multiple values from a dropdown in Selenium?

If the `<select >` tag contains *multiple attributes,* it means that the *dropdown* allows selecting multiple values. As we can see in the following screenshot from the web page ***"https://demoqa.com/select-menu":***



We can use any of the methods we used to select one value from the *dropdown* to select multiple values by invoking the methods multiple times for different values. The ***"Select "*** class provides a method, *isMultiple(),* using which we can first validate whether the *dropdown* allows us to select multiple values. Let's see how to use the *isMultiple()* method:

How to check whether dropdown is Multi-Select?

As we discussed, the Select class provides the ***"isMultiple() "*** method, which determines whether the web element in say supports multiple selections. It returns a boolean value, i.e., *True/False,* without taking any argument. It checks the attribute *'multiple'* in the *HTML* code for the web element. Consequently, it possesses the following syntax:

```
isMultiple(): boolean
```

Once you determine whether the web element is multi-select or not, you can use the Select class's various select methods on the multiple values you intend to select. The below example code shows the same-

```
Select oSel = new Select(driver.findElement(By.xpath(//*[@id='cars']);

if(oSel.isMultiple()){

        //Selecting multiple values by index
        oSel.selectByIndex(1);
        oSel.selectByIndex(2);

        //Or selecting by values
        oSel.selectByValue("volvo");
        oSel.selectByValue("audi");

        //Or selecting by visible text
        oSel.selectByVisibleText("Volvo");
        oSel.selectByVisibleText("Opel");
}
```

And that is how you can select multiple values from a multi-select *dropdown.*

Now that we have understood how we can select values from a *dropdown,* be it single-select or multi-select, we should have some way to check which values the *dropdown* contains and what all values are selected in the *dropdown.* The ***"Select "*** class provides methods to get options from the dropdown. Let's understand the details and usage of these methods:

How to get options from a dropdown in Selenium?

As highlighted by marker 2, in the image under the ***"Select "*** class section above, the *Select* class provides the following methods to get the options of a *dropdown:*

getOptions()
getFirstSelectedOption()
getSelectedOptions()

Let's understand the details of all these methods:

getOptions

There are times when you need to get all the options in a *dropdown* or multi-select box. This is where you can use the *getOptions()* method of the *Select* class. It possesses the following syntax:

```
getOptions(): List<WebElement>
```

As we can see, this method returns all the options of the *dropdown* as a list of ***WebElement.*** The following code snippet shows how we can get all the options of the *dropdown* on the page ***"https://demoqa.com/select-menu":***

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get all the options of the dropdown
List<WebElement> options = select.getOptions();
```

Using this method, we can retrieve all the options of a *dropdown (be it single-select or multi-select).*

getFirstSelectedOption()

This method returns the first selected option of the *dropdown.* If it is a single-select *dropdown,* this method will return the selected value of the *dropdown,* and if it is a multi-select *dropdown,* this method will return the first selected value of the *dropdown.* It possesses the following syntax:

```
getFirstSelectedOption(): WebElement
```

As we can see, this method returns a ***WebElement.*** The following code snippet shows how we can get the first selected option of the *dropdown* on the page ***"https://demoqa.com/select-menu":***

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get the first selected option of the dropdown
WebElement firstSelectedOption = select.getFirstSelectedOption();
```

Using this method, we can retrieve the first selected option of a *dropdown (be it single-select or multi-select).*

getAllSelectedOptions()

This method returns all the selected options of the *dropdown.* If it is a single-select *dropdown,* this method will return the only selected value of the *dropdown,* and if it is a multi-select *dropdown,* this method will return *all the selected values* of the *dropdown.* It possesses the following syntax:

```
getAllSelectedOptions():List<WebElement>
```

As we can see, this method returns a *list of* ***WebElements.*** The following code snippet shows how we can get all the selected options of the *dropdown* on the page ***"https://demoqa.com/select-menu":***

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

// Get all the selected option of the dropdown
List<WebElement> selectedOptions = select.getAllSelectedOptions();
```

Using this method, we can retrieve all the selected options of a *dropdown (be it single-select or multi-select).*

How to deselect a value from a dropdown in Selenium?

Just like we select values in a *DropDown & Multi-Select,* we can deselect the values too. But the deselect method works only for *Multi-Select.* You can deselect pre-selected options from a *Multi-select* element using the different *deselect* methods discussed here. As we would have observed in the screenshot showing methods of the ***"Select "*** class *(shown by marker 3),* the Select class provides the following methods to deselect values of a *dropdown:*

deselectAll()
deselectByIndex()
deselectByValue()
deselectByVisibleText()

Let's understand the details and usage of all these methods:

deselectAll

This method will clear all the selected entries of the *dropdown.* It possesses the following syntax:

```
deselectAll(): void
```

If there are few options already selected in a *dropdown,* you can deselect all the options using the method *deselectAll().* The following code snippet shows a sample example, how we deselect all the values from the *dropdown:*

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect all the options
select.deselectAll();
```

It will deselect all the options from the *dropdown.*

deselectByIndex

Similar to the *selectByIndex()* method, the Select class also provides the method to *deselect* an option from the *dropdown* using the *deselectByIndex()* method. You can use the option's index number to deselect it. It possesses the following syntax:

```
deselectByIndex(int arg0): void
```

So, if there are few options already selected in a *dropdown,* you can *deselect* one of the *options* using the method *deselectByIndex().* The following code snippet shows a sample example, how we deselect one of the values from the *dropdown* by specifying its index:

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect first value by index
select.deselectByIndex(1);
```

It will deselect the option at index 1 in the *dropdown.*

deselectByValue

Similar to the *selectByValue()* method, the *Select* class also provides the method to deselect an option from the *dropdown* using the ***deselectByValue()*** method. You can use the option's value to deselect it. It possesses the following syntax:

```
deselectByValue(String arg0): void
```

So, if there are few options already selected in a *dropdown,* you can *deselect* one of the options using *deselectByValue().* The following code snippet shows a sample example, how we deselect one of the values from the *dropdown* by specifying its value:

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

//Deselect option with value "6"
select.deselectByValue("6");
```

It will deselect the option with value in the *dropdown.*

deselectByVisibleText

Similar to the *selectByVisibleText()* method, the *Select* class also provides the method to *deselect* an option from the *dropdown* using the ***deselectByVisibleText()*** method. You can use the option's text to deselect it. It possesses the following syntax:

```
deselectByVisibleText(String arg0): void
```

So, if there are few options already selected in a *dropdown,* you can *deselect* one of the options using the method *deselectByVisibleText().* The following code snippet shows a sample example, how we deselect one of the values from the *dropdown* by specifying its text:

```
Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

 //Deselect option with text "White"
select.deselectByVisibleText("White");
```

It will deselect the option with the text ***"White "*** in the *dropdown.*

Examples illustrating Select class usage in Selenium

We will cover two examples, one with a simple *dropdown* and another with a multi-select *dropdown.* We will be using the ***ToolsQA demo*** website to automate these scenarios. Let us now quickly start *dropdown/*multi-select automation.

Example 1- Handling dropdown using Selenium WebDriver

Our use-case would follow the steps below-

Launch the browser.
*Open **"https://demoqa.com/select-menu"**.*
Select the Old Style Select Menu using the element id.
Print all the options of the dropdown.
Select 'Purple' using the index.
After that, select 'Magenta' using visible text.
Select an option using value.
Close the browser

Using the methods of *Select class* as discussed above, the code would look like below-

```
import java.util.List;
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 DropDown {

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

        //Creating instance of Chrome driver
        WebDriver driver = new ChromeDriver();

        //Step#2- Launching URL
        driver.get("https://demoqa.com/select-menu");

        //Maximizing window
        driver.manage().window().maximize();

        //Step#3- Selecting the dropdown element by locating its id
        Select select = new Select(driver.findElement(By.id("oldSelectMenu")));

        //Step#4- Printing the options of the dropdown
        //Get list of web elements
        List<WebElement> lst = select.getOptions();

        //Looping through the options and printing dropdown options
        System.out.println("The dropdown options are:");
        for(WebElement options: lst)
            System.out.println(options.getText());

        //Step#5- Selecting the option as 'Purple'-- selectByIndex
        System.out.println("Select the Option by Index 4");
        select.selectByIndex(4);
        System.out.println("Select value is: " + select.getFirstSelectedOption().getText(

        //Step#6- Selecting the option as 'Magenta'-- selectByVisibleText
        System.out.println("Select the Option by Text Magenta");
        select.selectByVisibleText("Magenta");
        System.out.println("Select value is: " + select.getFirstSelectedOption().getText(
```

```
        //Step#7- Selecting an option by its value
        System.out.println("Select the Option by value 6");
        select.selectByValue("6");
        System.out.println("Select value is: " + select.getFirstSelectedOption().getText(

        driver.quit();
    }

}
```

On executing the code, you will notice that the *dropdown* selections are made as per the select method used, and the console window would print the options as shown below:

```
The dropdown options are:
Red
Blue
Green
Yellow
Purple
Black
White
Voilet
Indigo
Magenta
Aqua
Select the Option by Index 4
Select value is: Purple
Select the Option by Text Magenta
Select value is: Magenta
Select the Option by value 6
Select value is: White
```

So this way, we can select and validate the values in a *dropdown* allowing single-select.

Example 2- Handling multi-select using Selenium WebDriver

To automate multi-select using *Selenium WebDriver's* Select class, we will use the following use-case:

Launch the browser.
*Open **"https://demoqa.com/select-menu"**.*
Select the Standard Multi-Select using the element id.
Verifying that the element is multi-select.
Select 'Opel' using the index and deselect the same using index.
Select 'Saab' using value and deselect the same using value.
Deselect all the options.
Close the browser.

We will use both the select and deselect methods of the Select class to automate the multi-select element. The code would look like below-

```
import java.util.List;

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 MultiSelect {

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

        //Creating instance of Chrome driver
        WebDriver driver = new ChromeDriver();

        // Navigate to the URL
        driver.get("https://demoqa.com/select-menu");


        //Maximizing window
        driver.manage().window().maximize();

        //Selecting the multi-select element by locating its id
        Select select = new Select(driver.findElement(By.id("cars")));

        //Get the list of all the options
        System.out.println("The dropdown options are -");

        List<WebElement> options = select.getOptions();

        for(WebElement option: options)
            System.out.println(option.getText());

        //Using isMultiple() method to verify if the element is multi-select, if yes go o
        if(select.isMultiple()){

            //Selecting option as 'Opel'-- ByIndex
            System.out.println("Select option Opel by Index");
            select.selectByIndex(2);
```

```
            //Selecting the option as 'Saab'-- ByValue
            System.out.println("Select option saab by Value");
            select.selectByValue("saab");

            // Selecting the option by text
            System.out.println("Select option Audi by Text");
            select.selectByVisibleText("Audi");

            //Get the list of selected options
            System.out.println("The selected values in the dropdown options are -");

            List<WebElement> selectedOptions = select.getAllSelectedOptions();

            for(WebElement selectedOption: selectedOptions)
                System.out.println(selectedOption.getText());


            // Deselect the value "Audi" by Index
            System.out.println("DeSelect option Audi by Index");
            select.deselectByIndex(3);

            //Deselect the value "Opel" by visible text
            System.out.println("Select option Opel by Text");
            select.deselectByVisibleText("Opel");

            //Validate that both the values are deselected
            System.out.println("The selected values after deselect in the dropdown option
            List<WebElement> selectedOptionsAfterDeselect = select.getAllSelectedOptions(

            for(WebElement selectedOptionAfterDeselect: selectedOptionsAfterDeselect)
                System.out.println(selectedOptionAfterDeselect.getText());

            //Step#8- Deselect all values
            select.deselectAll();
        }

        driver.quit();
    }


}
```

The above code on execution would select and deselect multiple options from the multi-select and print the multi-select options, as shown below:

```
The dropdown options are -
Volvo
Saab
Opel
Audi
Select option Opel by Index
Select option saab by Value
Select option Audi by Text
The selected values in the dropdown options are -
Saab
Opel
Audi
DeSelect option Audi by Index
Select option Opel by Text
The selected values after deselect in the dropdown options are -
Saab
```

You can select multiple options, just like we selected all the options from the multi-select box as per your requirement. Now you can use the different *Select* class methods in *Selenium* automation and easily automate a *dropdown* or a multi-select box to ease your execution.

Key Takeaways

*The Select class in selenium can be used by importing the **org.openqa.selenium.support.ui.Select package.***
Moreover, the Select Class provides different methods to select values from dropdown/multi-select.
The Select Class also provides deselection methods to deselect certain values from a dropdown.
The multiple attributes can differentiate the multi-select elements provided bt the `<select>` tag.
The select and deselect methods can use an index, visible text, or values to select/deselect the values from a dropdown.