CSS Selector

CSS-Selector is a combination of attribute name and attribute value which identifies the web element on the web page. (Or)

CSS selectors are patterns used to select the element(s). (Or)

CSS-selector is a string representation of HTML tags, attributes, ID and class. (Or)

CSS-selector is a pattern which identifies the element in a HTML tree.

Note:

- When do we use CSS selector?
  When we don't have any option to choose ID or Name or Class, We should prefer using the CSS selector locators.
- CSS stands for (Cascading Style Sheets).

Types of CSS selector

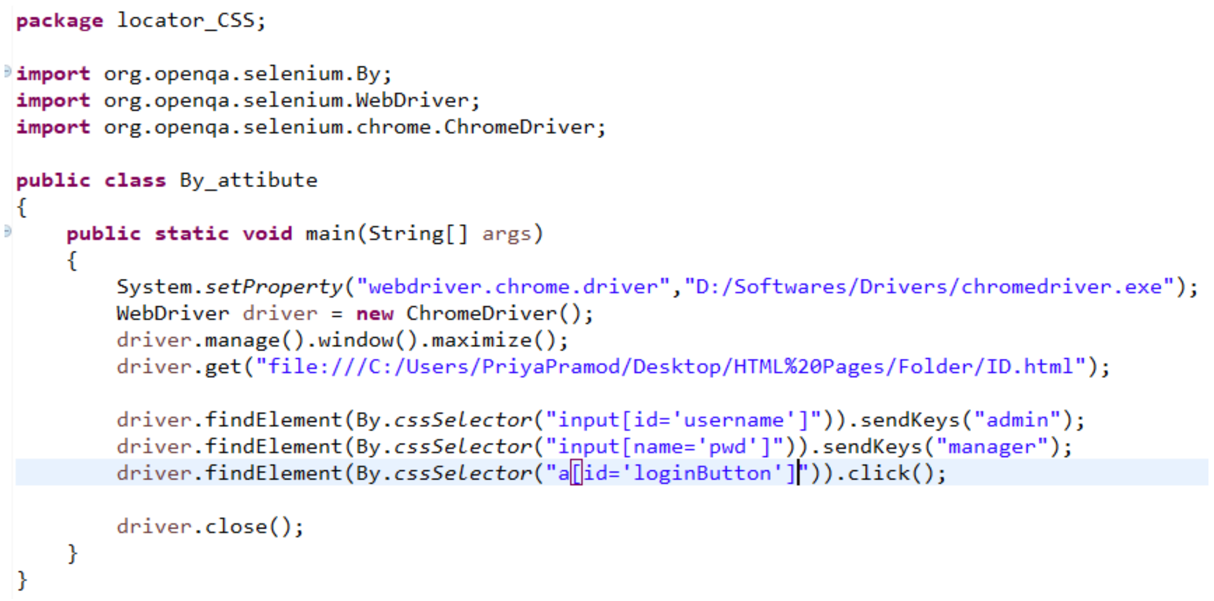
- CSS by Attribute
- CSS by ID
- CSS by Name
- CSS by Immediate Child
- CSS by Any Child
- CSS by Next Sibling
- CSS by Specific Match
  1. First-child
  2. Last-child
  3. Nth-child
  4. Nth-of-type
- CSS by String Match
  1. Prefix
  2. Suffix
  3. Contains

1. **CSS by Attribute**

- CSS by attribute selector is used to select elements with a specified attribute.

Syntax:
HTMLTAG [AttributeName='AttibuteValue']

Write a script to login to actiTIME using CSS by Attribute?



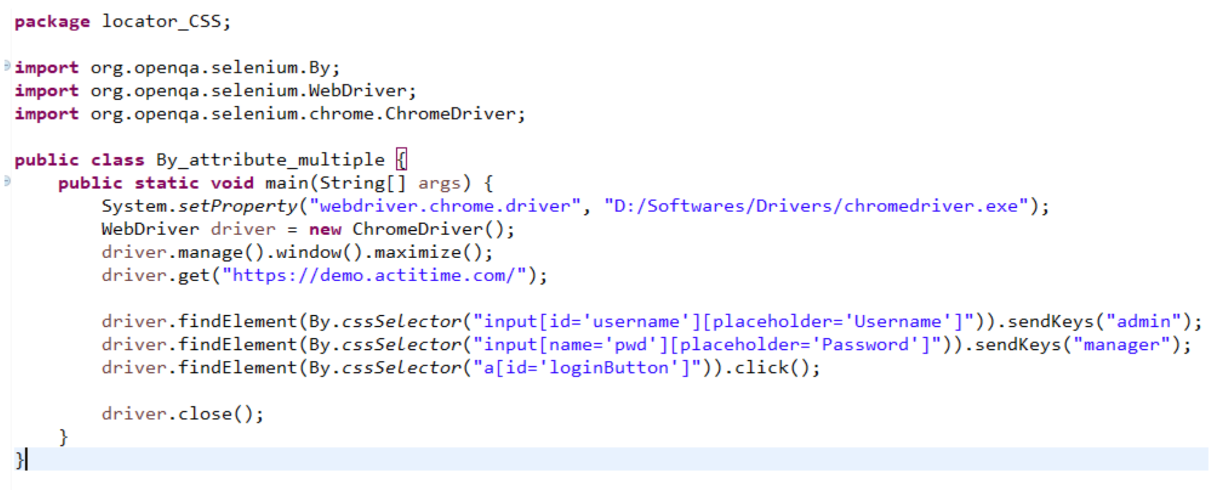
```
package locator_CSS;

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

public class By_attibute
{
    public static void main(String[] args)
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///C:/Users/PriyaPramod/Desktop/HTML%20Pages/Folder/ID.html");

        driver.findElement(By.cssSelector("input[id='username']")).sendKeys("admin");
        driver.findElement(By.cssSelector("input[name='pwd']")).sendKeys("manager");
        driver.findElement(By.cssSelector("a[id='loginButton']")).click();

        driver.close();
    }
}
```

Note: We can use multiple attributes to identify the element in CSS.

Syntax to use multiple attributes in CSS.

HTMLTAG [AttributeName='AttibuteValue'] [AttributeName='AttibuteValue']

Script to use multiple attributes to identify the elements?



```
package locator_CSS;

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

public class By_attribute_multiple {
    public static void main(String[] args) {
        System.setProperty("webdriver.chrome.driver", "D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        driver.findElement(By.cssSelector("input[id='username'][placeholder='Username']")).sendKeys("admin");
        driver.findElement(By.cssSelector("input[name='pwd'][placeholder='Password']")).sendKeys("manager");
        driver.findElement(By.cssSelector("a[id='loginButton']")).click();

        driver.close();
    }
}
```

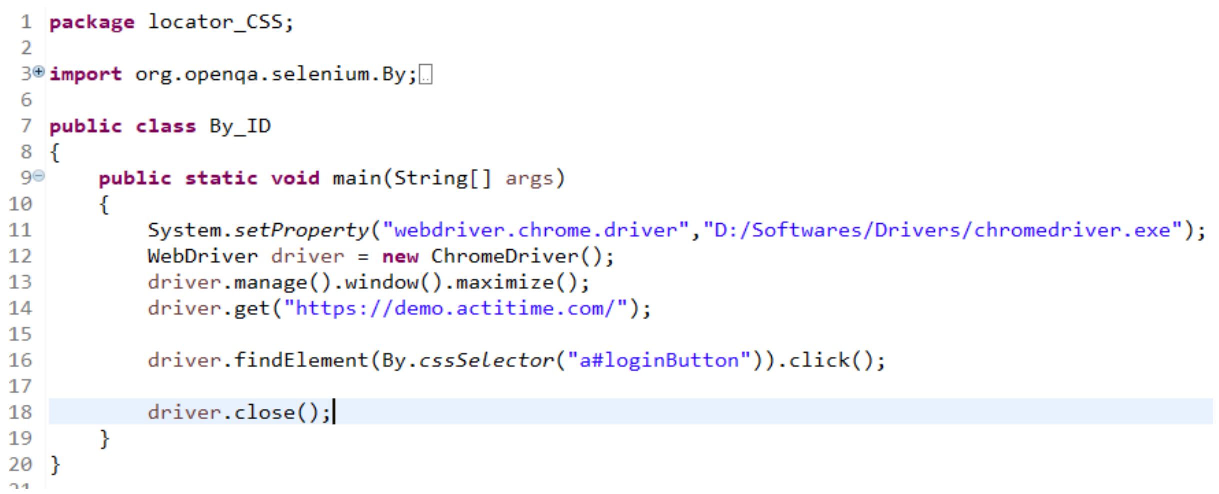
2. CSS by ID

- CSS selector uses the **ID** attribute of an HTML element to select a specific element.
- To select an element with a specific id, write a **hash (#)** character, followed by the id of the element.

  Syntax:

  "TagName#IDAttributeValue"

Example program:



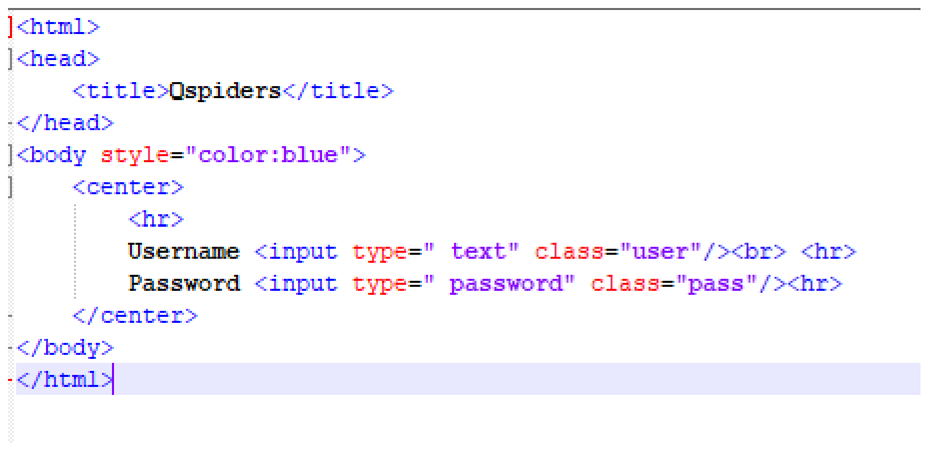
```
package locator_CSS;

import org.openqa.selenium.By;

public class By_ID
{
    public static void main(String[] args)
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/");

        driver.findElement(By.cssSelector("a#loginButton")).click();

        driver.close();
    }
}
```

3. CSS by Class name

- CSS selector selects elements with a specific class attribute.
- To select elements with a specific class, write a period (.) character, followed by the name of the class.

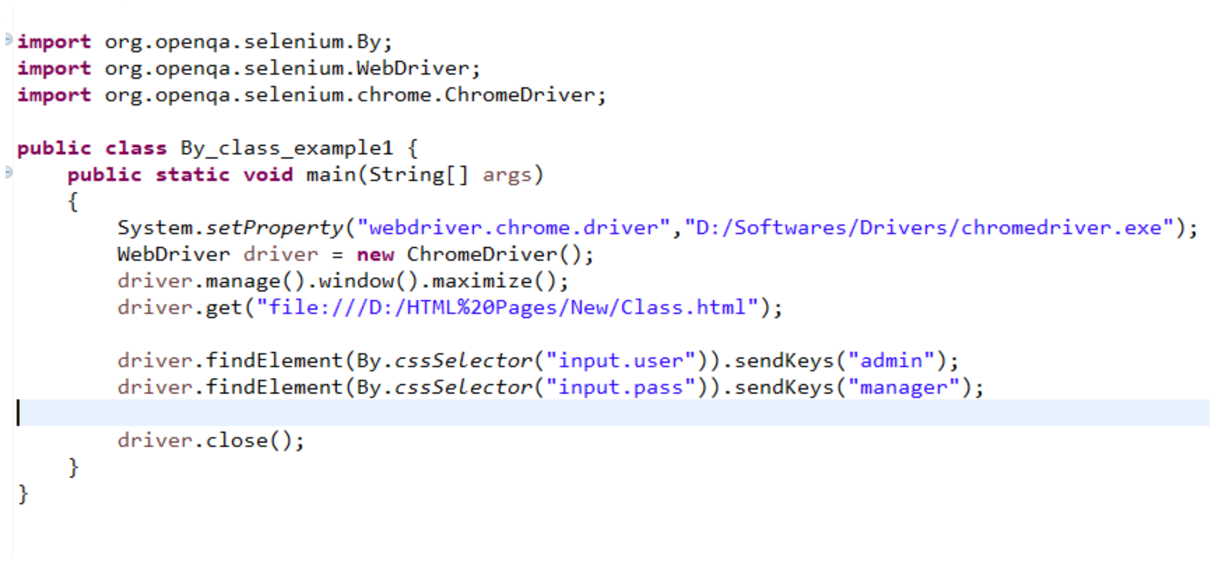
Syntax:

"TagName.ClassNamePropertyValue"



```
<html>
<head>
    <title>Qspiders</title>
</head>
<body style="color:blue">
    <center>
        <hr>
        Username <input type=" text" class="user"/><br> <hr>
        Password <input type=" password" class="pass"/><hr>
    </center>
</body>
</html>
```

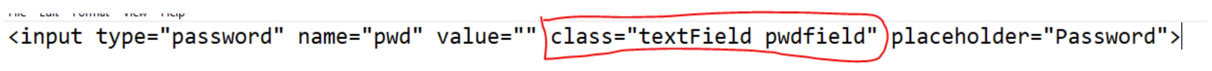
Example Program:



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

public class By_class_example1 {
    public static void main(String[] args)
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///D:/HTML%20Pages/New/Class.html");

        driver.findElement(By.cssSelector("input.user")).sendKeys("admin");
        driver.findElement(By.cssSelector("input.pass")).sendKeys("manager");

        driver.close();
    }
}
```

Note: If the class property value contains the multiple words and are separated by the space, than we need to remove the space by adding the dot between the words.

Example:



```
<input type="password" name="pwd" value="" class="textField pwdfield" placeholder="Password">
```

- In the above example, class property contains a pair of words, which is separated by space. In this case, we need to remove the space by adding the (.) dot between the words.

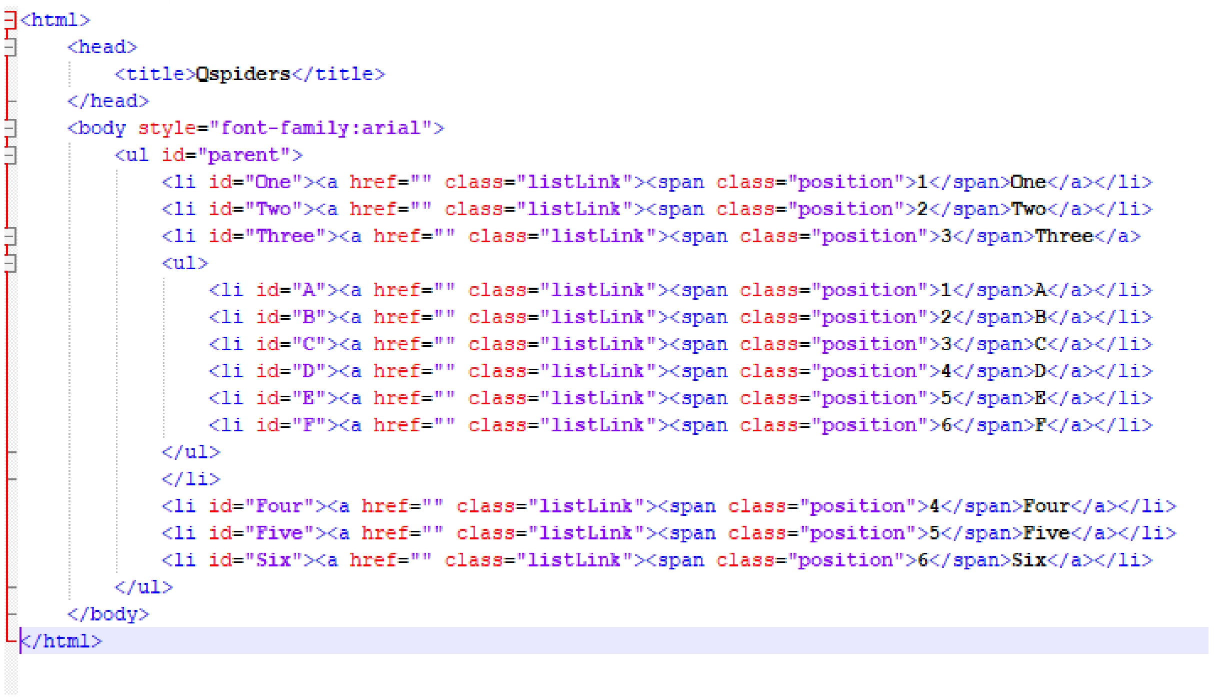
  Example: input.textField.pwdfield

4. Child CSS selector

- It will only select list items that are direct children of an unordered list. In other words, it only looks one level down the HTML structure, no deeper. So if there was another unordered list nested deeper, the list item children of it will not be targeted by this selector.
- A child Selector in CSS is the "greater than" symbol

Syntax:

“Parent Expression > child tag name/expression”



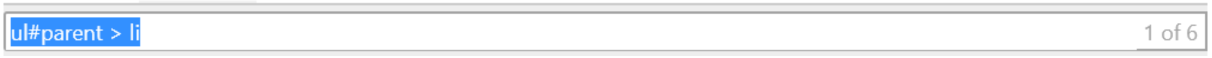
```
<html>
    <head>
        <title>Qspiders</title>
    </head>
    <body style="font-family:arial">
        <ul id="parent">
            <li id="One"><a href="" class="listLink"><span class="position">1</span>One</a></li>
            <li id="Two"><a href="" class="listLink"><span class="position">2</span>Two</a></li>
            <li id="Three"><a href="" class="listLink"><span class="position">3</span>Three</a>
            <ul>
                <li id="A"><a href="" class="listLink"><span class="position">1</span>A</a></li>
                <li id="B"><a href="" class="listLink"><span class="position">2</span>B</a></li>
                <li id="C"><a href="" class="listLink"><span class="position">3</span>C</a></li>
                <li id="D"><a href="" class="listLink"><span class="position">4</span>D</a></li>
                <li id="E"><a href="" class="listLink"><span class="position">5</span>E</a></li>
                <li id="F"><a href="" class="listLink"><span class="position">6</span>F</a></li>
            </ul>
            </li>
            <li id="Four"><a href="" class="listLink"><span class="position">4</span>Four</a></li>
            <li id="Five"><a href="" class="listLink"><span class="position">5</span>Five</a></li>
            <li id="Six"><a href="" class="listLink"><span class="position">6</span>Six</a></li>
        </ul>
    </body>
</html>
```

- The above nested list is a perfect example of why this selector is useful.
- Consider, you want to identify only “li” elements, which are direct children of “ul” element. Than we use CSS by child concept to identify only the direct children.

Example expression:

ul#parent > li

- The above expression will match 6 li elements, which are direct children of ul element.



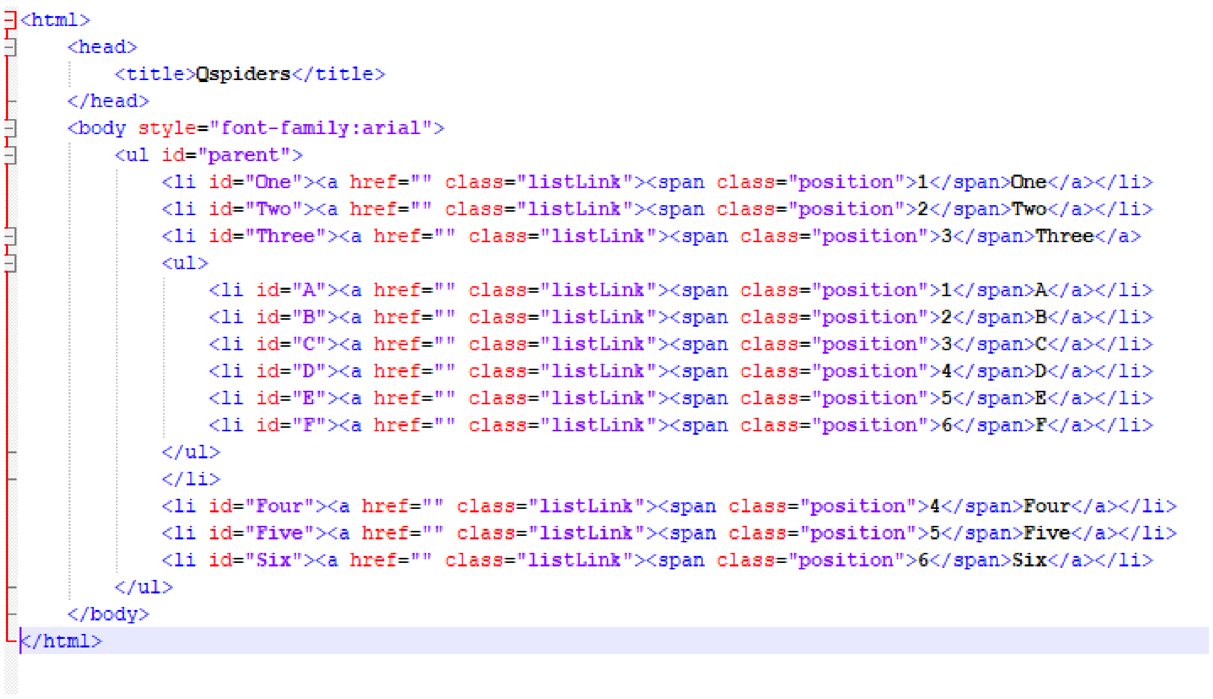
ul#parent > li 1 of 6

5. CSS by any child or descendant

- **Descendant selector** in CSS which matches all child elements that are descendants of the parent element. Descendant selector including elements that are not only direct descendants. These element may be a child, grandchild, great grandchild, and so on.
- A descendant selector is made up of two or more selectors separated by white space.

Syntax:

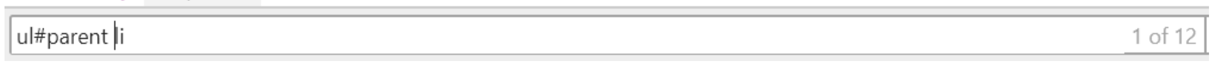
“Parent expression child element expression/TagName”



```
<html>
    <head>
        <title>Qspiders</title>
    </head>
    <body style="font-family:arial">
        <ul id="parent">
            <li id="One"><a href="" class="listLink"><span class="position">1</span>One</a></li>
            <li id="Two"><a href="" class="listLink"><span class="position">2</span>Two</a></li>
            <li id="Three"><a href="" class="listLink"><span class="position">3</span>Three</a>
            <ul>
                <li id="A"><a href="" class="listLink"><span class="position">1</span>A</a></li>
                <li id="B"><a href="" class="listLink"><span class="position">2</span>B</a></li>
                <li id="C"><a href="" class="listLink"><span class="position">3</span>C</a></li>
                <li id="D"><a href="" class="listLink"><span class="position">4</span>D</a></li>
                <li id="E"><a href="" class="listLink"><span class="position">5</span>E</a></li>
                <li id="F"><a href="" class="listLink"><span class="position">6</span>F</a></li>
            </ul>
            </li>
            <li id="Four"><a href="" class="listLink"><span class="position">4</span>Four</a></li>
            <li id="Five"><a href="" class="listLink"><span class="position">5</span>Five</a></li>
            <li id="Six"><a href="" class="listLink"><span class="position">6</span>Six</a></li>
        </ul>
    </body>
</html>
```

Example:

ul#parent li



ul#parent li   1 of 12

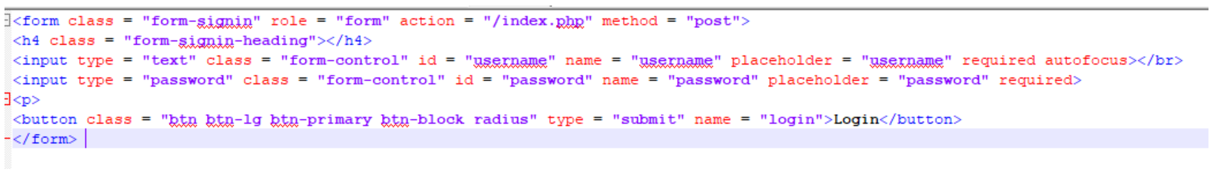
The above code tells the browser to select all “li” elements that are descendants of “ul” elements to apply the rule.

6. Adjacent Sibling Selector (CSS Next Sibling Selector)

- The **Adjacent Sibling Selector** selects an element`s next sibling, that is it allows you to select an element that is directly after another specific element.
- The special character used in Adjacent Sibling selector is the + (plus) character.

Syntax:

“Element + adjacent element”

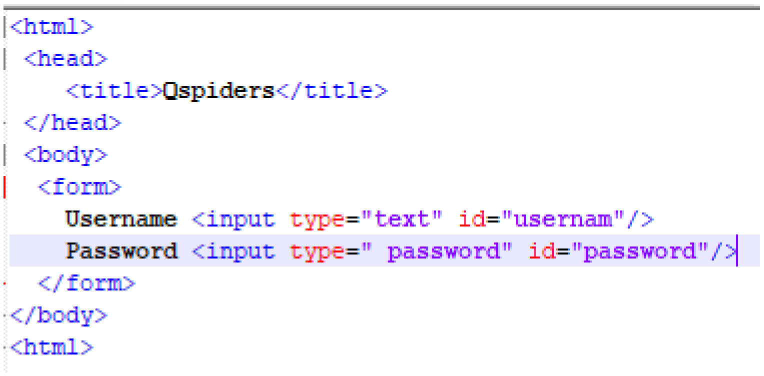


```
<form class = "form-signin" role = "form" action = "/index.php" method = "post">
<h4 class = "form-signin-heading"></h4>
<input type = "text" class = "form-control" id = "username" name = "username" placeholder = "username" required autofocus></br>
<input type = "password" class = "form-control" id = "password" name = "password" placeholder = "password" required>
<p>
<button class = "btn btn-lg btn-primary btn-block radius" type = "submit" name = "login">Login</button>
</form>
```

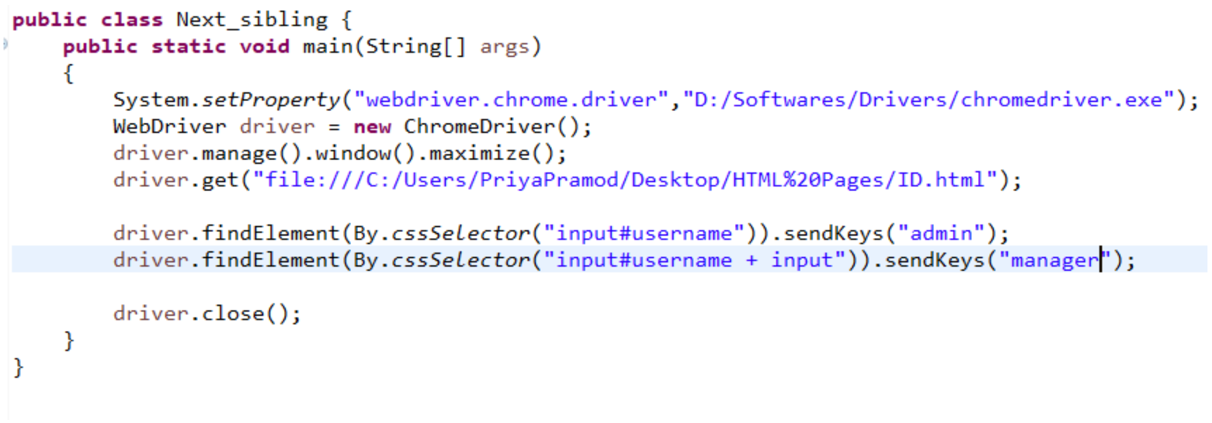
Example 1:

```
.username + input
```

Example 2:



```
<html>
 <head>
    <title>Qspiders</title>
 </head>
 <body>
  <form>
    Username <input type="text" id="usernam"/>
    Password <input type=" password" id="password"/>
  </form>
</body>
<html>
```



```
public class Next_sibling {
    public static void main(String[] args)
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///C:/Users/PriyaPramod/Desktop/HTML%20Pages/ID.html");

        driver.findElement(By.cssSelector("input#username")).sendKeys("admin");
        driver.findElement(By.cssSelector("input#username + input")).sendKeys("manager");

        driver.close();
    }
}
```

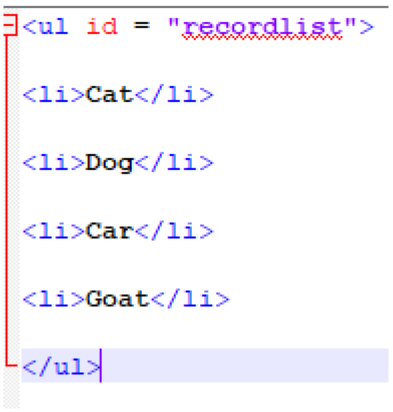
7. **CSS by Specific match**

- CSS selectors in Selenium allow us to navigate lists with more finesse that the above methods. If we have ‘a’ & ‘ul’ and we want to select its fourth li element without regard to any other elements.

CSS supports following methods to navigate lists

- **Nth-of-type**
- **Nth-child**
- **First-child**
- **Last-child**

Let’s take a simple html tree.

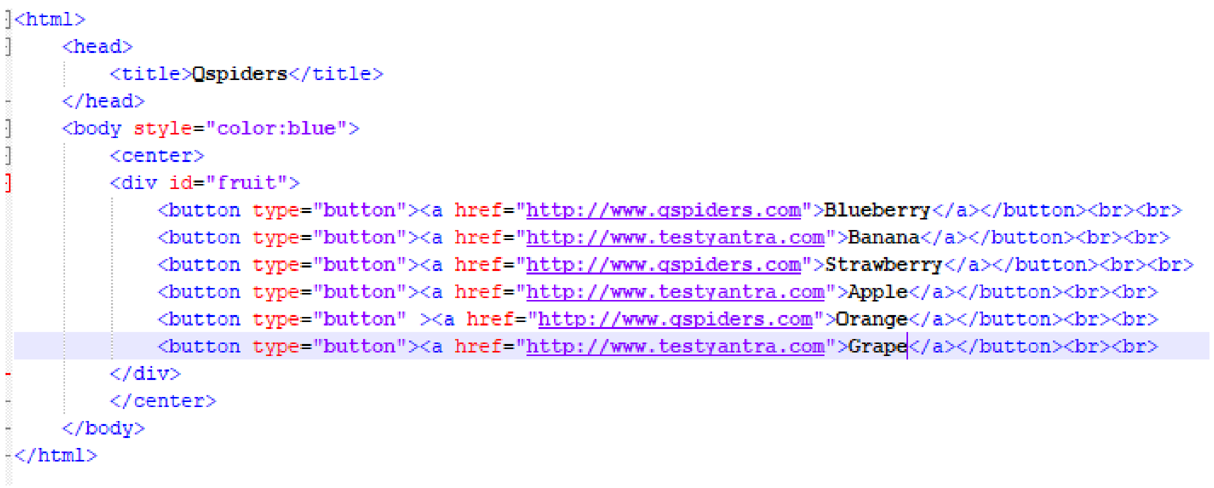


```
<ul id = "recordlist">

<li>Cat</li>

<li>Dog</li>

<li>Car</li>

<li>Goat</li>

</ul>
```

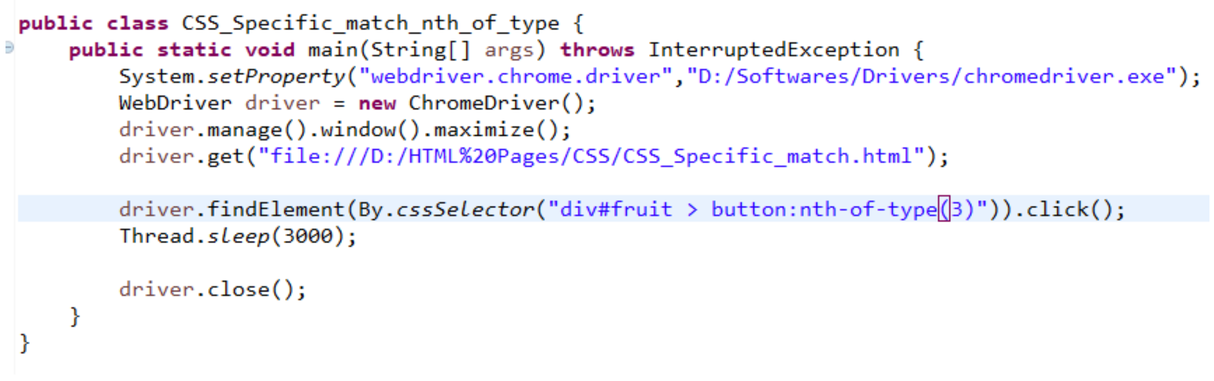
Nth-of-type

If we want to select the fourth li element (Goat) in this list, we can use the nth-of-type, which will find the fourth li in the list.

```
#recordlist li:nth-of-type(4)
```



```
<html>
    <head>
        <title>Qspiders</title>
    </head>
    <body style="color:blue">
        <center>
        <div id="fruit">
            <button type="button"><a href="http://www.qspiders.com">Blueberry</a></button><br><br>
            <button type="button"><a href="http://www.testyantra.com">Banana</a></button><br><br>
            <button type="button"><a href="http://www.qspiders.com">Strawberry</a></button><br><br>
            <button type="button"><a href="http://www.testyantra.com">Apple</a></button><br><br>
            <button type="button" ><a href="http://www.qspiders.com">Orange</a></button><br><br>
            <button type="button"><a href="http://www.testyantra.com">Grape</a></button><br><br>
        </div>
        </center>
    </body>
</html>
```



```
public class CSS_Specific_match_nth_of_type {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///D:/HTML%20Pages/CSS/CSS_Specific_match.html");

        driver.findElement(By.cssSelector("div#fruit > button:nth-of-type(3)")).click();
        Thread.sleep(3000);

        driver.close();
    }
}
```

Nth-Child

On the other hand, if we want to get the fourth element only if it is a li element, we can use a filtered nth-child which will select (Car) in this case.

```
#recordlist li:nth-child(4)
```

```
<html><head>
   <title>Qspiders</title>
</head>
  <body style="font-family:arial">
    <ul id="parent">
      <li id="One"><a href="" class="listLink"><span class="position">1</span>One</a></li>
      <li id="Two"><a href="" class="listLink"><span class="position">2</span>Two</a></li>
      <li id="Three"><a href="" class="listLink"><span class="position">3</span>Three</a>
        <ul>
            <li id="A"><a href="" class="listLink"><span class="position">1</span>A</a></li>
            <li id="B"><a href="" class="listLink"><span class="position">2</span>B</a></li>
            <li id="C"><a href="" class="listLink"><span class="position">3</span>C</a></li>
            <li id="D"><a href="" class="listLink"><span class="position">4</span>D</a></li>
            <li id="E"><a href="" class="listLink"><span class="position">5</span>E</a></li>
            <li id="F"><a href="" class="listLink"><span class="position">6</span>F</a></li>
        </ul>
    </li>
    <li id="Four"><a href="" class="listLink"><span class="position">4</span>Four</a></li>
    <li id="Five"><a href="" class="listLink"><span class="position">5</span>Five</a></li>
    <li id="Six"><a href="" class="listLink"><span class="position">6</span>Six</a></li>
    </ul>
    </body>
</html>
```

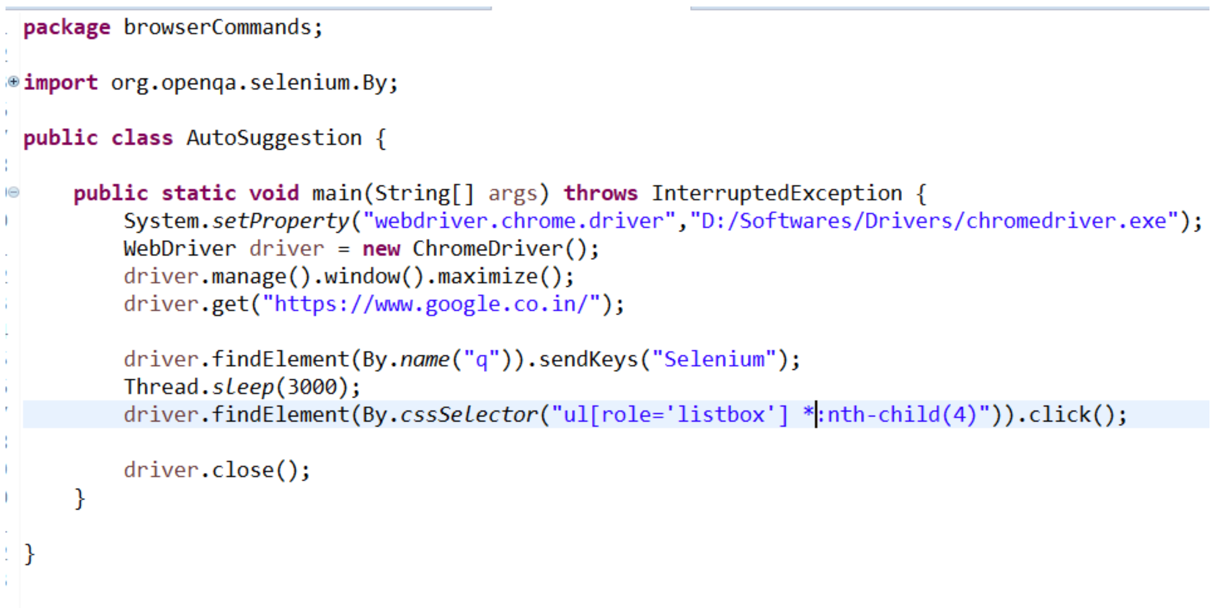
```java
public class CSS_Specific_match_nth_child {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///D:/HTML%20Pages/CSS/Demo.html");

        driver.findElement(By.cssSelector("ul#parent > li:nth-child(3)")).click();
        Thread.sleep(3000);

        driver.close();
    }
}
```

- **Example:**

```java
package browserCommands;

import org.openqa.selenium.By;

public class AutoSuggestion {

    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.google.co.in/");

        driver.findElement(By.name("q")).sendKeys("Selenium");
        Thread.sleep(3000);
        driver.findElement(By.cssSelector("ul[role='listbox'] li:nth-child(4)")).click();

        driver.close();
    }

}
```

Note, if you don't specify a child type for nth-child it will allow you to select the fourth child without regard to type. This may be useful in testing CSS layout in selenium.

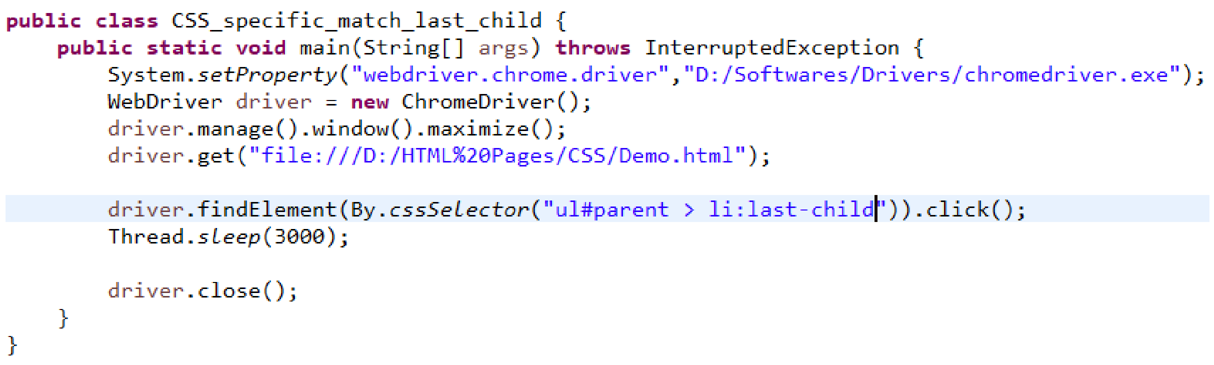
```
#recordlist *:nth-child(4)
```



```
package browserCommands;

import org.openqa.selenium.By;

public class AutoSuggestion {

    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://www.google.co.in/");

        driver.findElement(By.name("q")).sendKeys("Selenium");
        Thread.sleep(3000);
        driver.findElement(By.cssSelector("ul[role='listbox'] *:nth-child(4)")).click();

        driver.close();
    }

}
```

Last-child

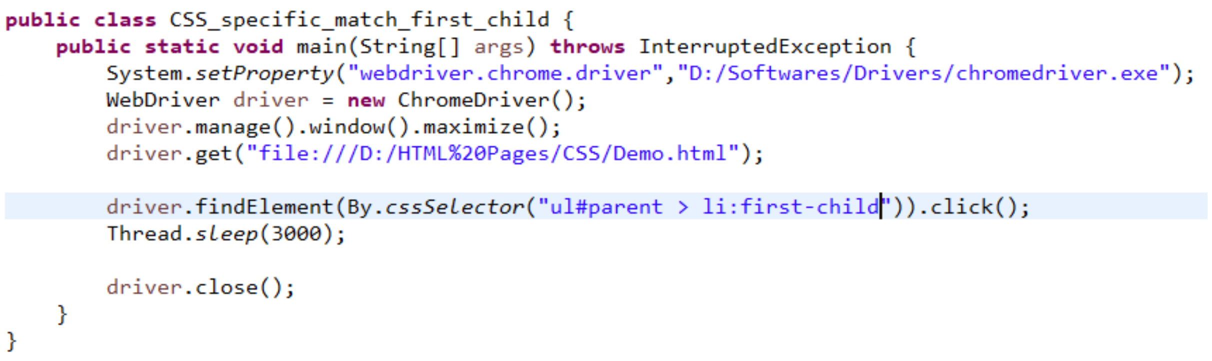
- Last child is the special function in the CSS for selecting the last element from the list.



```
public class CSS_specific_match_last_child {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///D:/HTML%20Pages/CSS/Demo.html");

        driver.findElement(By.cssSelector("ul#parent > li:last-child")).click();
        Thread.sleep(3000);

        driver.close();
    }
}
```

First-Child

- CSS selector provides another method to select the element from the list called as first-child. As the name suggest this function is used to select the first child from the list.



```java
public class CSS_specific_match_first_child {
    public static void main(String[] args) throws InterruptedException {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("file:///D:/HTML%20Pages/CSS/Demo.html");

        driver.findElement(By.cssSelector("ul#parent > li:first-child")).click();
        Thread.sleep(3000);

        driver.close();
    }
}
```

8. CSS by Sub-String Matches

CSS in Selenium has an interesting feature of allowing partial string matches.

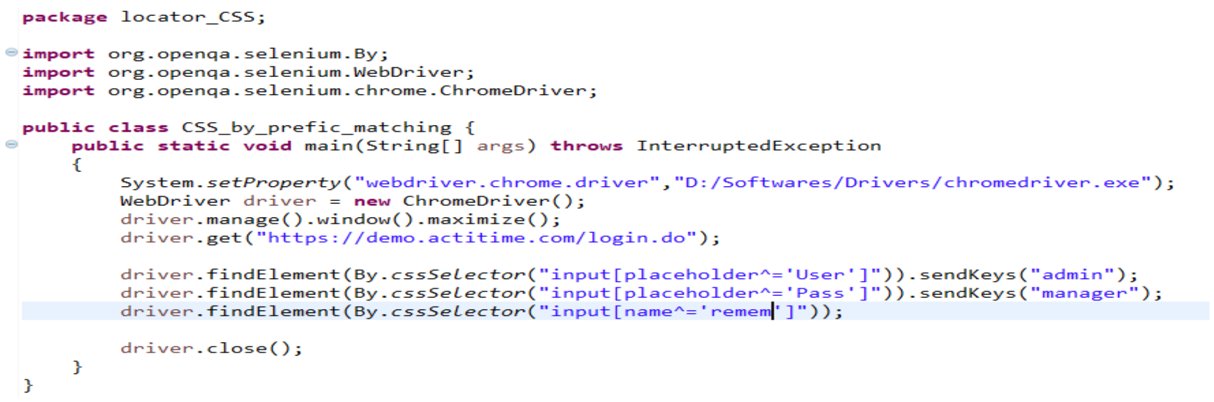
Following are the special symbols used for partial string match.

^= Match a prefix

$= Match a suffix

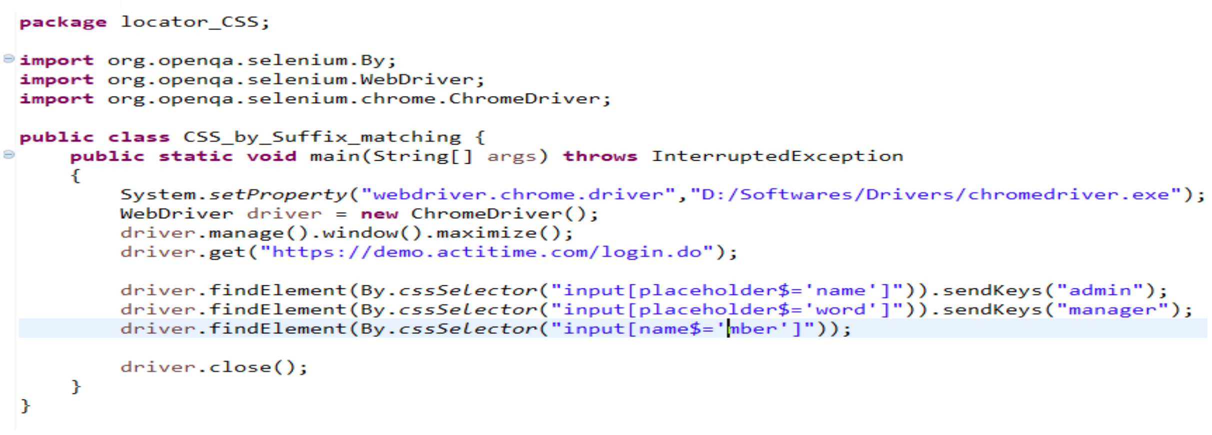
*= Match a substring

Example program for prefix matching



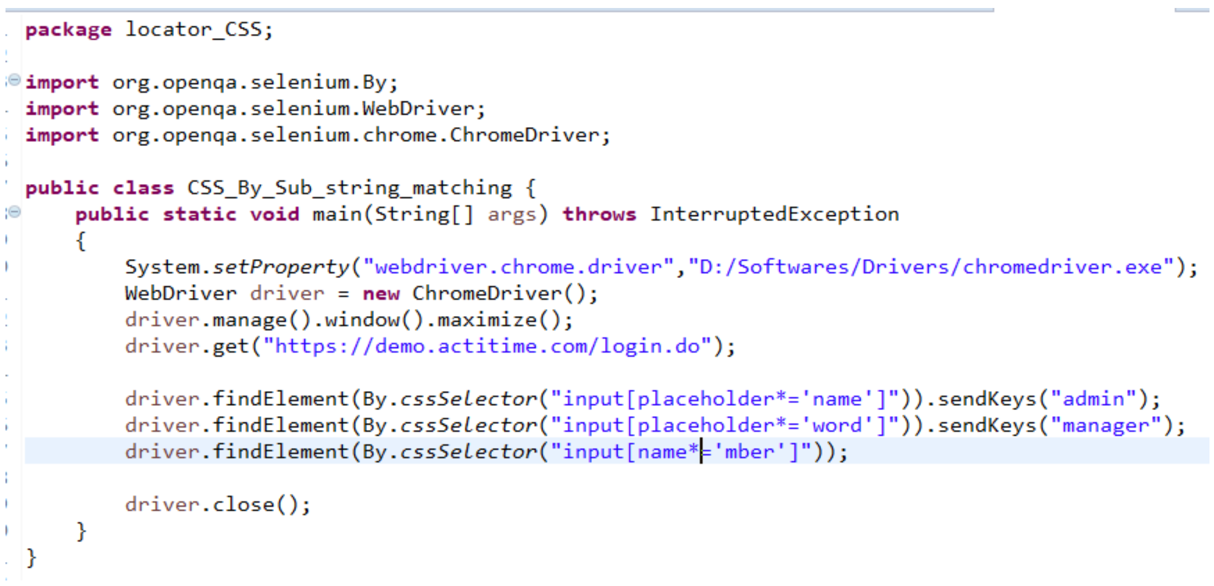
```java
package locator_CSS;

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

public class CSS_by_prefic_matching {
    public static void main(String[] args) throws InterruptedException
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/login.do");

        driver.findElement(By.cssSelector("input[placeholder^='User']")).sendKeys("admin");
        driver.findElement(By.cssSelector("input[placeholder^='Pass']")).sendKeys("manager");
        driver.findElement(By.cssSelector("input[name^='remem']"));

        driver.close();
    }
}
```

Example program for suffix matching



```java
package locator_CSS;

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

public class CSS_by_Suffix_matching {
    public static void main(String[] args) throws InterruptedException
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/login.do");

        driver.findElement(By.cssSelector("input[placeholder$='name']")).sendKeys("admin");
        driver.findElement(By.cssSelector("input[placeholder$='word']")).sendKeys("manager");
        driver.findElement(By.cssSelector("input[name$='mber']"));

        driver.close();
    }
}
```

Example program for sub string matching



```java
package locator_CSS;

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

public class CSS_By_Sub_string_matching {
    public static void main(String[] args) throws InterruptedException
    {
        System.setProperty("webdriver.chrome.driver","D:/Softwares/Drivers/chromedriver.exe");
        WebDriver driver = new ChromeDriver();
        driver.manage().window().maximize();
        driver.get("https://demo.actitime.com/login.do");

        driver.findElement(By.cssSelector("input[placeholder*='name']")).sendKeys("admin");
        driver.findElement(By.cssSelector("input[placeholder*='word']")).sendKeys("manager");
        driver.findElement(By.cssSelector("input[name*='mber']"));

        driver.close();
    }
}
```

Drawback of CSS

- Backward traversing is not possible.
- Identifying the elements using the visible text is not possible using the CSS selector.

Why CSS

- It is easier to learn/understand
- It can do almost everything XPath can
- Typically faster than Xpath