**What is Class?**

A Class is a group of object which have common properties. It is a template or blueprint from which objects are created. It is a logical entity. It can’t be physical.

A class in java can contain

* Field
* Methods
* Constructor
* Blocks
* Nested class and Interface

**What is Object?**

* An object is a real-world entity.
* An object is a runtime entity.
* The object is an entity which has state and behavior
* The object is an Instance of a class

What are the Data types available in java?

What is the Advantage of Method?

* Code Reusability
* Code Optimization

What is Method in java?

A **method** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation. It is used to achieve the **reusability** of code. We write a method once and use it many times. We do not require to write code again and again. It also provides the **easy modification** and **readability** of code, just by adding or removing a chunk of code. The method is executed only when we call or invoke it.

**What is Access Modifiers or Access Specifier? And What are the Access Specifier available in java?**

**Access Specifier:** Access specifier or modifier is the access type of the method. It specifies the visibility of the method. Java provides **four** types of access specifier:

* **Public:** The method is accessible by all classes when we use public specifier in our application.
* **Private:** When we use a private access specifier, the method is accessible only in the classes in which it is defined.
* **Protected:** When we use protected access specifier, the method is accessible within the same package or subclasses in a different package.
* **Default:** When we do not use any access specifier in the method declaration, Java uses default access specifier by default. It is visible only from the same package only.

|  |
| --- |
| Element Validation |

All the below methods return Boolean value

isEnabled()- return true if the element is enabled else return false.

isDisplayed()- return true if the element is displayed else return false.

isSelected()-return true if the element is selected else return false.

|  |
| --- |
| Handle Alert |

driver.switchTo().alert().getText()- To read the text from the alert

driver.switchTo().alert().accept()- To Accept the alert

driver.switchTo().alert().dismiss()- To cancel the alert

driver.switchTo().alert().sendKeys()- To pass the values in alert text box

Exceptions:

NoAlertPresentException- Trying to handle an alert which is not present

UnhandledAlertException- Trying to access main page without handling an alert

|  |
| --- |
| Selenium Navigators |

**Navigate to url :**

driver.get(String url);

driver.navigate().to( String url);

**Refresh page :**

driver.navigate().refresh();

**Navigate forwards in browser history :**

driver.navigate().forward();

**Navigate backward in browser history :**

driver.navigate().back();

|  |
| --- |
| Get Methods In Selenium |

getTitle() –To retrieve the title of the current webpage.

getCurrrentUrl() –To retrieve the url of the current webpage

getPageSource() –To retrieve the source of the webpage

getText() –To retrieve the text in the WebElement

getAttribute(String attributeName) –To retrieve the value of the attribute that we passed

|  |
| --- |
| Frames |

driver.switchTo().frame(int index); - switch to frame using index

driver.switchTo().frame(String nameOrId); - switch to frame using name or id

driver.switchTo().frame(WebElement frameElement); - switch to frame using WebElement of frame

parentFrame() – Applicable for nested frames. Control will move from child frame to immediate parent frame.

driver.switchTo().defaultContent() – Move the control from frame to the main html DOM.

**Exceptions:**

NoSuchFrameException - When the given frame (using index / id / name) is not available in the DOM

|  |
| --- |
| Window Handling |

getWindowHandle() – To get the reference for the current window

getWindowHandles()- To get the references for all the windows opened by WebDriver

driver.close() – To close the current window.

driver.quit() –To close all the windows opened by WebDriver.

**Exception:**

**NoSuchWindowException** - When the window handle does not exist

|  |
| --- |
| Mouse Actions |

Click() –Click the current mouse location.

ClickAndHold() - Clicks at the present mouse location (without releasing)

|  |  |  |
| --- | --- | --- |
|  | contextClick() -Performs a context-click (right click) at the current mouse location.  doubleClick() –Performs a double-click at the current mouse location.  dragAndDrop(WebElement source,WebElement target) - Invokes click-and-hold at the source location and moves to the location of the target element before releasing the mouse. source – element to grab, target – element to release  movetoElement() - Moves the mouse to the middle of the element.  dragAndDropBy(WebElement source,int xOffset,int yOffset) - Invokes click-and-hold at the source location and moves to  the location and release at the given offset  release() -Releases the depressed left mouse button at the current mouse location   |  | | --- | | TakeScreenShot |   A Screenshot in Selenium Webdriver is used for bug analysis.  **Syntax:**  File src = driver.getScreenshotAs(OutputType.FILE);  FileUtils.copyFile(scr,new File(path)); |

|  |
| --- |
| Keyboard Actions |

**sendKeys():** Use this method to simulate typing into an element, which may set its value.

**keyDown():** Sends a key press without release it. Subsequent actions may assume it as pressed. (example: Keys.ALT, Keys.SHIFT, or Keys.CONTROL)

**keyUp():** Performs a key release

|  |
| --- |
| Implicitly Wait |

This wait can be considered as element detection timeout. Once defined in a script, this wait will be set for all the Web Elements on a page. Selenium keeps polling to check whether that element is available to interact with or not.

Syntax:

**driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(20));**

|  |
| --- |
| Explicitly Wait |

This wait can be considered as conditional wait, and is applied to a particular Web Element with a condition.

**Syntax:**

**WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(20));**

|  |
| --- |
| findElement |

Returns a single **WebElement**

**Syntax:**

**WebElement findElement(By.locator())**

**Exception:**

NoSuchElementException

|  |
| --- |
| findElements |

Does not throw any exception

Returns an **Empty List of WebElement Object**

**Syntax:**

**java.util.List< WebElement> findElements(By.locator())**