**1. NoSuchElementException**

* **Cause**: This exception occurs when the WebDriver is unable to find an element using the specified locator.
* **Solution**: Ensure that the element locator is correct and that the element is present in the DOM.

**2. TimeoutException**

* **Cause**: Raised when a command takes longer than the specified timeout period.
* **Solution**: Increase the timeout duration for explicit waits, or check if the conditions you are waiting for are correct.

**3. StaleElementReferenceException**

* **Cause**: This occurs when the WebDriver tries to interact with an element that is no longer attached to the DOM (e.g., the page has changed).
* **Solution**: Re-locate the element before interacting with it, or use explicit waits to ensure the element is stable before accessing it.

**4. ElementNotVisibleException**

* **Cause**: Raised when trying to interact with an element that is present in the DOM but not visible on the page.
* **Solution**: Ensure that the element is visible and not hidden by CSS or other elements.

**5. ElementNotInteractableException**

* **Cause**: This happens when an element is present in the DOM but cannot be interacted with (e.g., it is disabled).
* **Solution**: Verify that the element is enabled and visible before trying to interact with it.

**6. InvalidElementStateException**

* **Cause**: Raised when an element is in a state that does not allow interaction (e.g., trying to enter text into a read-only field).
* **Solution**: Check the state of the element and ensure it is in an appropriate state for the intended action.

**7. NoAlertPresentException**

* **Cause**: This occurs when attempting to switch to an alert that is not present.
* **Solution**: Ensure that the alert is indeed present before trying to switch to it.

**8. WebDriverException**

* **Cause**: A generic exception that can occur for various reasons related to the WebDriver, such as issues with the browser session.
* **Solution**: Check the error message for specifics and troubleshoot based on the context.

**9. JavaScriptException**

* **Cause**: Raised when a JavaScript error occurs while executing a script through Selenium.
* **Solution**: Review the JavaScript code you are trying to execute for errors.

**10. MoveTargetOutOfBoundsException**

* **Cause**: Occurs when an element cannot be scrolled into view for actions like clicking.
* **Solution**: Use JavaScript to scroll the element into view before attempting to interact with it.

11. abstract methods

locators

waits

// Set implicit wait

driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

// Set explicit

waitWebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));

// Wait until the element is clickable

wait.until(ExpectedConditions.elementToBeClickable(By.id("some\_id"))).click();

// Switch to the alert

Alert alert = driver.switchTo().alert();

* **alert.accept()**: Clicks the "OK" button on the alert.
* **alert.dismiss()**: Clicks the "Cancel" button on the alert.
* **alert.getText()**: Retrieves the text displayed in the alert.
* **alert.sendKeys(String text)**: Sends input text to a prompt alert.
* //1. Through Typecasting  
  // TakesScreenshot ts = (TakesScreenshot)driver;  
  // File ss = ts.getScreenshotAs(OutputType.FILE);  
    
  Select sel=new Select(ddAddr);//Address of Dropdown  
  List<WebElement> allOptions = sel.getOptions();  
  System.out.println("Total Options in DD: "+allOptions.size());  
    
  1. Handling mouse hover action - moveToElement(addr).perform()  
  2. a) Perform Right Click - contextClick(addr).perform()  
  b) Handle Right Clicked Options - Robot Class - keyPress/KeyRelease  
  3. Perform Double Click - doubleClick(addr).perform()  
  4. Perform Drag and Drop action - dragAndDrop(fromAddr, toAddr).perform()  
    
  // Create an instance of   
  JavascriptExecutorJavascriptExecutor js = (JavascriptExecutor) driver;               
  // Scroll to the specific element             
   js.executeScript("arguments[0].scrollIntoView(true);", elementToScrollTo);