**Selenium WebDriver Exceptions**

**1. WebDriverException:**

* **Cause:** This is the base class for most Selenium exceptions. It can occur due to various issues, such as problems starting the browser driver, network issues, or errors in the WebDriver implementation itself.
* **Solution:** Identify the specific reason by checking the underlying cause (e.g., wrapped exceptions). You might need to:
  + Check your WebDriver configuration (e.g., correct path to the browser driver executable).
  + Address network connectivity issues.
  + Refer to the Selenium documentation for specific error messages.

**2. TimeoutException:**

* **Cause:** Occurs when an operation (e.g., waiting for an element) exceeds the predefined time limit.
* **Solution:**
  + Increase the implicit wait timeout using driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10)).
  + Use explicit waits with WebDriverWait and expected conditions to wait for specific element states (e.g., visibilityOf).
  + Optimize your web application's performance to reduce waiting times.

**3. ElementClickInterceptedException:**

* **Cause:** Thrown when an element you're trying to click is obscured by another element (like a modal dialog).
* **Solution:**
  + Handle the obstructing element (e.g., close a modal).
  + Use JavaScript execution to click the element directly if necessary:

WebElement element = driver.findElement(By.id("myElement"));

((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);

**4. ElementNotInteractableException:**

* **Cause:** Occurs when the element is present but not in a clickable state (e.g., disabled, hidden behind another element).
* **Solution:**
  + Check for element visibility and enabled state before clicking.
  + Handle any conditions that might disable the element.
  + Consider using element.isEnabled() and element.isDisplayed() before interacting.

**5. ElementNotVisibleException:**

* **Cause:** Thrown when an element is present in the DOM but not visible (e.g., hidden by CSS).
* **Solution:**
  + Use explicit waits with visibilityOf condition to wait for the element to become visible.
  + Check for element visibility using element.isDisplayed() before interaction.
  + If the element is dynamically loaded, use JavaScript or waiting mechanisms to handle its appearance.

**6. InvalidSelectorException:**

* **Cause:** Occurs when the locator used to find an element is invalid or doesn't match any element in the DOM.
* **Solution:**
  + Double-check the locator syntax for the specific element type (ID, name, CSS selector, XPath).
  + Use developer tools in your browser to inspect elements and confirm selectors.
  + Consider using more robust locators that are less likely to become invalid due to DOM changes.

**7. MoveTargetOutOfBoundsException:**

* **Cause:** Occurs when you try to move a mouse cursor to a location outside the browser window or viewport.
* **Solution:**
  + Ensure the target coordinates are within the visible area of the browser window.
  + Adjust your mouse movement logic to stay within the browser bounds.

**8. NoAlertPresentException:**

* **Cause:** Thrown when you try to switch to an alert that is not present on the page.
* **Solution:**
  + Verify that an alert is actually displayed before using driver.switchTo().alert().
  + Use conditional checks or waits to ensure the alert exists before switching.

**9. NoSuchElementException:**

* **Cause:** Occurs when the element you're trying to interact with is not found in the DOM. Reasons might include incorrect locator, dynamic elements that haven't loaded yet, or elements removed from the page.
* **Solution:**
  + Review your locators for accuracy and robustness.
  + Use explicit waits with presenceOfElementLocated to ensure the element is present before interaction.
  + Consider handling dynamic elements using JavaScript or waiting mechanisms.

**10. NoSuchFrameException:**

* **Cause:** Thrown when you try to switch to a frame that doesn't exist or is not currently available.
* **Solution:**
  + Verify that the frame exists in the DOM before switching using driver.findElements(By.tagName("iframe")).
  + Use correct frame identification methods (e.g., driver.switchTo().frame(frameElement)) or frame index if applicable.

**11. NoSuchWindowException:**

* **Cause:** Thrown when you try to switch to a window that doesn't exist or is not currently open.
* **Solution:**
  + Ensure the target window is open before switching.
  + Use methods like driver.getWindowHandles() to get a list of available windows and switch using their handles.
  + If windows are dynamically created, implement logic to identify and switch to the correct one.

**12. SessionNotCreatedException:**

* **Cause:** Occurs when the WebDriver session cannot be created due to issues starting the browser driver or problems with the browser itself.
* **Solution:**
  + Verify your WebDriver configuration (correct path to the browser driver executable).
  + Check for compatibility between your WebDriver version and browser version.
  + Ensure the browser is installed and accessible.
  + Address any browser-related errors in the console logs.

**13. StaleElementReferenceException:**

* **Cause:** Thrown when you try to interact with an element that has become stale (the DOM has changed since the element was found). This can happen due to page reloads, dynamic content updates, or using stale references.
* **Solution:**
  + Refresh the page using driver.navigate().refresh() if necessary.
  + Refetch the element using your locators before interaction.
  + Consider using WebDriverWait with expected conditions like presenceOfElementLocated to ensure the element is fresh before interaction.

**14. InsecureCertificateException**

* **Cause:** Occurs when you try to access a website with an invalid or untrusted SSL certificate.
* **Solution:**
  + **For development environments only:** You can use DesiredCapabilities to accept insecure certificates, but **never do this in production** as it exposes security risks.
  + **For production environments:** Fix the website's SSL certificate or use a trusted certificate authority.

**15. InvalidArgumentException:**

* **Cause:** This is a general exception that can occur for various reasons, often due to invalid arguments passed to WebDriver methods.
* **Solution:**
  + Carefully review the documentation for the specific WebDriver method you're using to ensure you're passing valid arguments.
  + Check the error message for clues about the specific invalid argument.
  + Double-check variable values and expected data types for the arguments.