What is Exception?

An exception is known as an unusual or unprecedented event that occurs during the execution of a software program or application. It is a runtime error of an unexpected result or event which influence and disrupt usual program flow. An exception is also considered as a fault.

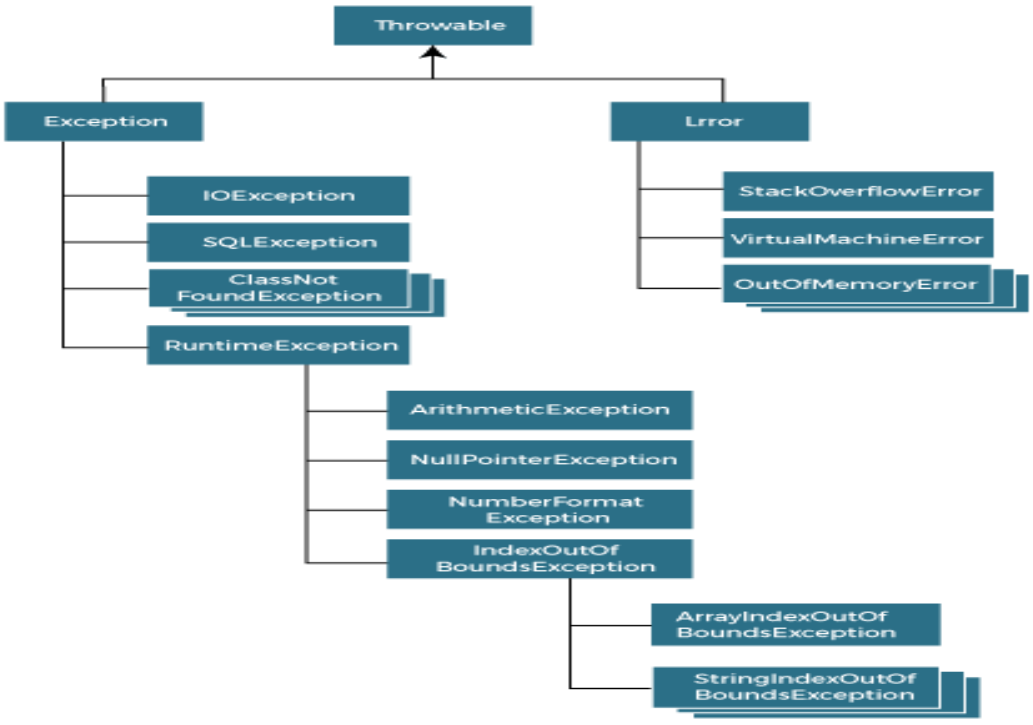
Types of Exception

1. Checked Exceptions

Checked Exceptions are handled during the process of writing codes. These exceptions are handled before compiling the code, therefore, such exceptions are examined at the compile time.

2. Unchecked Exceptions

These exceptions are thrown at runtime. Unchecked exceptions are more catastrophic than the compile-time exception as it causes problems while running Automation pack in headless.



Rules to use Try-Catch block to handle the exception

Keyword	Description
try	The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
catch	The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
finally	The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
throw	The "throw" keyword is used to throw an exception.
throws	The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.

Common Exceptions in Selenium:

NoSuchElementException:

- Happens when the locators are unable to find or access elements on the web page or application. Ideally, the exception occurs due to the use of incorrect element locators in the findElement(By.locator()) method

How to handle the Exception

To handle this exception, use the wait command. Use Try/Catch to ensure that the program flow is interrupted if the wait command doesn't help.

StaleElementReferenceException

- This exception occurs when the web element is no longer part of the web page.
- The element may have been part of the source code, but it is no longer part of the window.
- There can be multiple reasons for this exception. It can occur either from a switch to a different page, the element is no longer part of DOM or due to frame/window switch
- When page get refreshed when searching/applying filter in webpage, the element is no longer part of DOM due to loading of web elements in the page

How to handle the Exception:

- To handle this exception, use the wait command. Use Try/Catch to ensure that the program flow is interrupted if the wait command doesn't help.

InvalidSelectorException;

- This exception occurs due to an incorrect selector. More often than not, the selector syntax is wrong.

How to handle the Exception:

- To avoid this exception, first, check the locator syntax. If it is incorrect, make sure the locator is syntactically correct.

TimeOut Exception:

- This exception is thrown if the command did not execute or complete within wait time. As already mentioned, waits are used to avoid NoSuchElementException. However, TimeoutException will be thrown after the page components fail to load within wait time.

How to handle the Exception:

- Avoiding this exception is simple. All one needs to do is to calculate the average page load time and adjust the wait time accordingly.

NoSuchWindowException:

- One of the most frequent exceptions in Selenium Webdriver, NoSuchWindowException occurs if the current list of windows is not updated. The previous window does not exist, and you can't switch to it.

How to handle the Exception:

- To handle this exception, use webdriver methods called "driver.getWindowHandles() and confirm the available opened windows for the switches"

NoSuchFrameException:

- Similar to NoSuchWindowException, the NoSuchFrameException occurs when switching between multiple frames is not possible.

How to handle the Exception:

- Check for the available frames in the webpage

NoAlertPresentException:

- Happens when the user is trying to you switch to an alert which is not present. In simple terms, it means that the test is too quick and is trying to find an alert that has not yet been opened by the browser.

How to handle the Exception:

- To handle or simply avoid this exception, use explicit or fluent wait in all events where an alert is expected.

InvalidSelectorException:

- This exception occurs due to an incorrect selector. More often than not, the selector syntax is wrong.

How to handle the Exception:

- To avoid this exception, first, check the locator syntax. If it is incorrect, make sure the locator is syntactically correct.

ElementNotVisibleException:

- This exception occurs when the WebDriver tries to find an element that is hidden or invisible.

How to handle the Exception:

- To handle this exception, it is essential that the exact reason is identified, which can be due to nested web elements or overlapping web elements.
- In the first case, you have to locate the specific element. In the second case, use explicit wait.

ElementNotSelectableException:

- This exception belongs to the same class as InvalidElementStateException. In such exceptions, the element is present on the web page, but the element cannot be selected

How to handle the Exception:

- To handle this exception, the wait command must be used.

NoSuchSessionException:

- As the name suggests, the exception is thrown if a method is called post the browser is closed. Other reasons for this exception include a browser crash.

How to handle the Exception:

- To avoid this handle, ensure that your browser is updated and stable.

ElementClickInterceptedException:

- The command could not be completed as the element receiving the events is concealing the element which was requested clicked.

How to handle the Exception

- To handle this exception, use the wait command. The other way is to use Actions class to click the element directly or the javaScript click to click the required web element

ElementNotInteractableException:

- This Selenium exception is thrown when an element is presented in the DOM but it is impossible to interact with such element.

How to handle the Exception

- To handle this exception, use the wait command. The other way is to use Actions class to click the element directly or the java Script click to click the required web element