Exception Information displaying methods are:

1.printStackTrace(): prints the stack trace , exception name and description.

2.toString(): returns a text message describing the exception name and description.

3.getMessage(): displays the description of exception

Unchecked Exceptions:

unchecked exceptions are not checked by compiler at the time of compilation. the exceptions which are extended by RuntimeException class are all unchecked exceptions. Some of the unchecked exceptions are AritmeticException, NullPointerException etc

In selenium we see unchecked exceptions such as NoSuchElementException, StaleElementReferenceException , NoSuchWindowException, TimeoutException etc.

NoSuchElementException - Thrown by WebDriver.findElement(By by) and WebElement.findElement(By by) - When ever the element is not found in DOM, you will get this exception.

StaleElementReferenceException - This Exception occurs when driver is trying to perform action on the element which is no longer exists or not valid. You can here for more alternate ways of Handling StaleElementReferenceException

NoSuchWindowException - When ever driver tries to switch to the window which is not available, it will throw 'NoSuchWindowException'. We need to check the window id that we pass or wait for some time until the new window appears. Check here for working with windows

NoSuchFrameException - When ever driver is Unable to navigate to frame with element, it throws 'NoSuchFrameException'. Check here for Working with Frames

TimeoutException - When the Element was not displayed in the specified time. When ever we work with waits, we may see these exceptions. Check the example below:

/Explicit wait for state dropdown field/ WebDriverWait wait = new WebDriverWait(driver, 10); wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("statedropdown")));

The above statement waits up to 10 seconds before throwing Exception (TimeoutException - Timed out after 10 seconds waiting for visibility of element) or if it finds the element, it will return in 0 - 10 seconds.

The exceptions that i have listed above are selenium webdriver exceptions. Here selenium webdriver class 'WebDriverException' extends 'RunTimeException' class. In turn there are sub classes like 'StaleElementReferenceException', 'TimeoutException', etc which extends 'WebDriverException'.

There is an other class which extends 'WebDriverException' is 'NotFoundException' and again this has sub classes 'NoSuchElementException', 'NoSuchFrameException' etc.

[10:36, 5/22/2017] +91 99628 41353: The below are the five keywords which plays the role in Exception handling : -

1. try

2. catch

3. finally

4. throw

5. throws

1. try:

try block contains the code that might raise an exception.

syntax:

try{ statements....//this code may raise an exception }

try block Example:

try{ int x=0; int y=10; System.out.println(y/x);// here we will get an exception as x is initialized as 0. so we should place these code in try block. }

2. catch:

catch block contains handling code if any exception occurs in try block. try must follows catch block. try after catch or finally is mandatory.

Syntax:

catch(Exception e){ statements....//contains handling code }

Example:

try{ int x=0; int y=10; System.out.println(y/x); } catch(Exception e){ System.out.println("Exception has been handled" + e);//once an exception raises instead of terminating the flow of program cursor jumps to particular catch block to handle this exception and prints a user friendly message and the rest of code executes normally } System.out.println("code after try catch block"); }

when an exception raises it creates the exception object e. This object contains name of the exception class, cause and location of exception. if we already know the exception class we can directly pass the particular exception class as parameter in catch block. if we dint know what kind of exception class we can pass the Exception class as parameter as it is the parent class for all exception classes.

Some important Note:

1. try always follow catch or finally block.

2. there may be nested try, multiple catch blocks for each try but there should be only one finally block.

3. In between try catch there should not be finally block.

4. try with finally is possible without catch.

5. there should not be any code in between try, catch or finally block

However if you don't need Javascript for your Selenium test you can try the following when you launch your driver :

FirefoxProfile p = new FirefoxProfile();

p.setPreference("javascript.enabled", false);

driver = new FirefoxDriver(p);