Waits

- When Selenium opens any browser, it opens with factory settings i.e. No add-ons, No History, No Auto Complete, No cookies. This is done to run the script faster.

SYNCHRONIZATION

- Process of matching Selenium speed with application is called as Synchronization. On real time applications when Selenium try to find the element it may throw 'NoSuchElementException' even though specified locator is correct. To handle this we can use 'Sleep' method of thread class.

Write a program without sleep method to demonstrate the issue without sleep.
Write a program with sleep method to demonstrate the issue with sleep.

Implicit wait

- If we use sleep() method we should specify it in all locations where application is slow. This will increase time taken to execute script, it consumes lot of space increases maintenance of the script and it always waits specified duration.
- **Ex:** if the duration is 20 sec, it will always waits for 20 sec even though element is displayed in the 5 sec. To overcome all these limitations, we should use the synchronization option given by Selenium called implicitlyWait as shown below.

- The duration specified in implicitlyWait statement is used only by findElement() and findElements() and not by any other methods. It takes two arguments first one is a duration and the 2nd argument is the TimeUnit such as
  - DAYS
  - HOURS
  - MINUTES
  - SECONDS
  - MILLISECONDS
  - MICROSECONDS
  - NANOSECONDS
- If we use implicitlyWait then if the element is not Located, the findElement() method will keep searching for the element after every 500 MILLISECONDS. This is duration is called as “Poling Period”. This is specified in a class called FluentWait. If the element is not located even after the duration then we get **“NoSuchElementException”**.

Write a Program with Explicit wait to demonstrate the use of explicit wait.

Explicit wait

- The explicit wait is used to tell the Web Driver to wait for certain conditions (Expected Conditions) or the maximum time exceeded before throwing an "ElementNotVisibleException" exception.
- The explicit wait is an intelligent kind of wait, but it can be applied only for specified elements. Explicit wait gives better options than that of an implicit wait as it will wait for dynamically loaded Ajax elements.
- Wherever we can’t use implicitlyWait (Other than findElement) we should use Explicit Wait. Since we specify the waiting condition explicitly. it is called as Explicit Wait.
- When the control comes to wait.until statement it will keep checking the condition after every 500 MilliSeconds. If the condition is satisfied it will go to next statement. If the condition is not satisfied even after the duration we get “TimeoutException”.

- Once we declare explicit wait we have to use "ExpectedConditions" or we can configure how frequently we want to check the condition using Fluent Wait. These days while implementing we are using Thread.Sleep() generally it is not recommended to use. “WebDriverWaits” itself is called Explicit Wait.

```
driver.get("https://online.actitime.com/na8/login.do");
driver.findElement(By.id("username")).sendKeys("kmrajay237");
driver.findElement(By.name("pwd")).sendKeys("zxcv1234");
driver.findElement(By.id("loginButton")).click();
WebDriverWait wait = new WebDriverWait(driver, 20);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("logoutLink")));
String title = driver.getTitle();
System.out.println(title);
```

- In the above example, wait for the amount of time defined in the "WebDriverWait" class or the "ExpectedConditions" to occur whichever occurs first.
- The above Java code states that we are waiting for an element for the time frame of 20 seconds as defined in the "WebDriverWait" class on the web page until the "ExpectedConditions" are met and the condition is "visibilityofElementLocated".
- **Expected conditions** : Because it is quite a common occurrence to have to synchronise the DOM and your instructions, most clients also come with a set of predefined expected conditions. As might be obvious by the name, they are conditions that are predefined for frequent wait operations. The conditions available in the different language bindings vary, but this is a non-exhaustive list of a few.

- alertIsPresent()
- elementToBeClickable(By locator)
- elementToBeClickable(WebElement element)
- elementToBeSelected(WebElement element)
- invisibilityOf(WebElement element)
- invisibilityOfAllElements(java.util.List<WebElement> elements)
- invisibilityOfAllElements(WebElement... elements)
- textToBePresentInElement(By locator, java.lang.String text)
- titleContains(java.lang.String title)
- titleIs(java.lang.String title)
- urlContains(java.lang.String fraction)
- visibilityOf(WebElement element)
- visibilityOfAllElements(WebElement... elements)
- visibilityOfAllElements(java.util.List<WebElement> elements)
- numberOfElementsToBeLessThan(By locator, java.lang.Integer number)
- numberOfWindowsToBe(int expectedNumberOfWindows)
- presenceOfAllElementsLocatedBy(By locator)

Fluent Wait

- The fluent wait is used to tell the web driver to wait for a condition, as well as the frequency with which we want to check the condition before throwing an "ElementNotVisibleException" exception.
- **Frequency:** Setting up a repeat cycle with the time frame to verify/check the condition at the regular interval of time.
- Let's consider a scenario where an element is loaded at different intervals of time. The element might load within 10 seconds, 20 seconds or even more then that if we declare an explicit wait of 20 seconds. It will wait till the specified time before throwing an exception. In such scenarios, the fluentwait is the ideal wait to use as this will try to find the element at different frequency until it finds it or the final timer runs out.

```
Wait wait = new FluentWait(WebDriver reference)
.withTimeout(timeout, SECONDS) ->Maximum Time
.pollingEvery(timeout, SECONDS) ->frequency
.ignoring(Exception.class);
```

```
Wait<WebDriver> wait=new FluentWait<WebDriver>(driver)
.withTimeout(30,TimeUnit.SECONDS)
.pollingEvery(5, TimeUnit.SECONDS)
.ignoring(NoSuchElementException.class);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("locator")));
```

- Frequency is set to 5 seconds and the maximum time is set to 30 seconds. Thus this means that it will check for the element on the web page at every 5 seconds for the maximum time of 30 seconds. If the element is located within this time frame it will perform the operations else it would throw “ElementNotVisibleException".

Difference Between Implicit, Explicit and Fluent Wait

- **Implicit Wait:** During Implicit wait if the Web Driver cannot find it immediately because of its availability, it will keep polling (around 500 milliseconds) the DOM to get the element. If the element is not available within the specified Time an NoSuchElementException will be raised. The default setting is zero. Once we set a time, the Web Driver waits for the period of the WebDriver object instance.
- **Explicit Wait:** There can be instance when a particular element takes more than a minute to load. In that case you definitely not like to set a huge time to Implicit wait, as if you do this your browser will going to wait for the same time for every element. To avoid that situation you can simply put a separate time on the required element only. By following this your browser implicit wait time would be short for every element and it would be large for specific element.

- **Fluent Wait:** Let's say you have an element which sometime appears in just 1 second and some time it takes minutes to appear. In that case it is better to use fluent wait, as this will try to find element again and again until it find it or until the final timer runs out.
- **Solutions:** We always get confused when it comes to using Wait commands, to better understand it we need to remember that there is a difference between several scenarios:
  - An element not being present at all in the DOM.
  - An element being present in the DOM but not visible.
  - An element being present in the DOM but not enabled. (I.e. clickable)
- There are pages which get displayed with the JavaScript, the elements are already present in the browser DOM, but are not visible. The implicit wait only waits for an element to appear in the DOM, so it returns immediately, but when you try to interact with the element you get a "NoSuchElementException". You could test this hypothesis by writing a helper method that explicit wait for an element to be visible or clickable.

Why Do We Need Waits In Selenium?

- Most of the web applications are developed using Ajax and Javascript. When a page is loaded by the browser the elements which we want to interact with may load at different time intervals. Not only it makes this difficult to identify the element but also if the element is not located it will throw an "ElementNotVisibleException" exception. Using Waits, we can resolve this problem.
- Let's consider a scenario where we have to use both implicit and explicit waits in our test. Assume that implicit wait time is set to 20 seconds and explicit wait time is set to 10 seconds.
- Suppose we are trying to find an element which has some "ExpectedConditions "(Explicit Wait), If the element is not located within the time frame defined by the Explicit wait(10 Seconds), It will use the time frame defined by implicit wait(20 seconds) before throwing an "ElementNotVisibleException".