Pre-Req:

Basic Manual Testing Concepts:

* Different types of manual Testing
* Regression Testing
* Test case
* Test Plan
* Bug life cycle
* Testing Methodologies
  + **Agile**
  + Verification and Validation

**Java**

* Basic components of Java
  + JDK
  + JRE
  + JVM
  + VERSIONS OF JAVA
* Basics of Java
  + OOPS Concept
  + class
  + interface
  + packages
  + constructors
* Variables
  + declare a variable
  + initialize a variable
  + Types of Variables
    - Local Variables
    - Global Variables
      * static variables
      * instance variables
      * constants
* Data types
  + primitive Data types
  + derived Data types
* Access Specifiers
  + private
  + default
  + protected
  + public
* Access Modifiers
  + static
  + abstract
  + final
  + synchronized
* Looping
  + for
  + while
  + do,while
  + foreach
* Conditional statements
  + if
  + if, else
  + if, elseif , else
  + switch
* String (java.lang)
  + charAt
  + split
  + indexOf
  + lastindexOf
  + toUpperCase
  + toLowerCase
  + equals
  + equalsIgnoreCase
* Exception Handling
  + try
  + catch
  + throw
  + throws
  + finally
  + User Defined Exceptions
* File Handling
  + Read file (properties file, text file, excel file)
  + Write into a file (properties file, text file, excel file)
* Arrays, Enumerator
* Collections
  + List
    - ArrayList
    - LinkedList
    - Vector
  + set
    - HashSet
    - TreeSet
    - LinkedHashSet
  + Map
    - HashMap
    - TreeMap
    - LinkedHashMap
* Generics

**Selenium:**

* What is Regression Testing
* What are the drawbacks of Manual Testing
* Why Automation testing is required



* Why Selenium????
* 
* History of Selenium
* Selenium IDE
  + Beginners
  + Addon FF and Chrome
* ~~Selenium RC~~
* Selenium WebDriver / Selenium2.0 / Selenium3.0 / Alpha Selenium4.0
  + type
  + click
  + mouse movement
  + double click
  + right click
  + browser back / forward / refresh / maximize
  + read values from application
  + read Webtable
  + Auto suggestions
  + Automate tooltip info
  + Handle different / multi browser /tabbed browser
  + Calendar
  + Images
  + Popup
    - alerts
    - confirmation popup
    - hidden division popup
    - authentication popup
    - file download popup
    - file upload popup
  + Load the browser with customized settings
  + Execute Test on Different Browsers – IE, Chrome, FF, Opera, Safari
  + Data driven testing
    - excel
    - properties
  + End to end tests
* Selenium Grid
* Cucumber – BDD Approach
* Frame Work
  + Page Object Model (POM)
  + TestNG
    - annotations
    - validating HTML Reports
    - grouping the tests
    - Data driven testing
* Maven – Build Automation Tool
  + Maven Repository
    - Local Repository
    - Global Repository
  + POM.xml
  + setup a maven project
  + pre settings to be updated
  + Maven Build Life cycle phases
  + Execute the test without opening eclipse – Using Batch file
* GIT – Version Control Tool
* Jenkins – CI/CD Tool
  + how to configure Jenkins
  + how to create builds
  + how to execute tests in Jenkins
  + how to execute tests daily

====================================================================================================JAVA==========================================================================================================

1. Installation
2. Set the environment Variables
   1. Set JAVA\_HOME
   2. Update PATH Environment Variable



c. Open Command prompt and verify java



1. Components of Java



* JDK 🡪 Java development KIT
  + Compile the Java Program
* JRE 🡪 Java Runtime Environment
  + Run the Java Program
* JVM 🡪 Java Virtual Machine
  + Loads code
  + Verifies the code
  + Executes the code
  + Provides runtime environment
* History:

1. Editor – Eclipse, netbeans, intelliJ
2. HelloWorld JAVA Program
3. Understand the Java Program in detail.
   1. 

b. 

1. Data types
   1. data types

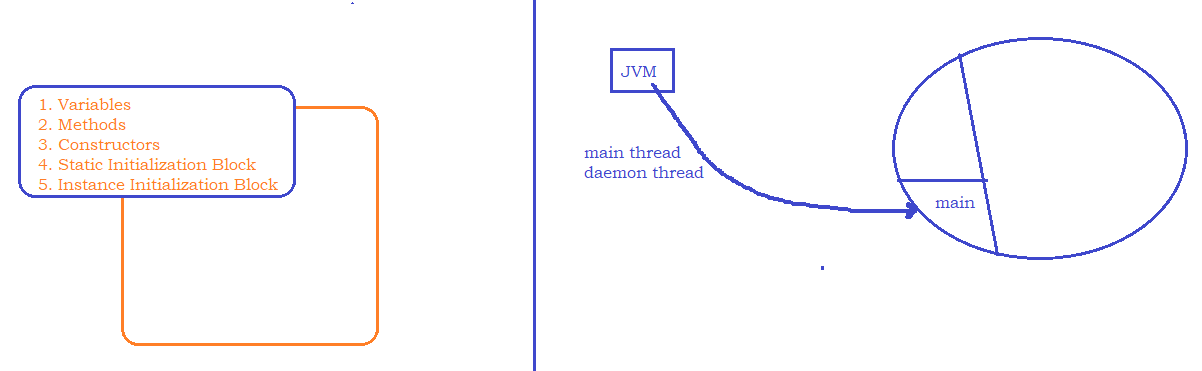


b. 

1. Packages
   1. always package should start com / org
   2. followed by company name / client name
   3. followed by project name
   4. followed by module / sub-module name
   5. ****
2. Variables
   1. declare 🡪 data\_type variable\_name
   2. initialize 🡪 variable\_name = Value;
   3. Types

****

**Members of Class : Execution by main thread**

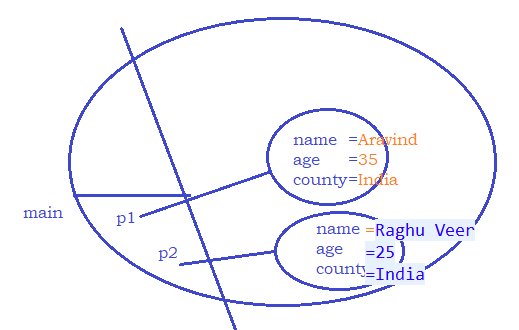
****

**Access members of one class inside another –**

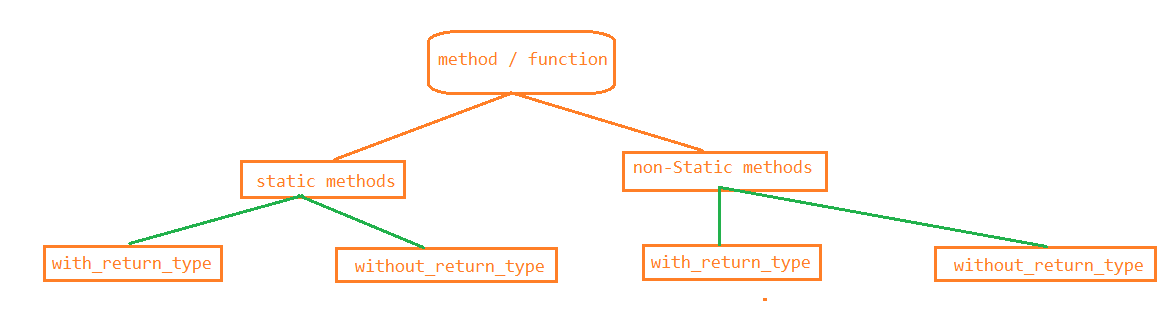
1. Create Object to the Class

2. Use dot operator to access the member of a class

Instance Variables



1. **Methods – Describes the behaviour of a class**

****

**[Access\_specifier] [Access\_modifier] return\_type name\_of\_method(ArgumentList )**

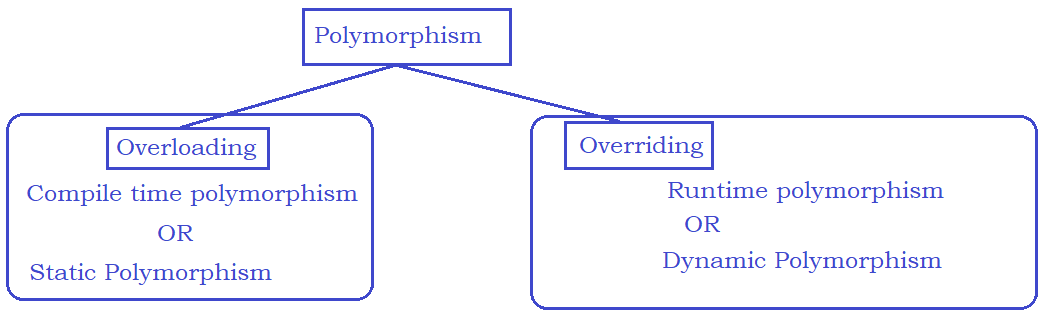
**{**

**--------------------**

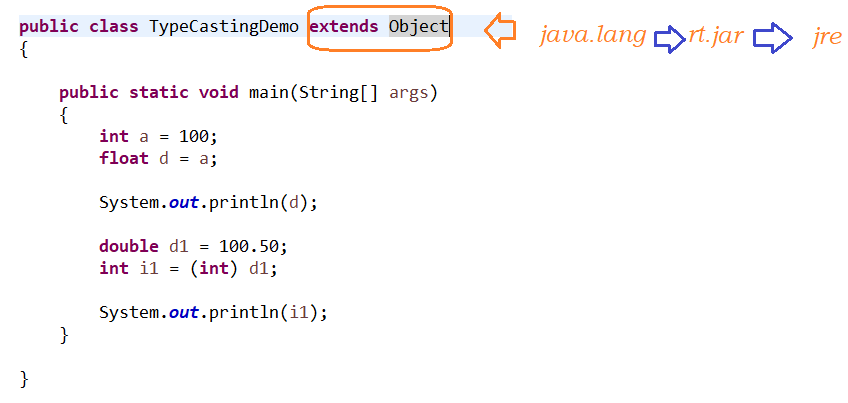
**--------------------**

**}**

* 1. **Variable number of Arguments to a function**
  2. **Polymorphism**

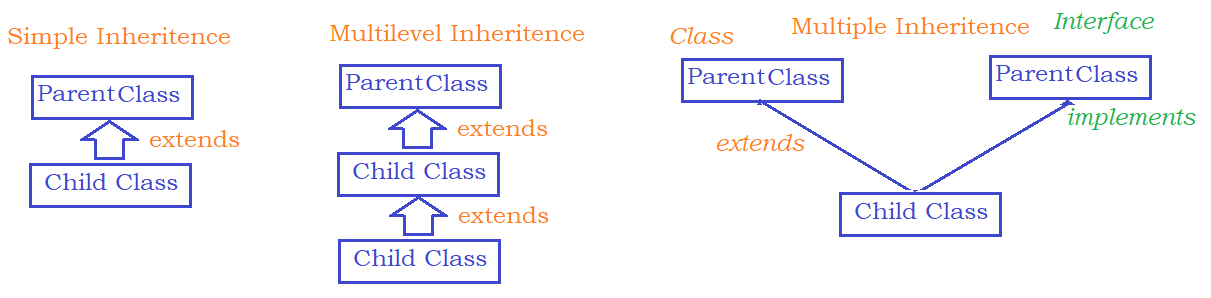
****

1. **Type Casting**

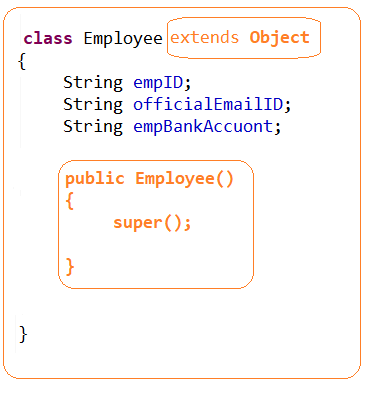
****

* 1. Auto type casting -> int to double, byte to int
  2. Explicit type casting -> double to int, int to byte

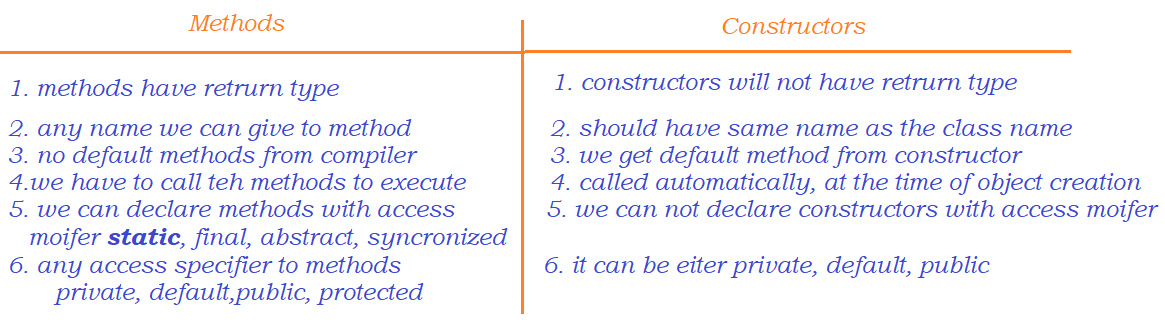
1. **Inheritance**

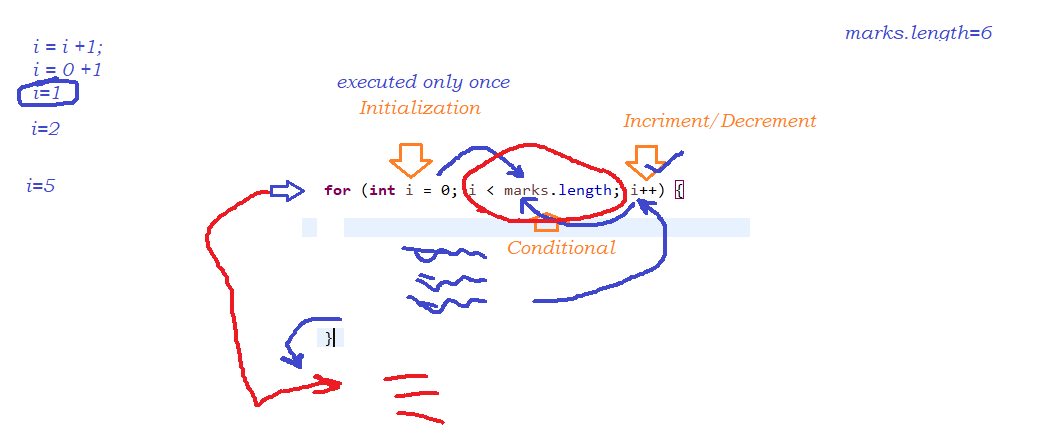
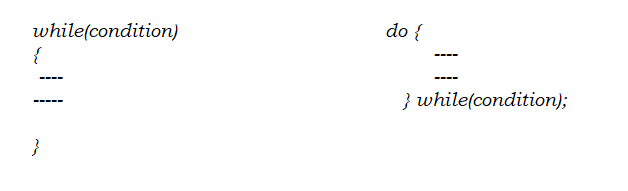
****

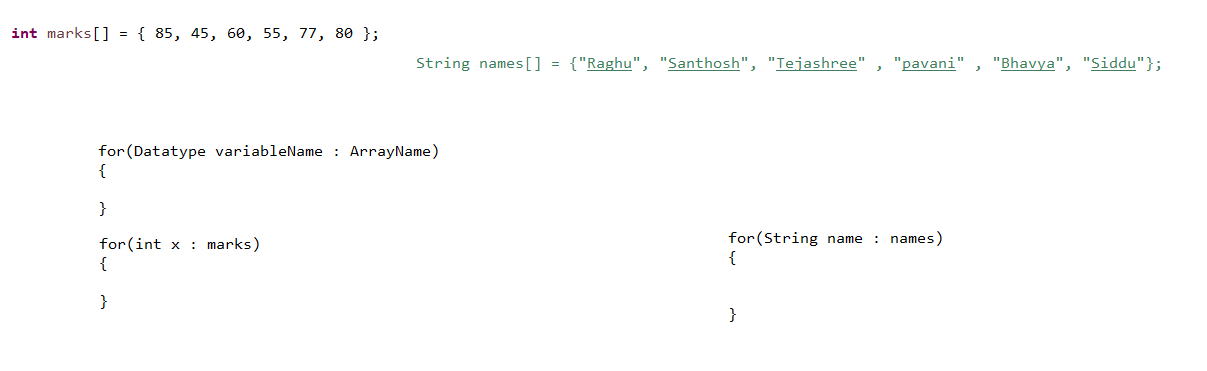
1. **Constructor**
   1. **Default Constructor**

****

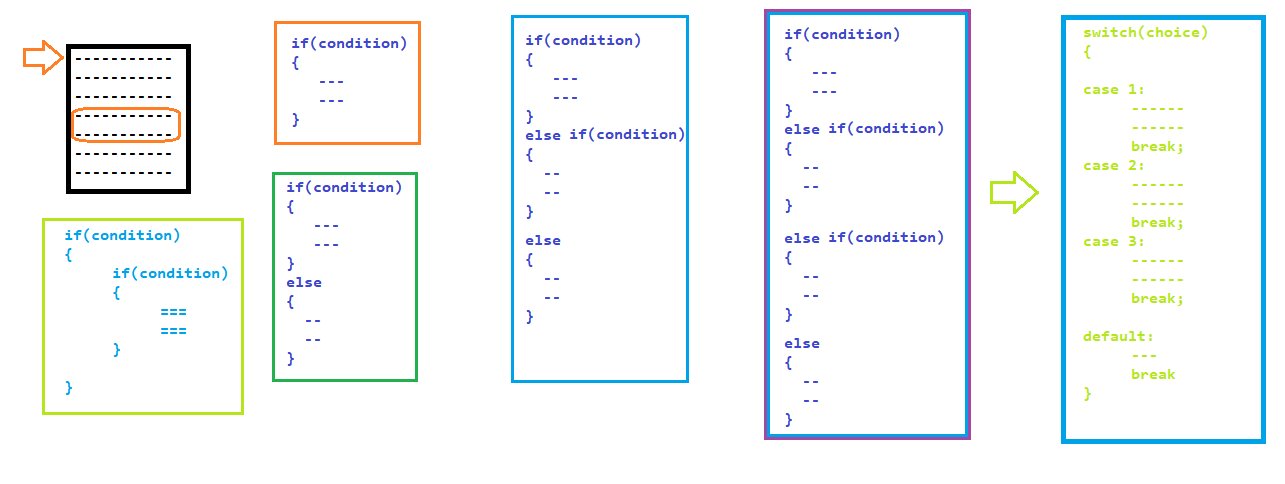
* 1. **Parameterized Constructor**
  2. **Difference between Methods and constructors**



1. **Arrays**
   1. single dimensional
   2. Multidimensional array
2. **Looping Statements**
   1. for
      1. ****
      2. ****
   2. while
      1. ****
   3. do,while
   4. foreach

****

1. Conditional Statements



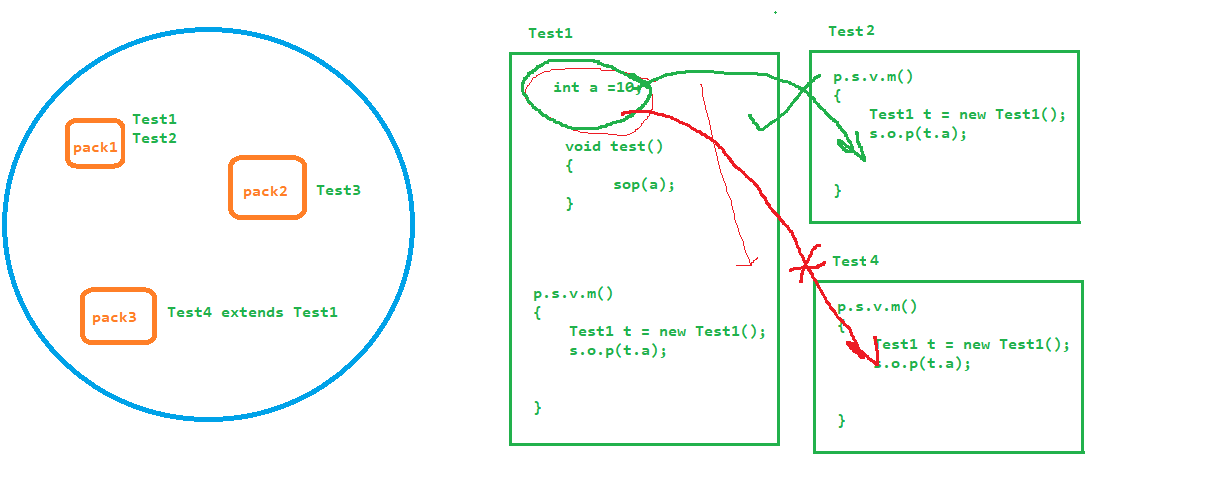
1. SIB and IIB
   1. Static initialization Block – will be called before executing main method
   2. Instance initialization Block – will be called before calling a constructor
2. **Abstract Class**

Class can be declared as abstract in 2 scenarios

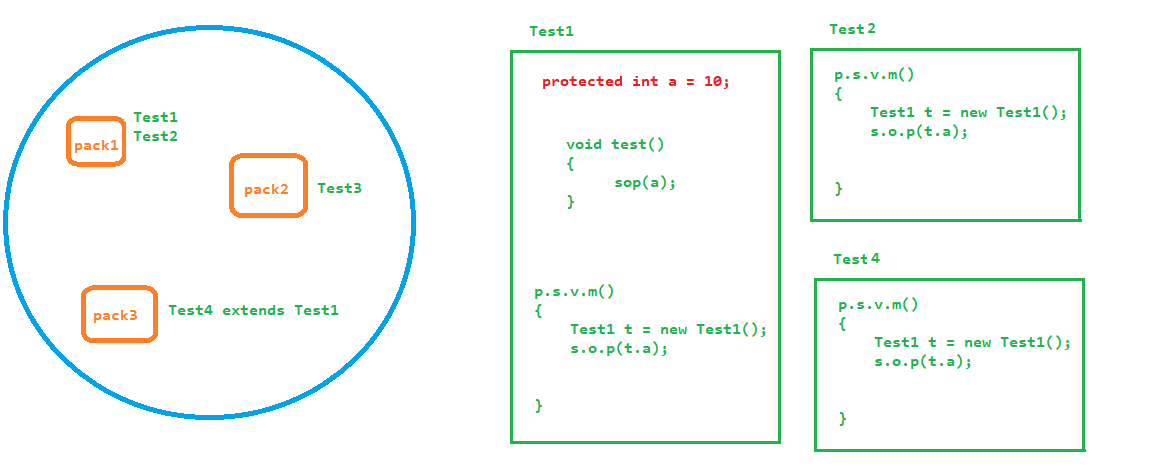
1. if it has any unimplemented methods / abstract methods

2. if you want to avoid creating an object to your class.

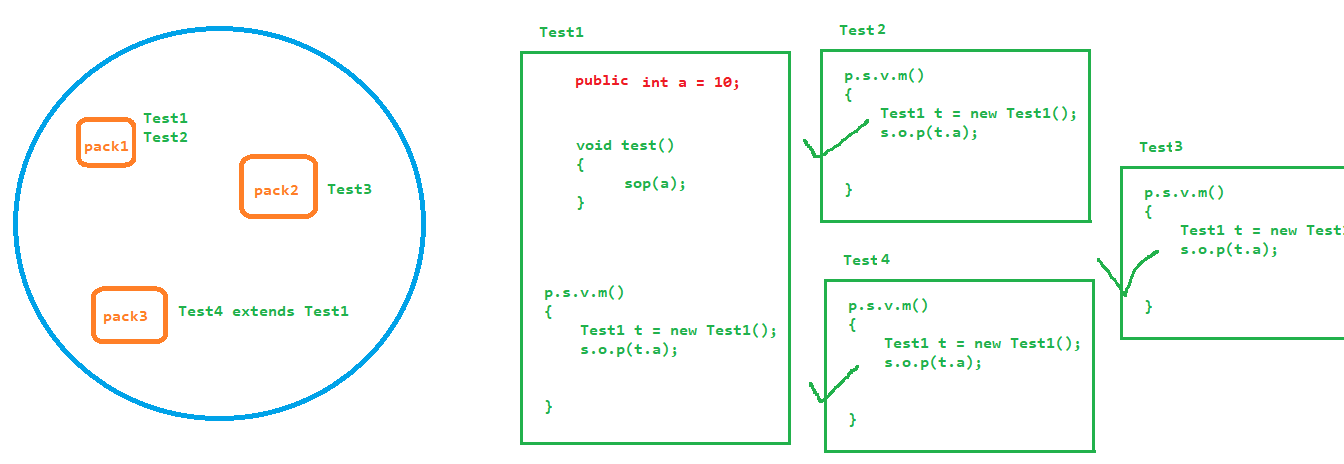
1. **Access Specifiers – *Specifies the accessibility of a member (variables, methods, constructors )***
   1. private
   2. package (default )

****

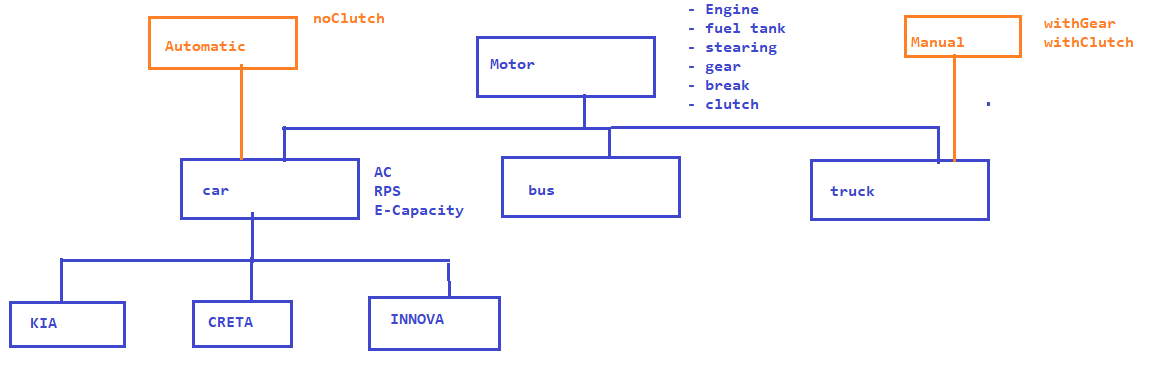
* 1. protected

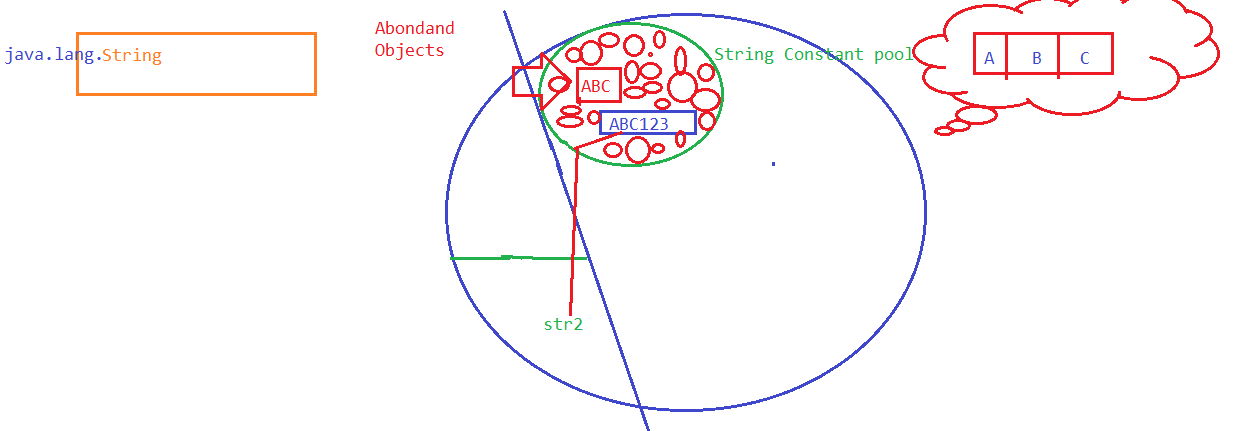
****

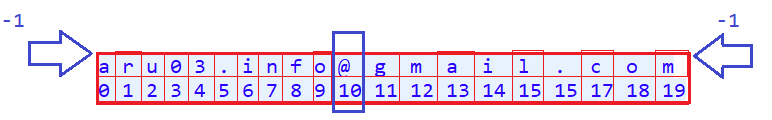
* 1. public

****

1. **Access Modifiers – *Change the behaviour of a member (variables, methods, constructors )***
   1. static
   2. final
   3. abstract
   4. synchronized
2. **Interface**

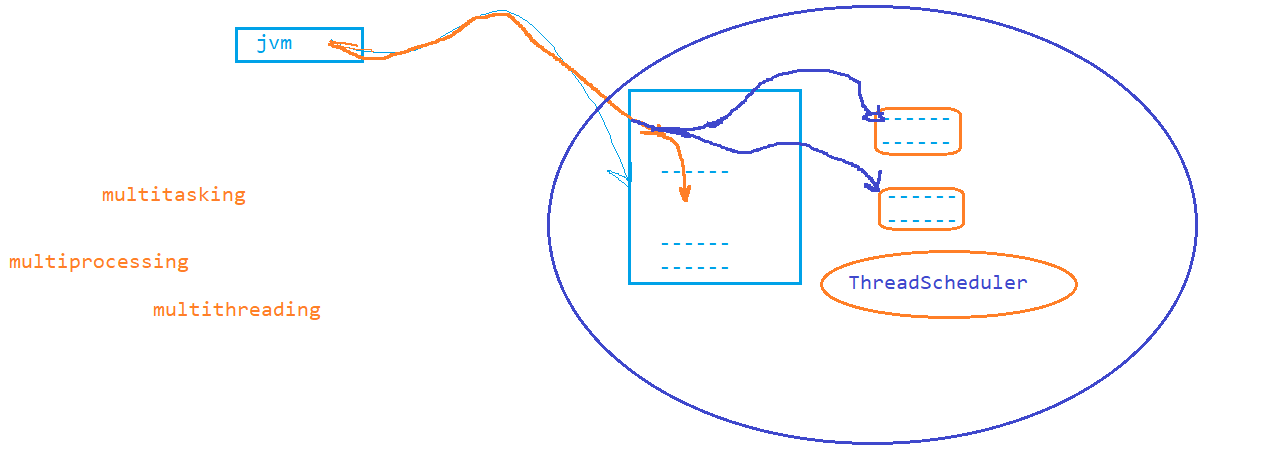
****

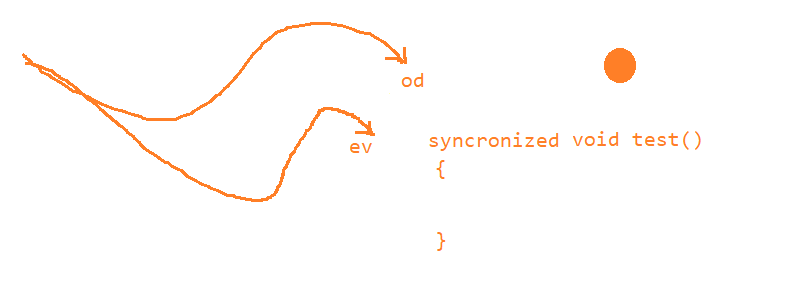
1. **Enumerator**
2. **Strings**
   1. ****



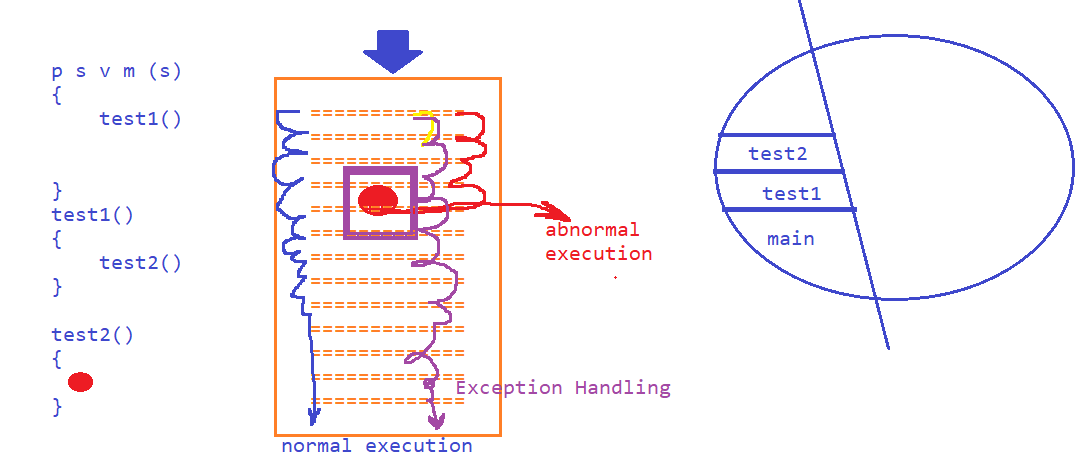
* 1. **StringBuffer**
  2. **StringBuilder**

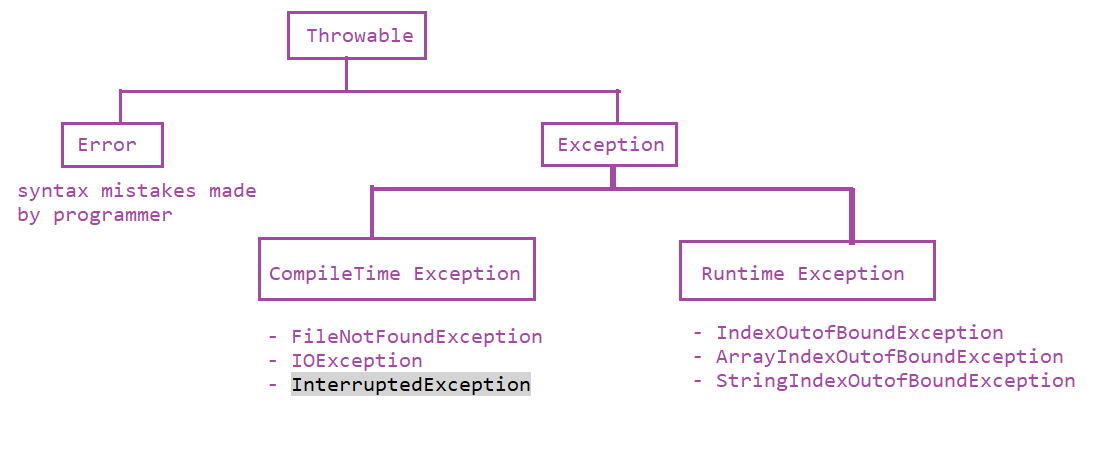
1. **Threads**



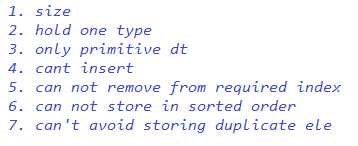
****

1. **Exceptions**

****

****

* 1. **try**
  2. **catch**
  3. **throw**
  4. **throws**
  5. **finally**

1. **FileHandling**
   1. **InputStream**
   2. **OutputStream**
   3. **File**
2. **Collections**
   1. **Drawbacks of Array**
      1. ****
   2. **List**
      1. **ArrayList**
      2. **LinkedList**
      3. **Vector**
   3. **Set**
      1. **HashSet**
      2. **TreeSet**
      3. **LinkedHashSet**
   4. **Map**
      1. **HashMap**
      2. **LinkedHashMap**
      3. **TreeMap**
3. **Generics**
4. **Wrapper Clases**
   1. **int -> Integer**
   2. **float -> Float**
   3. **double ->Double**
   4. **char -> Character**
   5. **Boolean ->Boolean**
   6. **long->Long**
   7. **---**

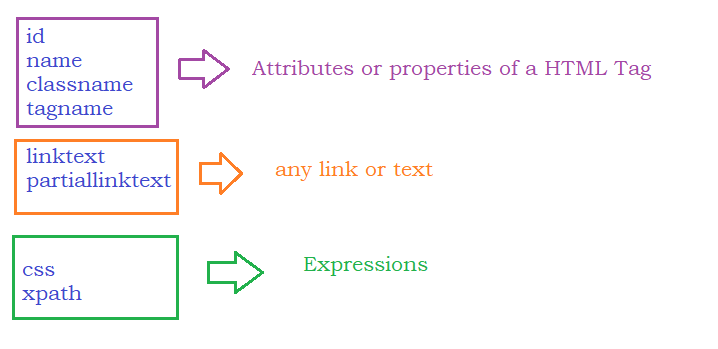
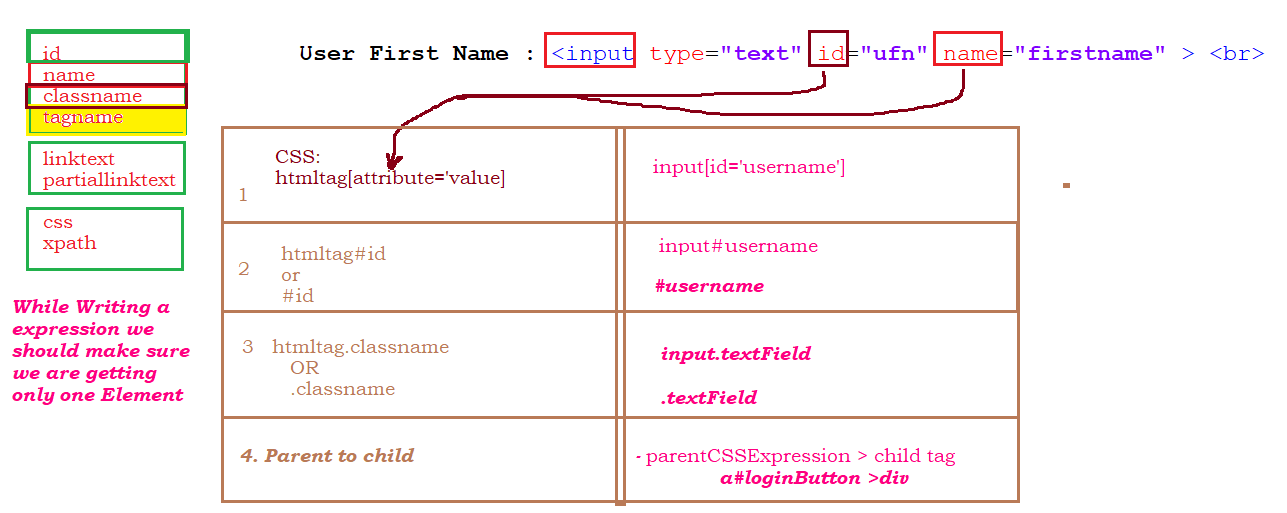
**Selenium:**

* What is Regression Testing
* What are the drawbacks of Manual Testing
* Why Automation testing is required



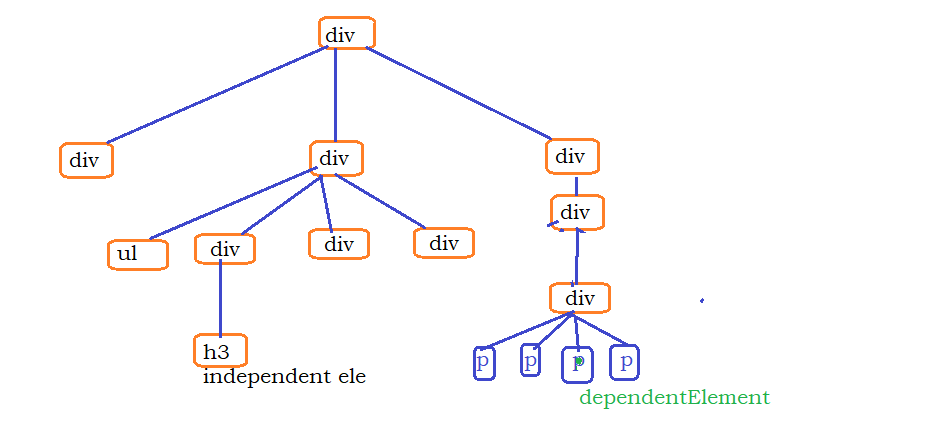
* Why Selenium????



* History of Selenium – Jason H
* Selenium IDE
  + Beginners
  + Addon FF and Chrome
* ~~Selenium RC~~
* Selenium WebDriver / Selenium2.0 / Selenium3.0 / Alpha Selenium4.0
* 
* 

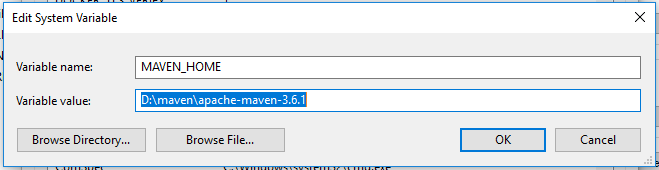
**Xpath: - XML Path –gives the address of the element in the webpage**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **INDEX** | **Xpath** | **Syntax** | **Example** | |
| **1** | **Basic Xpath** | //htmltag[@attribute=’value’] | //input[@name='username']  //input[@name='remember'] | |
| **2** | **Using**  **Expressions** | //htmltag[@attribute1=’value1’  and @ attribute2 = ‘value2]  //htmltag[@attribute1=”value1”  or @ attribute2 = ‘value2’]  //htmltag[@attribute1=’value1’  not @ attribute2 = ‘value2’] | //input[@type='text' and @name='username']  //input[@type='text' and @type='password']  //input[(@type='text' or @type='password') and not(@type='checkbox')] | |
|  |  | Calendar from redbus.in | //td[(@class='current day' or @class='we day' or @class='wd day') and text()='4'] | //td[text()='5' and not(@class='past day')] |
| **3** | **Using**  **Functions**  **1. text()** | //htmlTag[functionname=’value’] | //div[text()='Login '] | |
|  | **2. contains**  **(arg1,arg2)** | //htmlTag[contains(arg1, arg2)]  arg1 - > attribute or Function  arg2 -> partial value | //td[contains(text(),'yourself')]  //img[contains(@src,'timer')] | |
|  | **3. starts-with**  **(arg1,arg2)** | //htmlTag[starts-with(arg1, arg2)]  arg1 - > attribute or Function  arg2 -> partial value | //div[starts-with(text(),'syn-BDD')]  //img[starts-with(@id,'timeTrack')] | |
| **4** | **Traversing**  **From parent**  **To Child** | //parent\_expresstion/child\_expression | //a[@id='loginButton']**/**div  //td[@id='loginButtonContainer']**//**div | |
| //div[@class='info\_hero']/div[contains(text(),  'Safety')] | |
| **5** | **Traversing**  **From Child**  **To Parent** | *Whenever we have dependent and*  *Independent element*  *//xpath for child item*  //parent\_html\_tag[*xpath for*  *child item*] | //tr[th[text()='Directed by']]//a  //div[div[div[h3[contains(text(),'Romantic')]]]]  //p[contains(@class,'Bold')] | |
| **6** |  | **Axes**  **Functions** |  | |
|  |  | 1. following-sibling | //th[text()='Directed by']/following-sibling::td/a | |
|  |  | 2. preceding-sibling | //li[a[span[text()='Production']]]/preceding-sibling::li | |
|  |  | 3. following | //h2[span[text()='Production']]/following::a | |
|  |  | 4. preceding | //h2[span[text()='Production']]/preceding::a | |
|  |  | 5. parent | //span[text()='Production']/parent::h2 | |
|  |  | 6. child | //th[text()='Directed by']/following-sibling::td/child::a | |
|  |  | 7. ancestor | //h3[contains(text(),'Essential')]/ancestor::div[contains(@class,'boxShadow')]//p[contains(@class,'black')] | |

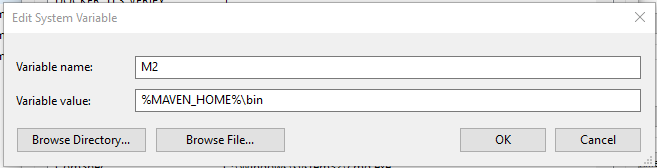
****

**Selenium with Maven**

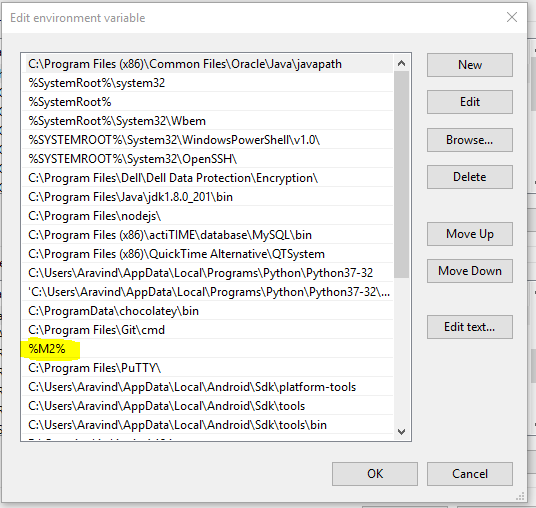
* Maven – Build Automation Tool
* Maven Plugin with eclipse
* Installation on Windows
  + Download the software(zip file) from official website
  + Keep it on any location and extract it
  + Set Environment Variables
  + MAVEN\_HOME



* + M2



* + PATH



* Drawback without Build Automation tool
  + Building Project Steps:

-----------------------

1. Download the libs required (replace if any latest version is present )

2. Writing code

3. Delete the previous Class files

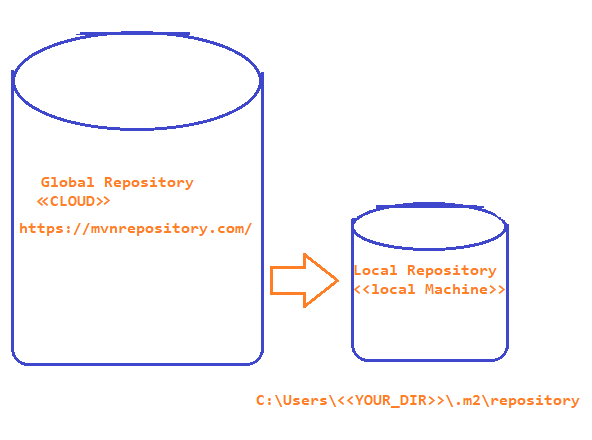
4. Compile the code

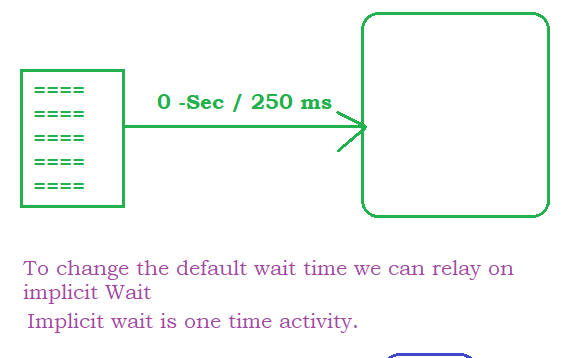
5. Run the unit test

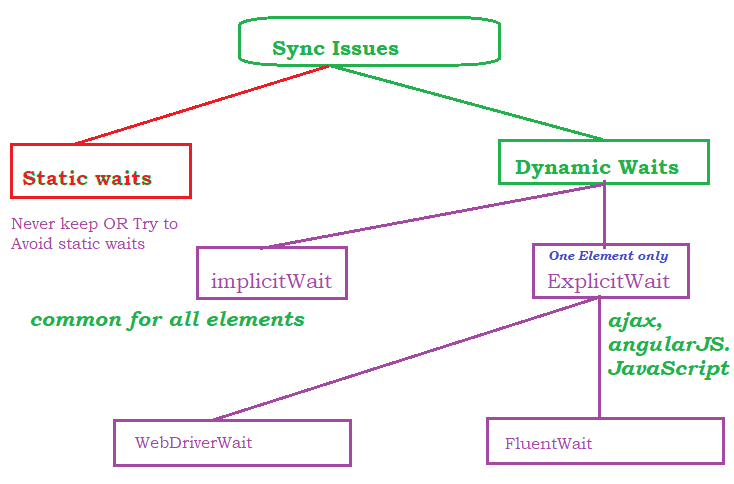
6. Create a jar file or war file

7. Deploy in a QA Server / Staging Server / Production Server

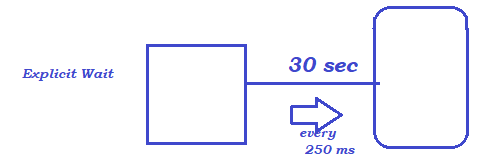
* Build Automation tools:
  + Ant
  + Maven
  + Gradle
  + 
* Creating Eclipse Project
  + Create project Quick start plugin
  + Change the complier to latest version
  + Update the jre to be used from latest jdk version
  + POM.xml – in detail
  + Updating dependencies in POM.xml
  + Maven Repository
    - Local Repository
    - Global Repository



* + POM.xml
  + setup a maven project
  + pre settings to be updated
  + Maven Build Life cycle phases
  + Execute the test without opening eclipse – Using Batch file
* WebDriver
  + get(“String”)
  + findElement(By)
* WebElement
  + sendKeys(“String”)
  + click()
  + getText()
  + getAttribute(String)
* Sync Issues (20% Challege)
  + 
  + Types



Explicit Wait



* **WebDriverWait**

**--------------------------------------------**

1.Create a object to WebDriverWait Class by passing max Wait time

2. from wait object call until method

3. inside until method pass ExpectedConditions class and call the required method

* **Fluent Wait :**

**----------------------**

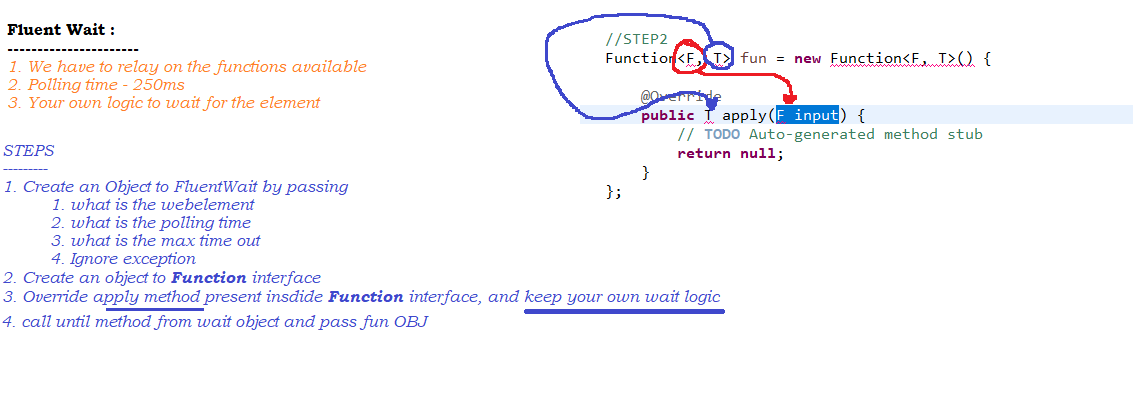
*1. We have to relay on the functions available*

*2. Polling time - 250ms*

*3. Your own logic to wait for the element*

* + **STEPS**

**---------**

****

* + 