Core Java

* Data types
  + primitive DT
  + derived DT
* Variables (data)
* methods (behaviour)
* constructor
* static Blocks
* Instance Blocks
* Classes
* Interfaces in Java
* Access specifiers
  + private
  + default
  + protected
  + public
* Access modifiers
  + static
  + abstract
  + synchronized
  + final
* Strings in Java
  + functions
  + StringBuffer
  + StringBuilder
* Threads
* Exception Handling
  + try
  + catch
  + throw
  + throws
  + finally
  + User Defined Exception
* Files
  + properties , text file, excel file
  + Read a file
  + Write to a file
* Loops
  + for
  + foreach
  + while
  + do,while
* conditional Statements
  + if
  + if else
  + if elseif else
  + switch
* Abstract Classes
* Wrapper Classes
* Arrays
* Collections
  + List
  + Set
  + Map
* Generics

Automation Testing :

* What is Software Testing
* What are the drawbacks of Manual testing
* What is regression Testing
* What are the different tools available for automation testing
* Why Selenium is popular ???
* **Selenium 4.0**
* Selenium IDE
  + Record and Playback
* ~~Selenium RC~~
* Selenium WebDriver
  + Object identification
    - id
    - name
    - classname
    - tagname
    - linktext
    - partiallinktext
    - css
    - xpath
      * Basic xpath
      * xpath using Logical Operator
      * xpath using function
      * traversing from parent to child
      * traversing from child to parent
      * xpath using axes functions
        + Traversing to previous siblings
        + Traversing to next siblings
        + Traversing to parent
        + Traversing to child
        + Traversing to ancestor
  + How to play with textbox, radio, checkbox, hyperlinks, images
  + How to play with web table, dynamic elements
  + How to perform validation
  + Select Class
  + Actions Class – KB and Mouse operation
  + End to End Automation
  + Data driven Testing
  + Popups
    - alert
    - confirmation
    - hidden division
    - file download
    - file upload
* AutoIT
* Selenium Grid
* Frameworks
  + ~~Function Driven Automation framework~~
  + ~~Keyword Driven Automation framework~~
  + ~~Hybrid Driven Automation framework~~
  + Page Object Model
  + TestNG
    - annotations

Build Automation Tools :

* Maven
* gradle

Jenkins :

GIT :

Interview Questions : - On the topics covered

2 Sample Applications :

Installations :

1. Java - JDK 1.8
   1. <https://www.oracle.com/in/java/technologies/javase/javase8-archive-downloads.html>
      1. JDK – Java development kit
      2. JRE- Java Runtiime Environment
   2. Open command prompt
   3. java –version ( ON any error set the Environment variables )
2. Editor : Visual Studio Code, **Eclipse**, IntelliJ, ...
   1. https://www.eclipse.org/downloads/packages/release/2021-09/r/eclipse-ide-java-developers
3. GIT
   1. <https://git-scm.com/downloads>
   2. Configuration
      1. signup to gitbub and create your own account
      2. update user name and email in the git commandline tool
         1. git config --global user.name "<<USERNAME>>"
         2. git config --global user.email <<EMAIL>>
   3. Create a Repository in your machine
      1. git init
      2. create some files
      3. add the file to version control -> git add file1, file2......filen OR git add .
      4. git commit –m “Message ” 🡪 this will generate commit id
      5. git push origin master OR
      6. git push --set-upstream origin master

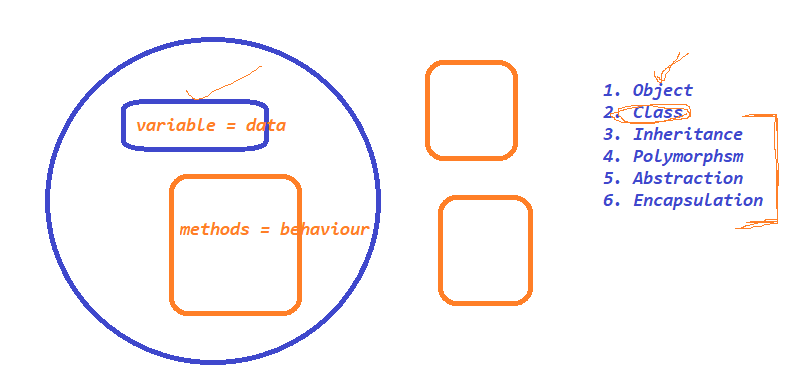
NOTE : link your local repository with the global repository by executing

git remote add origin <<Remote Repository URL>>

JAVA :

­

OOPS Concept :



­­­

Features of JAVA :

1. Simple

2. Object oriented Programming language

3. Platform Independent

4. Secure

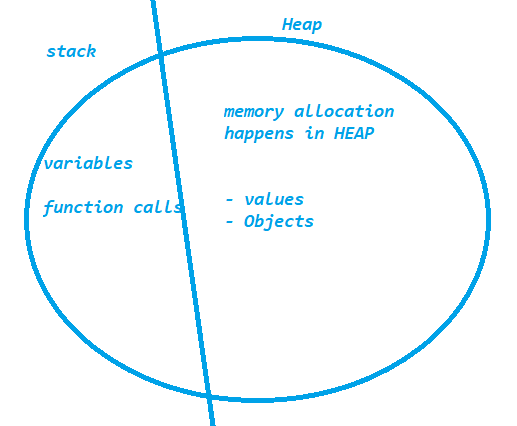
5. Robust

6.Architecture-neutral

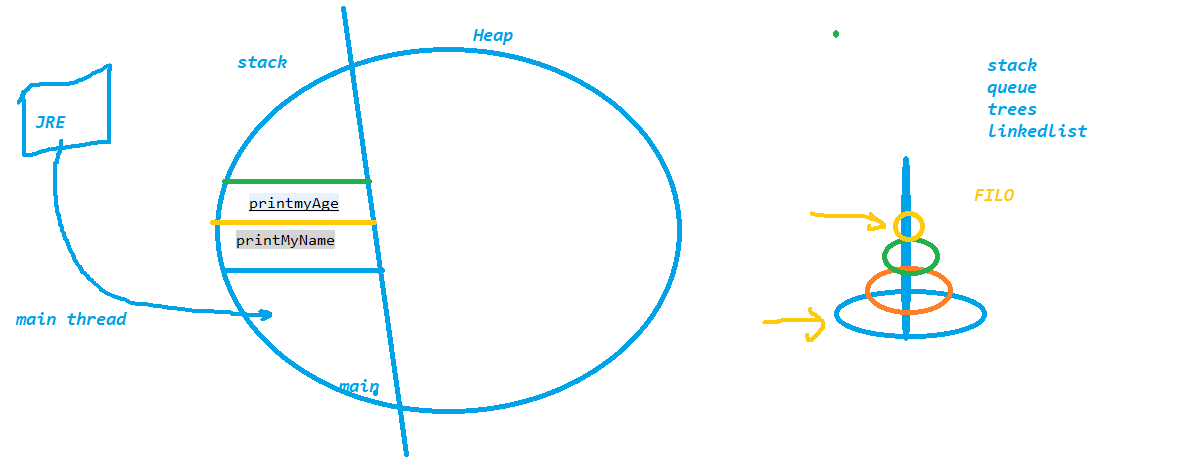
7. Portable

8. Multi-Threading

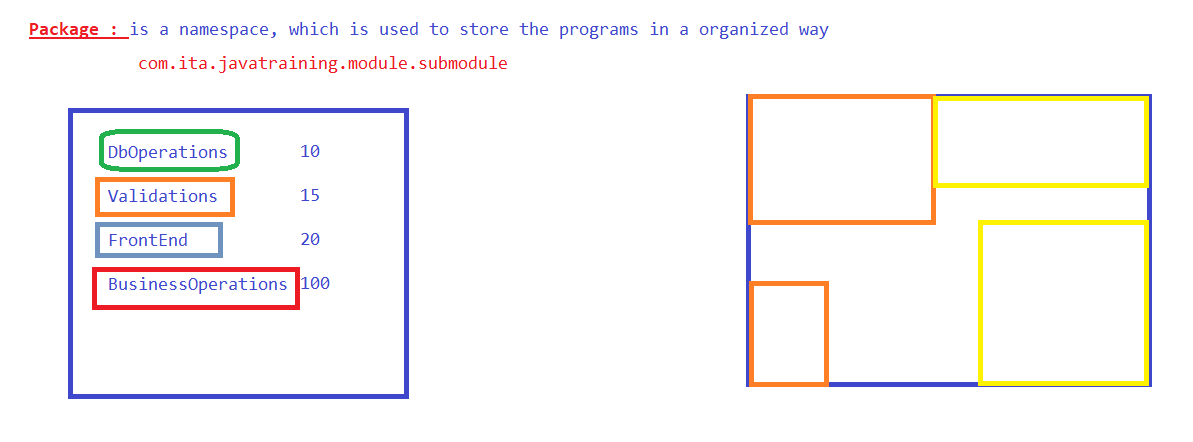
HelloWorld JAVA Program



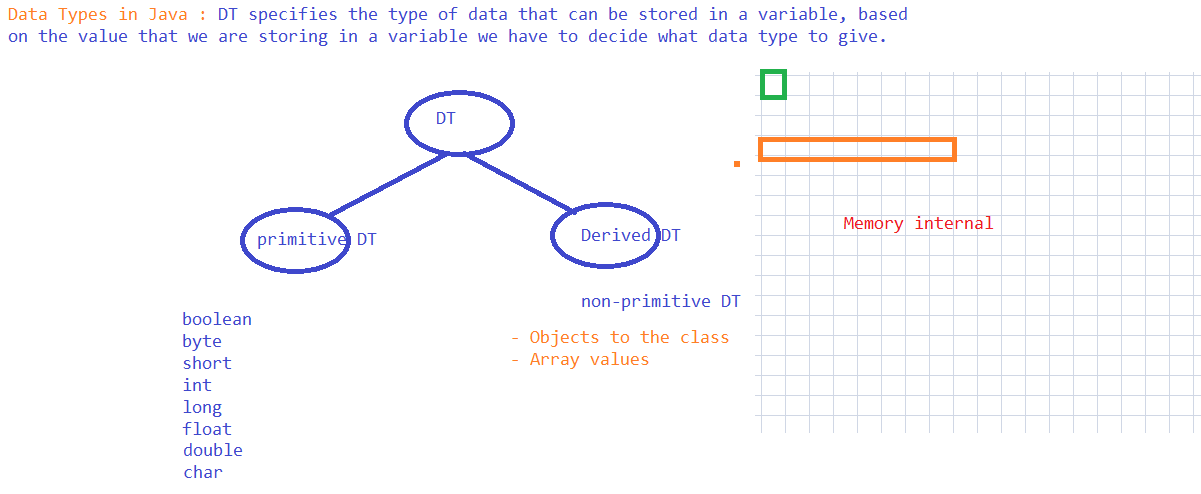
Stack Trace :

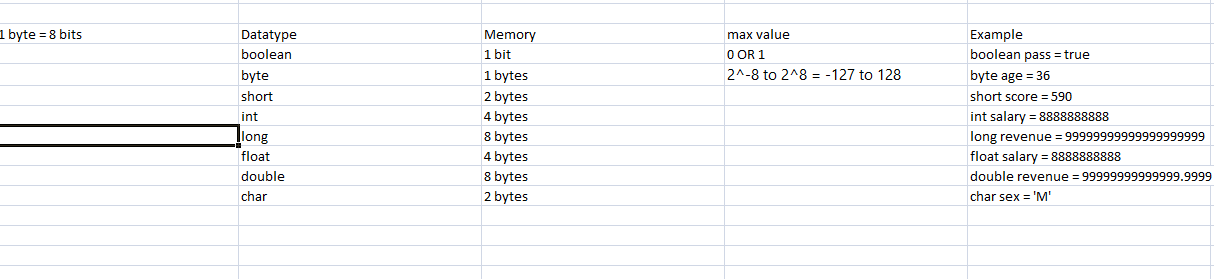


Packages in java :

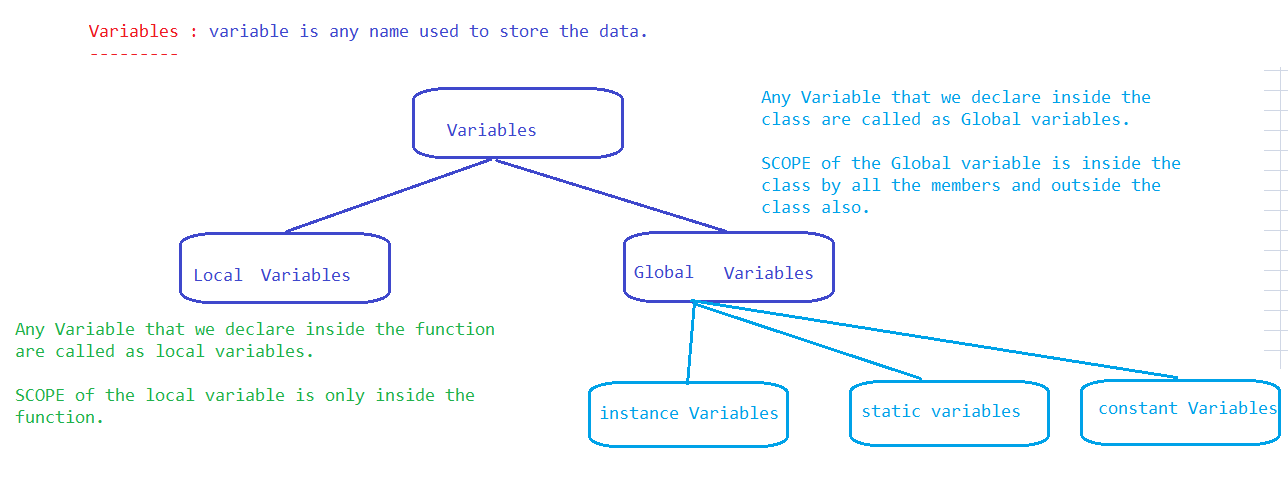


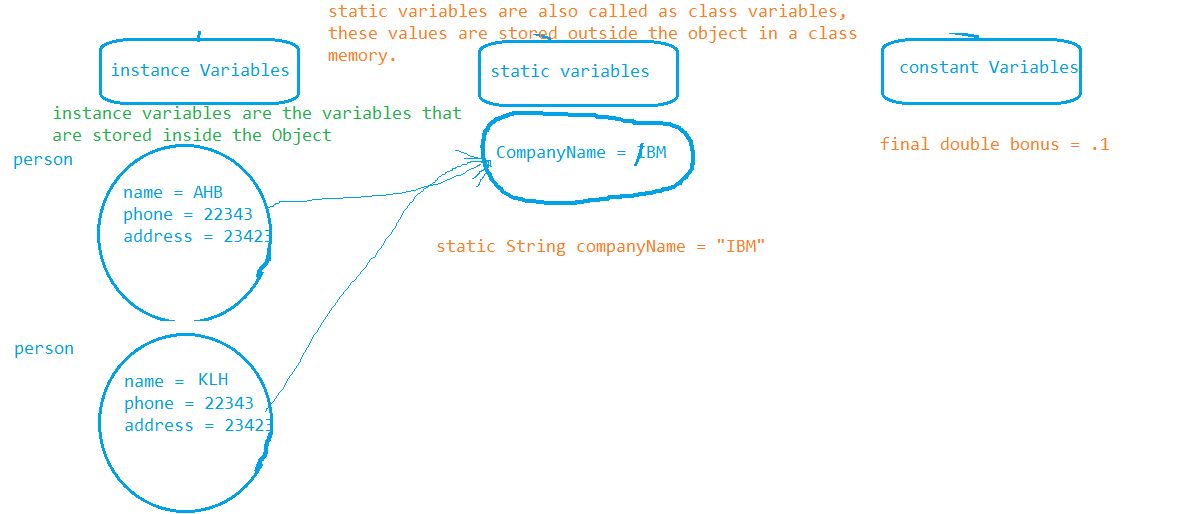
Data Types in JAVA :



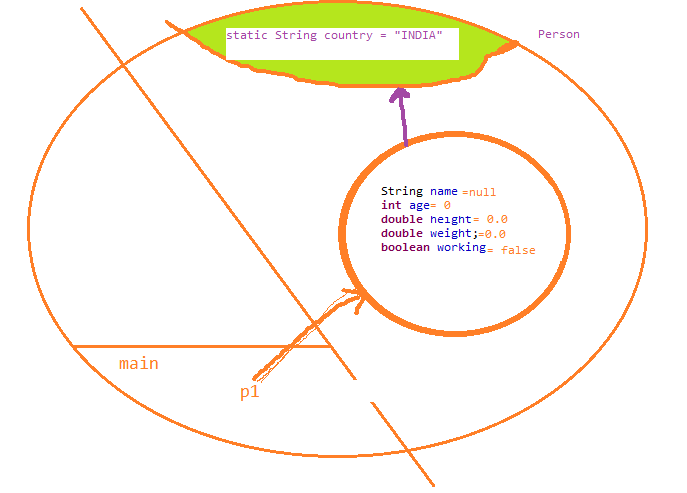


Variables :

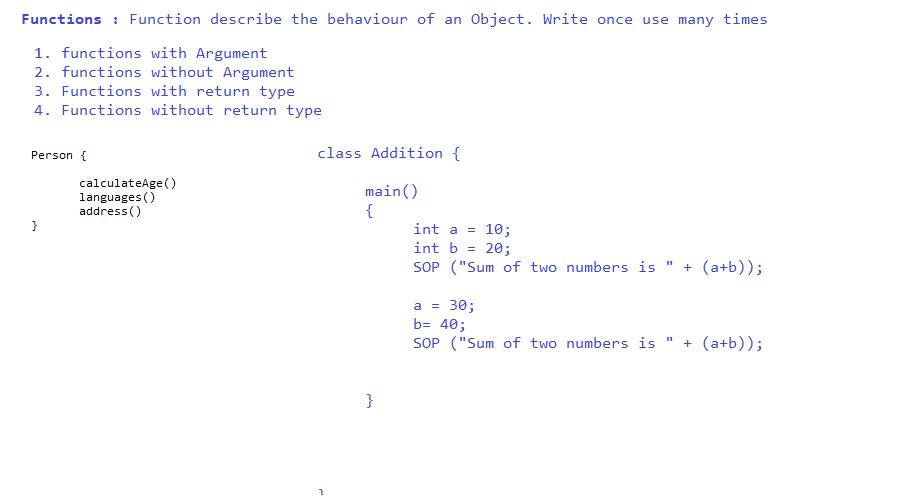


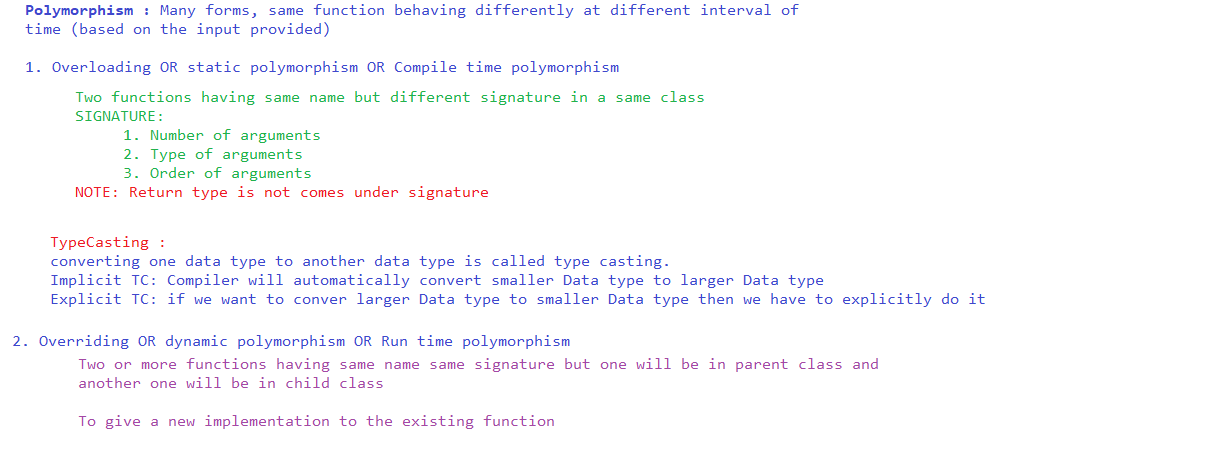


static variables demo :

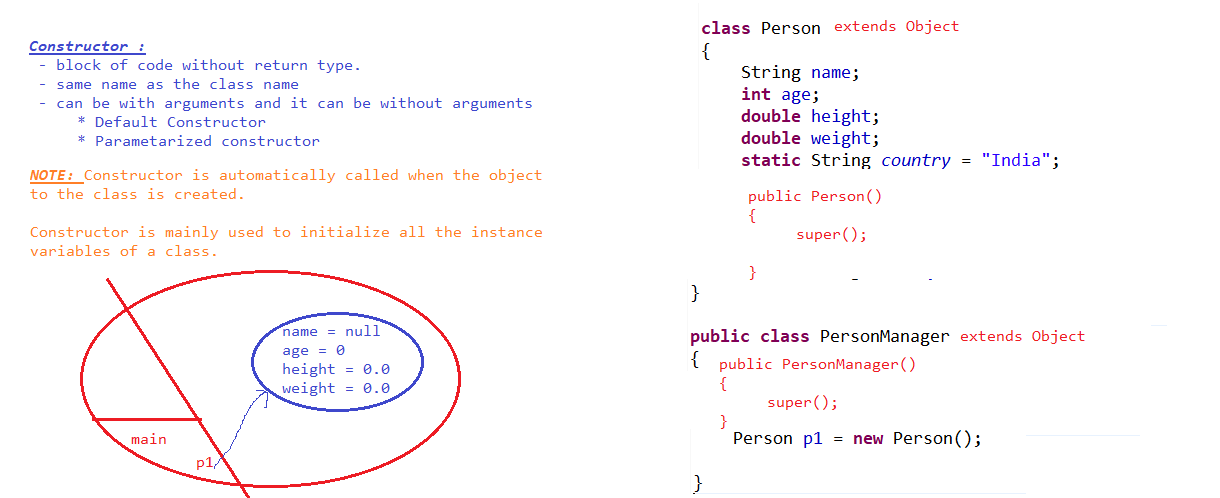


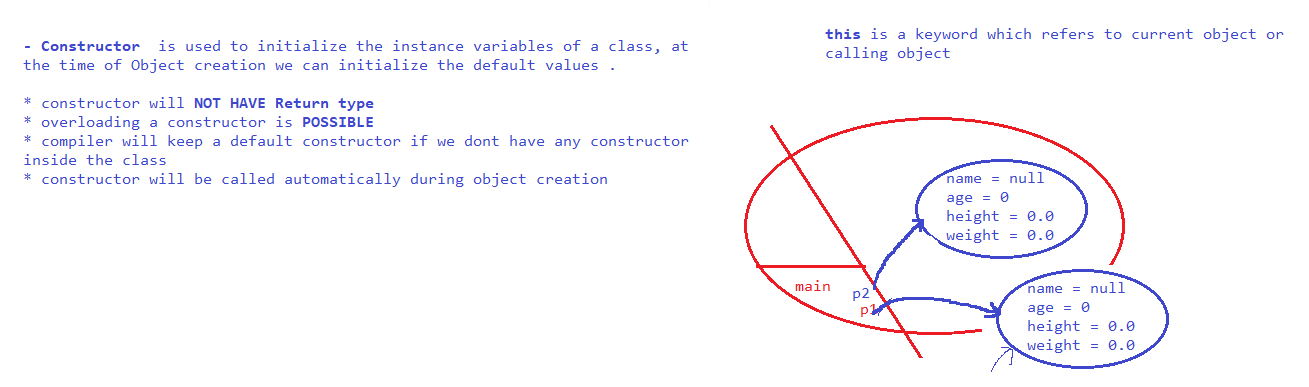
**Functions OR methods**

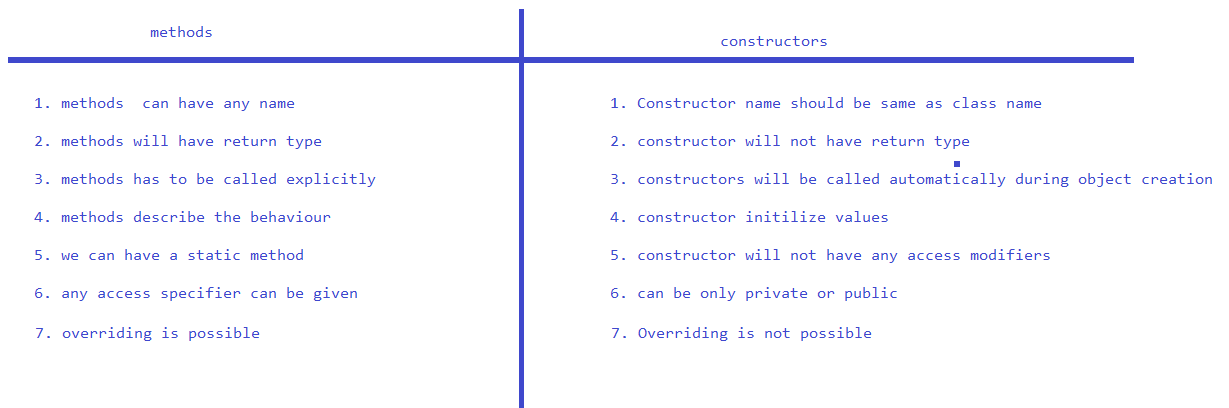
****



Constructors :







Overriding :

with toString method.

Static initialization Block

static

{

}

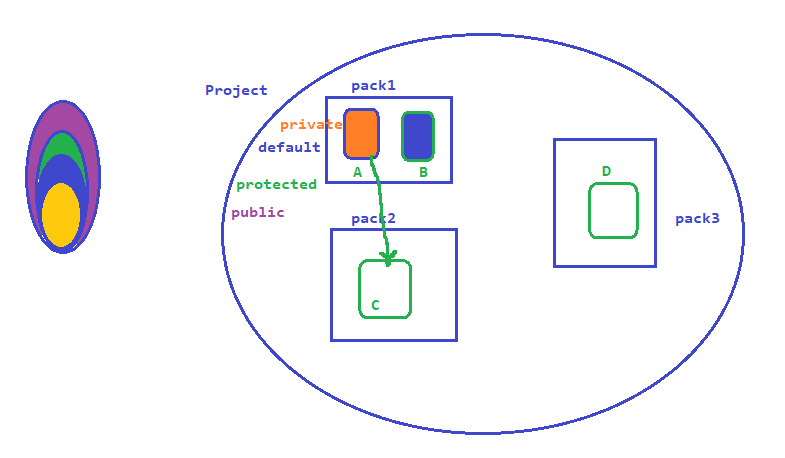
Instance initialization Block

{

}

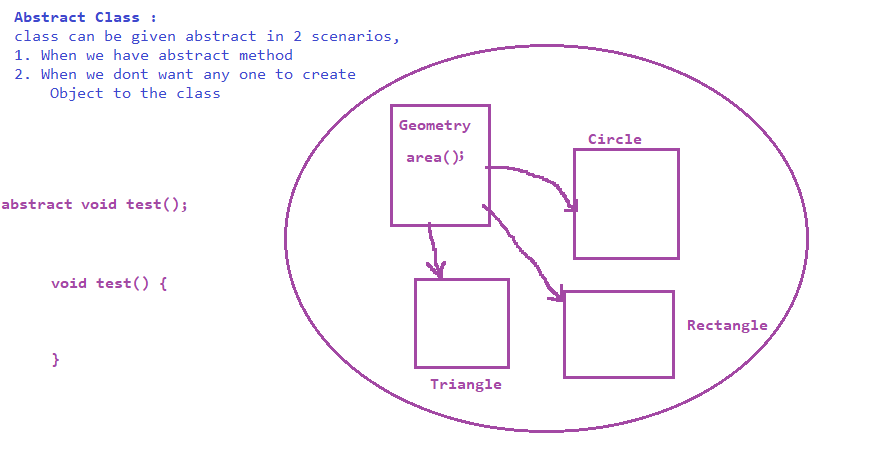
Access Specifiers

* private - private members are accessible only inside the class by all other members
* default – default members are accessible by all the classes within the same package
* protected – protected members are accessible by all the classes within the same package, outside the package it is accessible only if there is a inherited class (child class )
* public - wider access, it can be accessible by all the classes with in the project



Access modifiers

* static – class member, without creating a object we can access these members
* final –
  + variables : it acts as a constant value, no one can change / alter the value
  + methods : we cannot override
  + class : we cannot make it as a parent class
* abstract

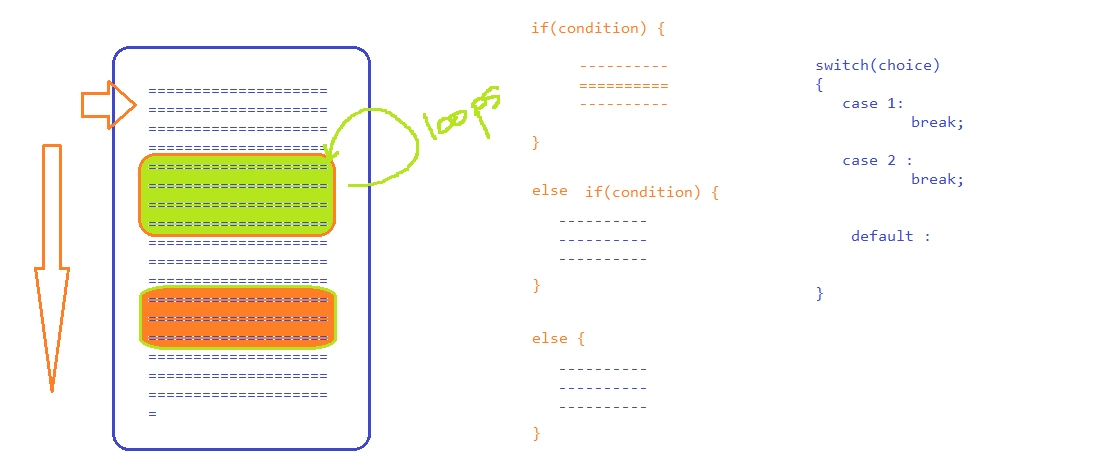


* + methods: when there is no method definition we declare method as abstract
  + class :
    - when we have any abstract method
    - when we want to restrict creating objects to the class
* synchronized –synchronized can be given to method so that we can restrict multiple threads entering to the method

Looping and Conditional Statements :

* for
* while
* do,while
* foreach / Extended for loop

Conditional statements



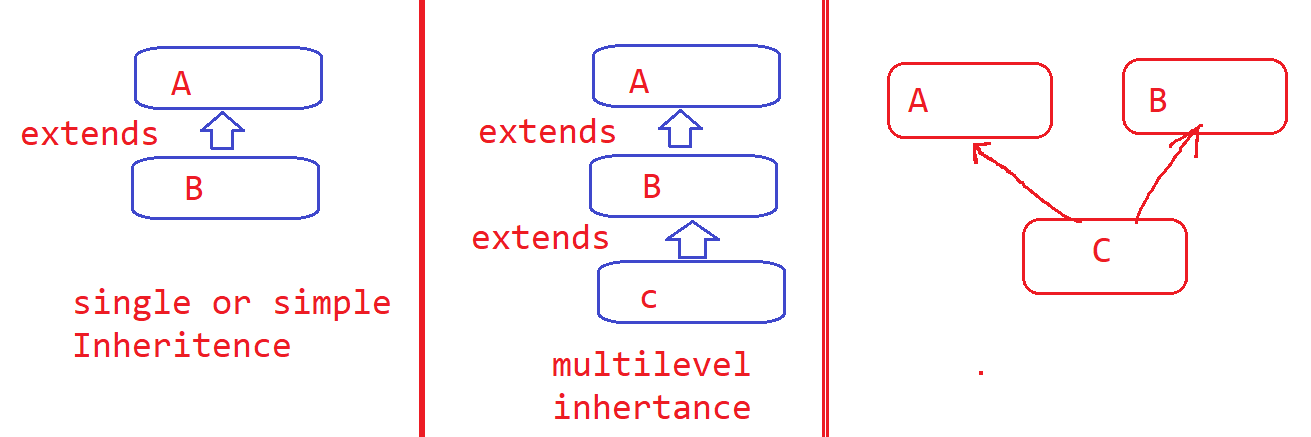
Strings :

- String functions

- String Buffer

- String Builder

Inheritance :



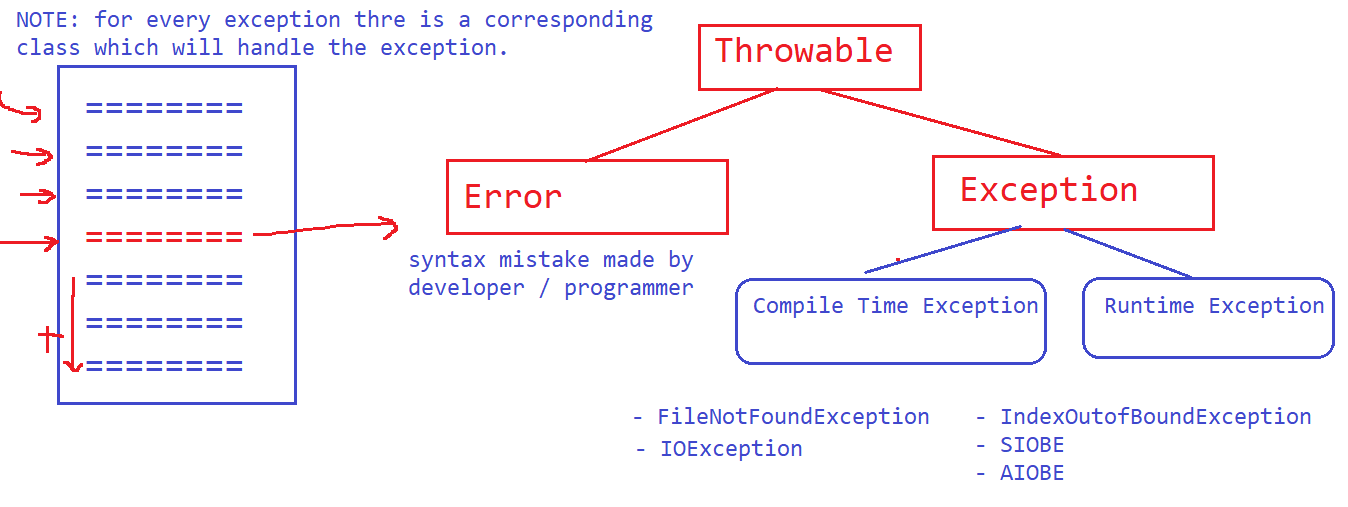
To achieve multiple inheritance we use interfaces.

NOTE : Interfaces are similar to classes, but interfaces are 100% pure abstract class.

Interfaces:

whatever method we declare inside interface is abstract by default and whatever the variable we declare inside the public static final by default

Exception Handling

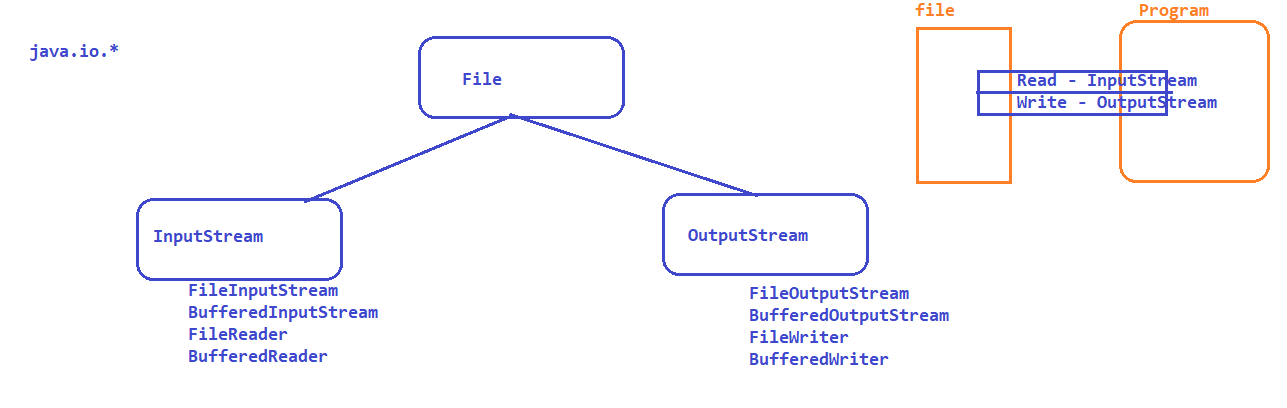


Exception Handling is a mechanism using which we can ask compiler to continue the execution till the last line even though there is a abnormal statement.

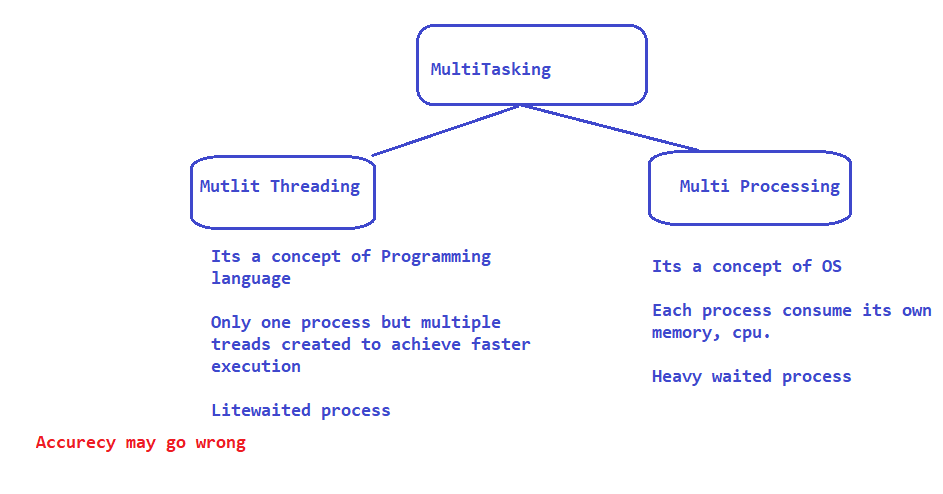
To handle exception we have to use try block and catch block

* try : abnormal statements can be kept inside try block
* catch : corresponding class which can handle the exception .
  + for one try we can keep multiple catch blocks
  + there should not be any valid java code between try and catch
* throws : whenever we don’t want to handle the exception from the place where it has occurred then we use thows. using throws, exception will go to the called place (Function call)
* throw : when we want to raise exception explicitly
* finally : to execute all the time ( exception raised or not raised )

File Handling in JAVA :



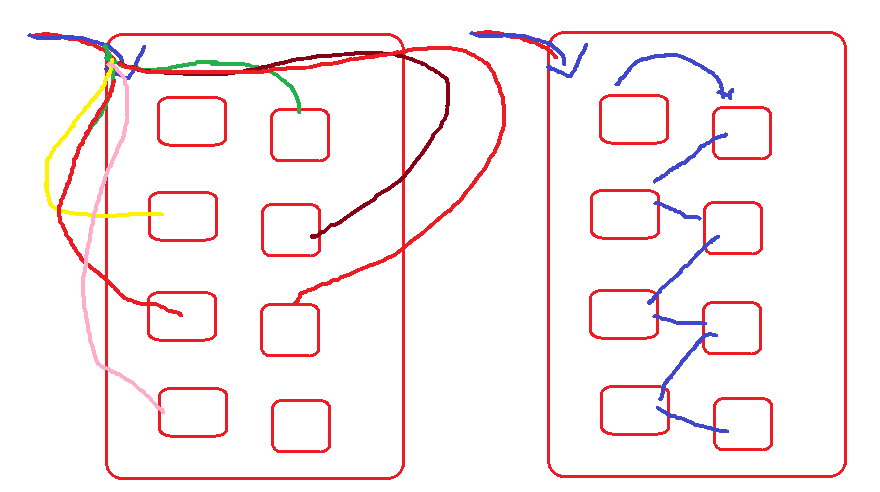
Threads :



Ways to create thread :

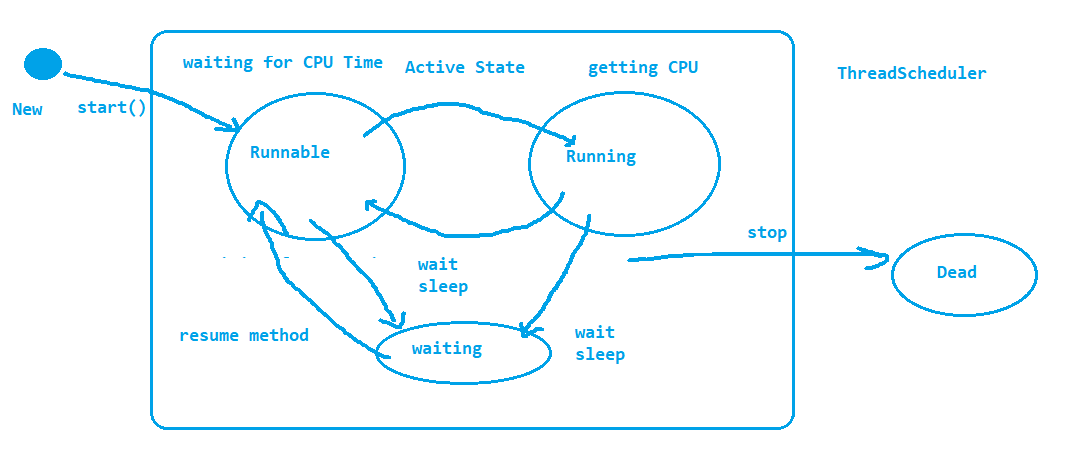
1. using Runnable interface

2. Using Thread class

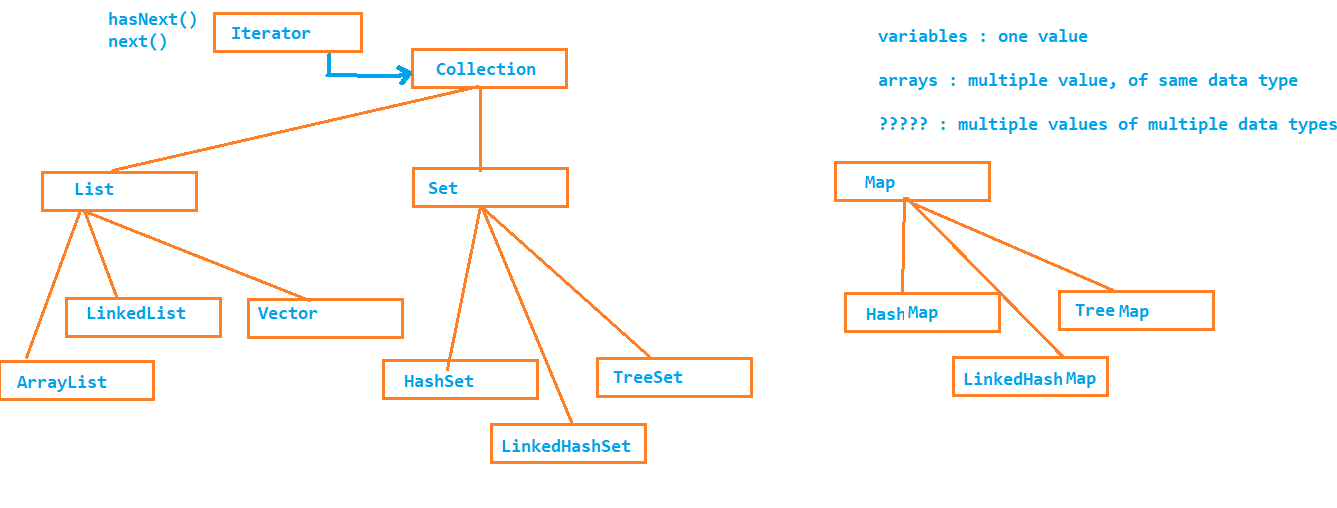


Implementing Threads in a program :

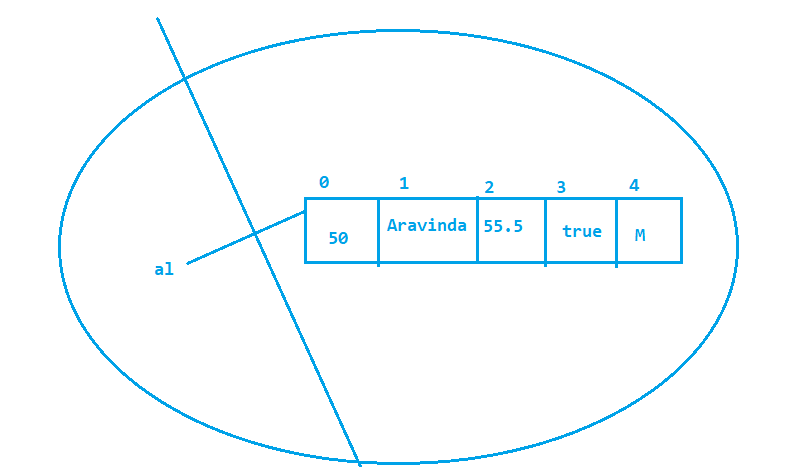
1. Write a class by making Thread class as a super class
2. Override run method and keep the thread task inside **run** method
3. From the main class create an object to Thread class (created by you ) and call **start** method



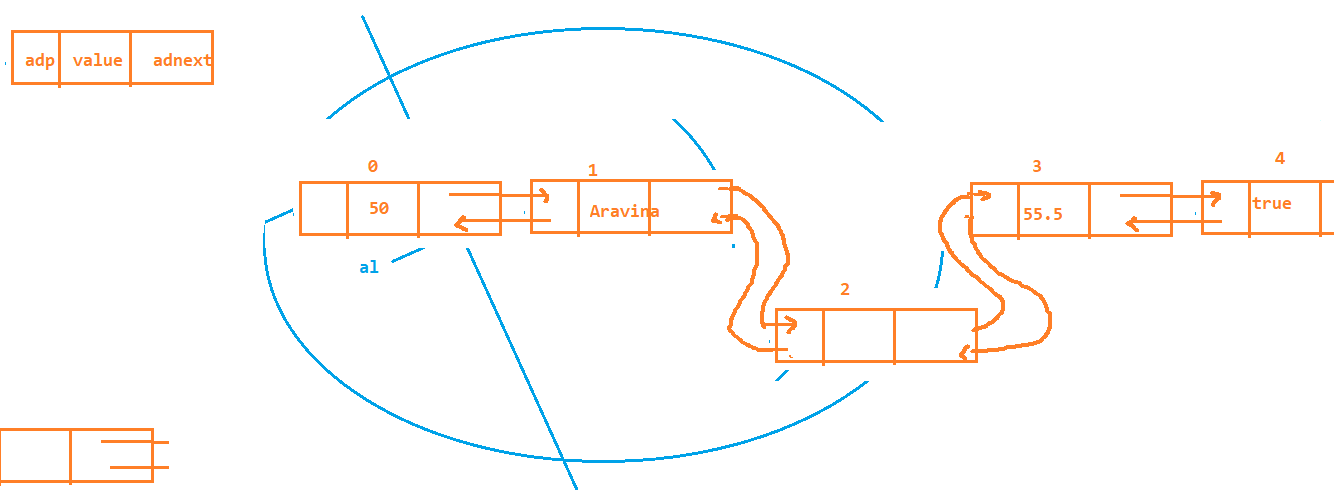
Collections :



ArrayList :

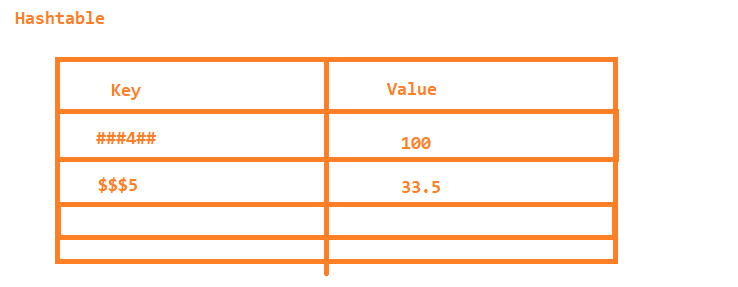


LinkedList :



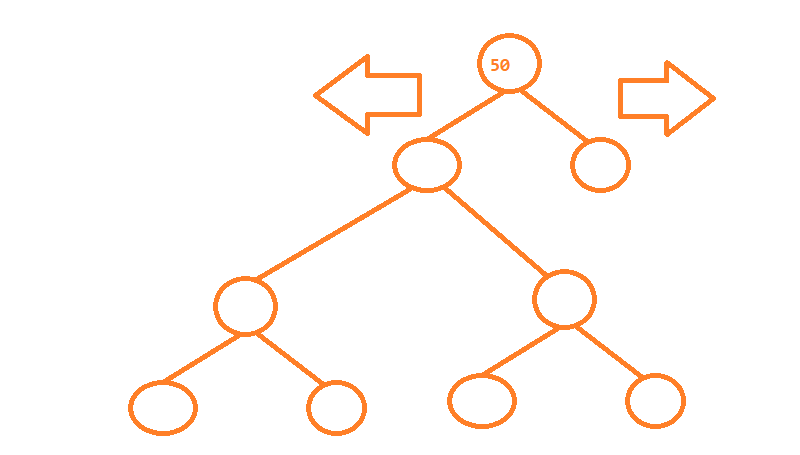
Set :

Hash Set :

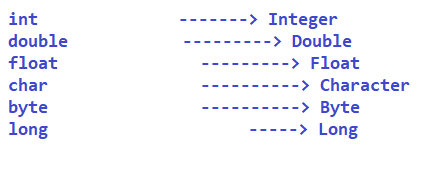


Linked HashSet

TreeSet

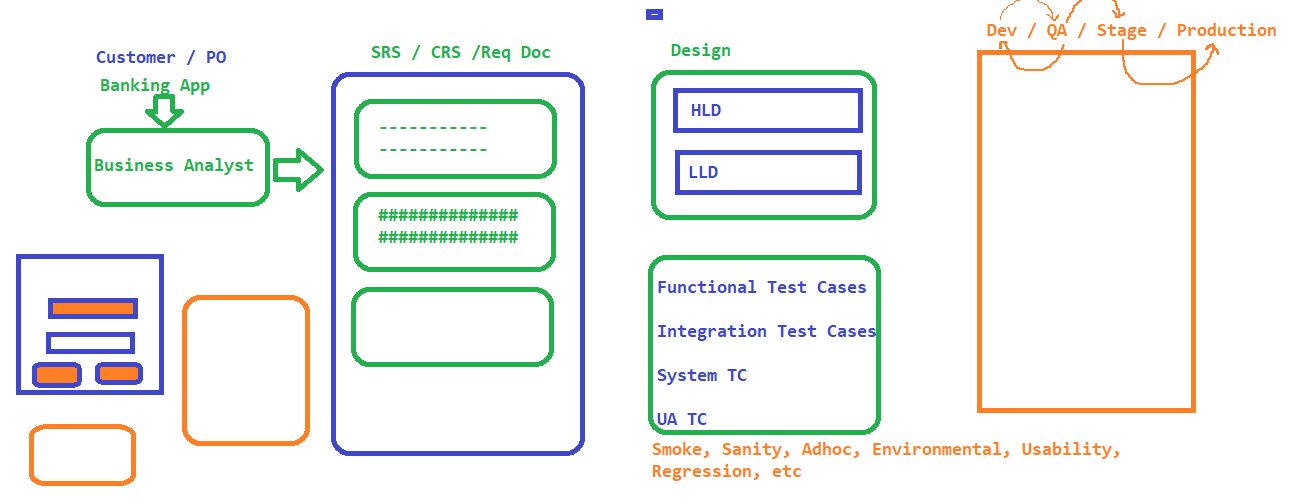


Wrapper class

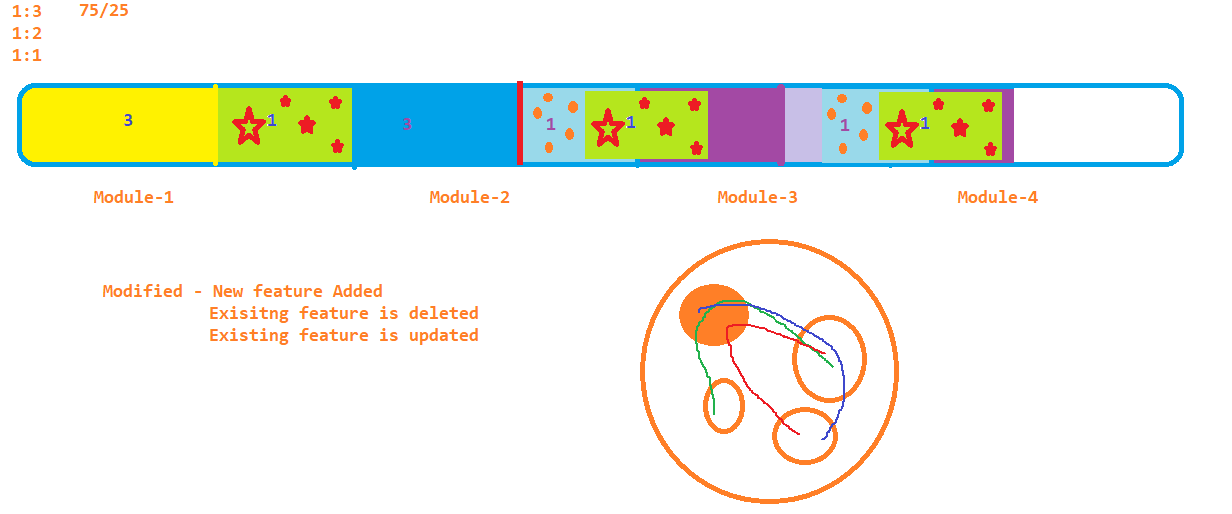


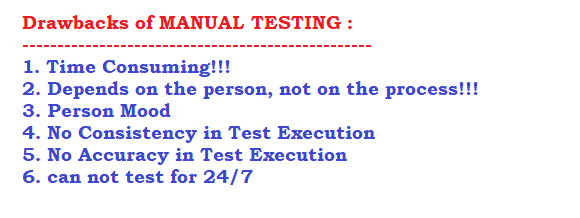
Testing ...????

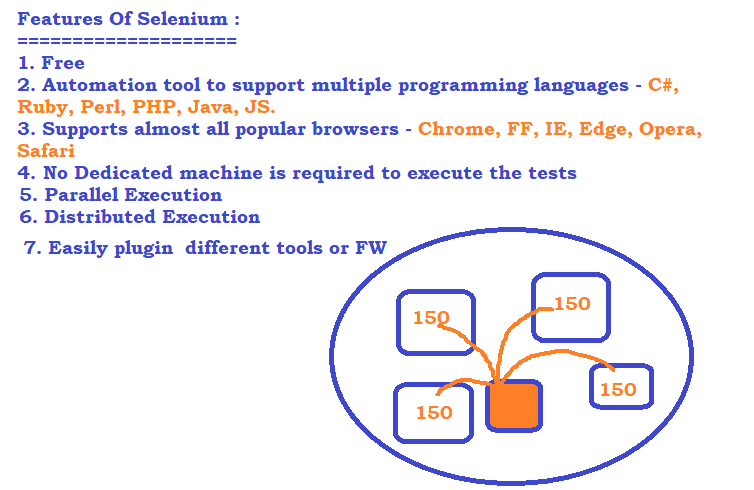
SDLC :

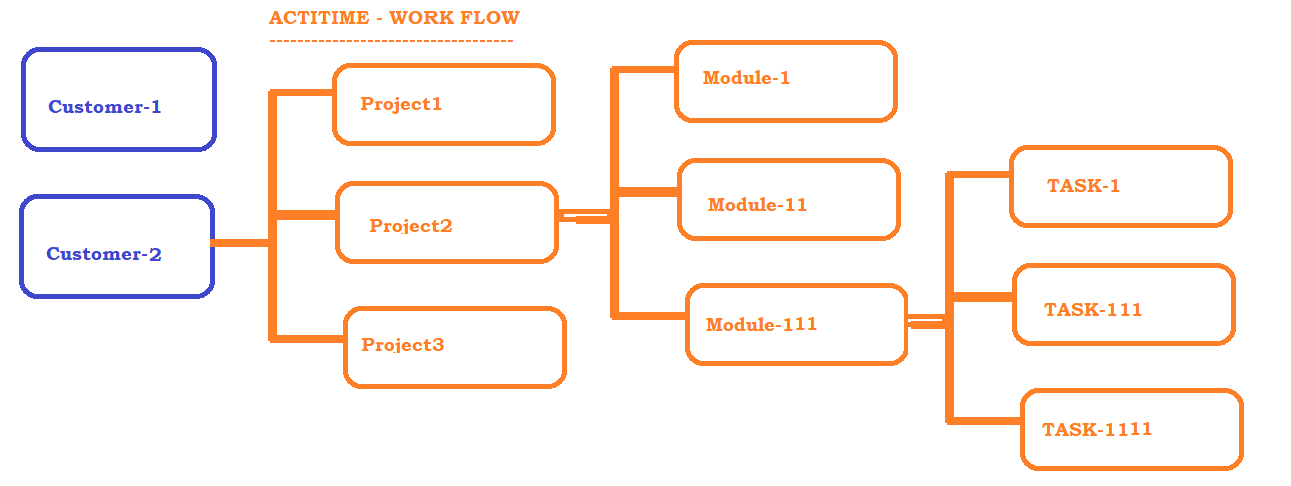


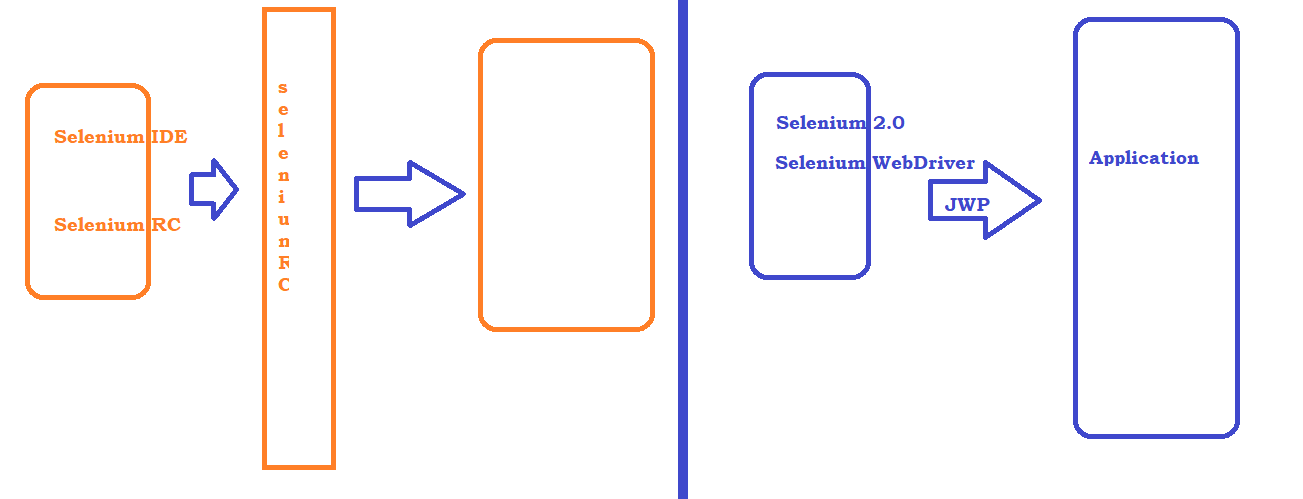
Manual Testing Process / Regression Testing





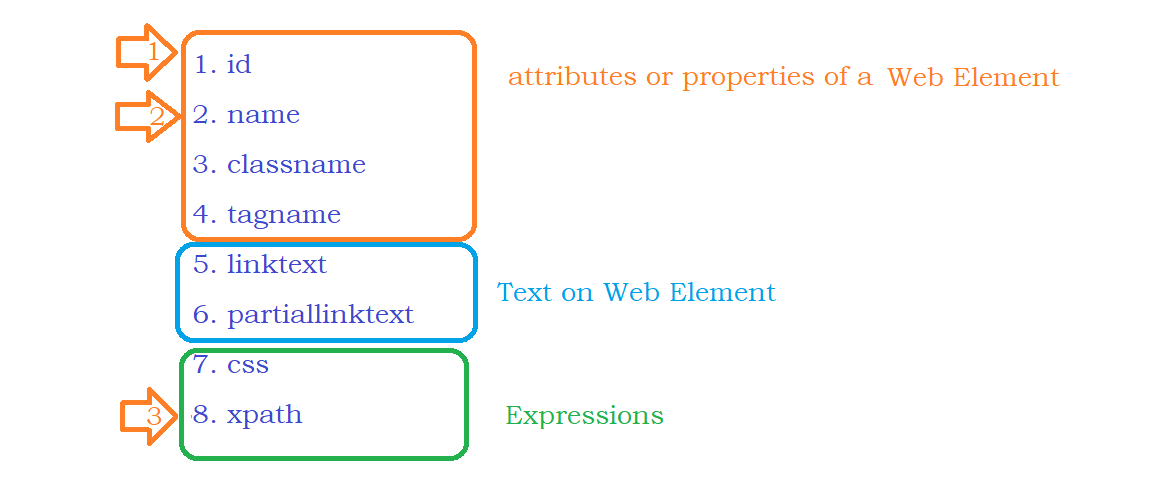




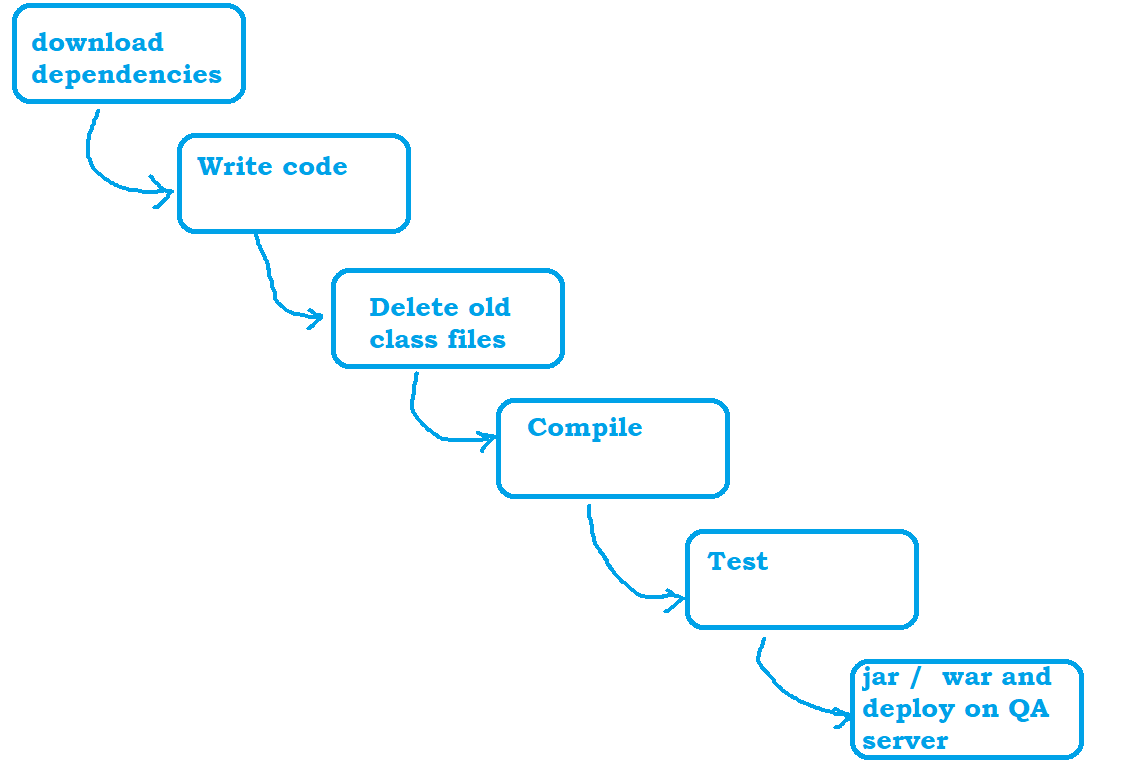


Identification Mechanism in Selenium :

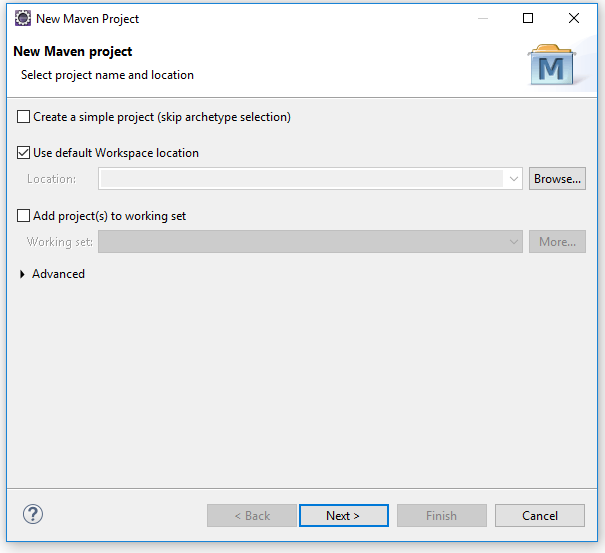
1. OPS Concept : id
2. namename
3. classname
4. tagname
5. linktext
6. partiallinktext
7. css
8. xpath

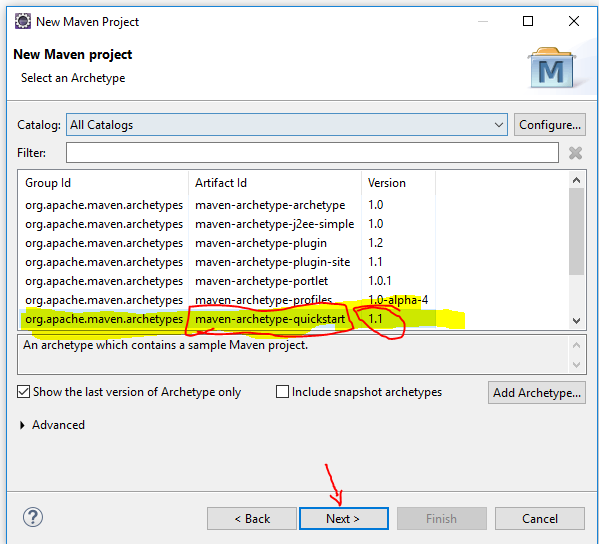


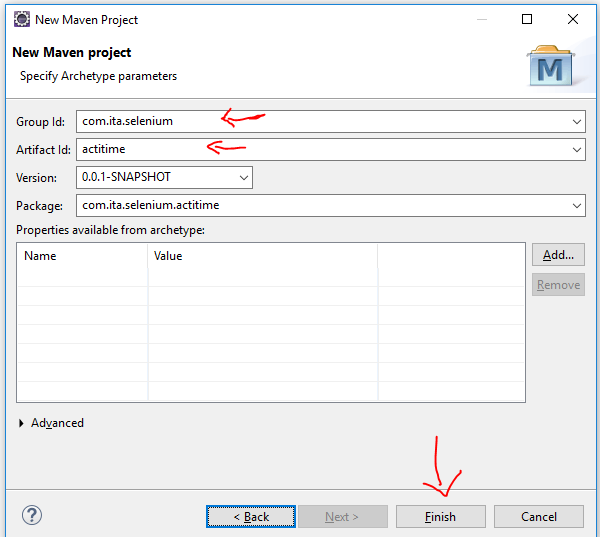
Build Automation Tool :



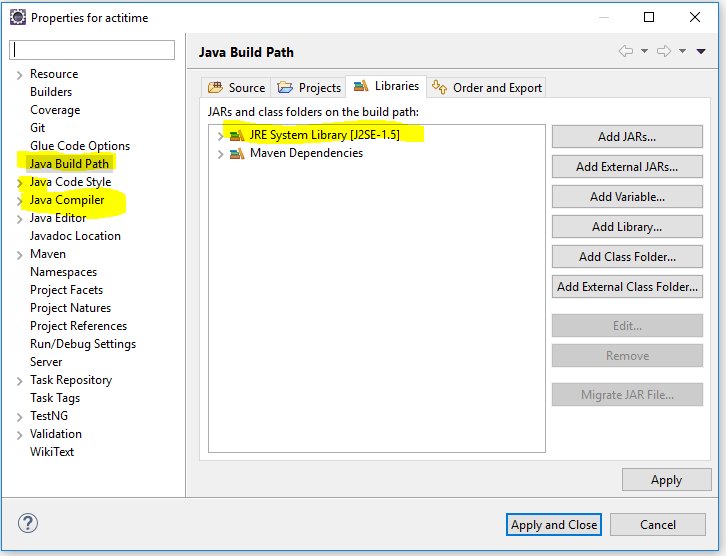
Create Maven Project :

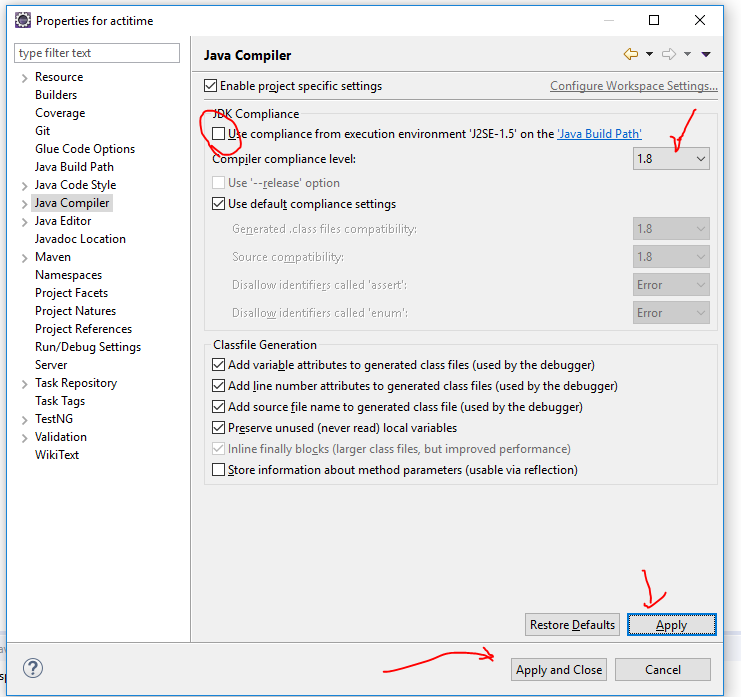
1. 

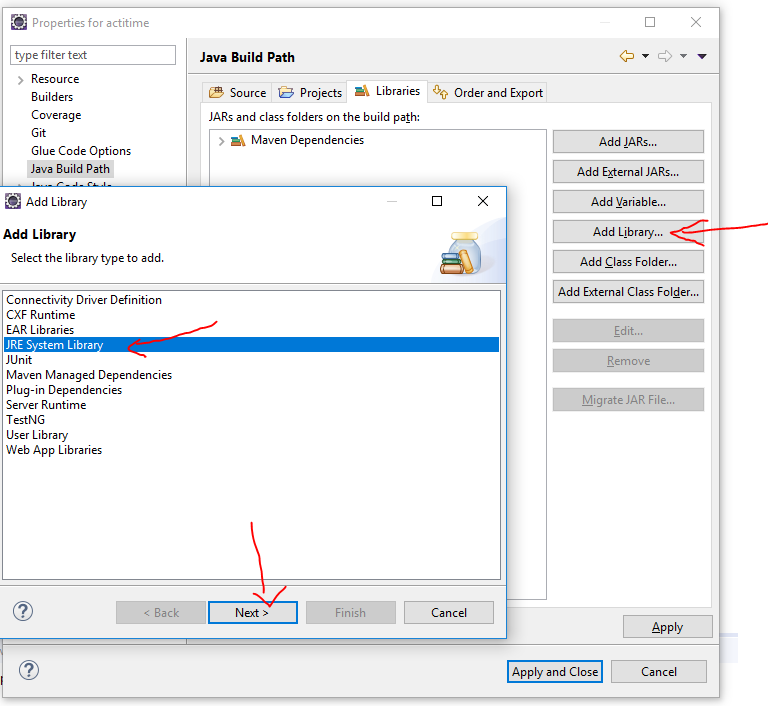
2. 

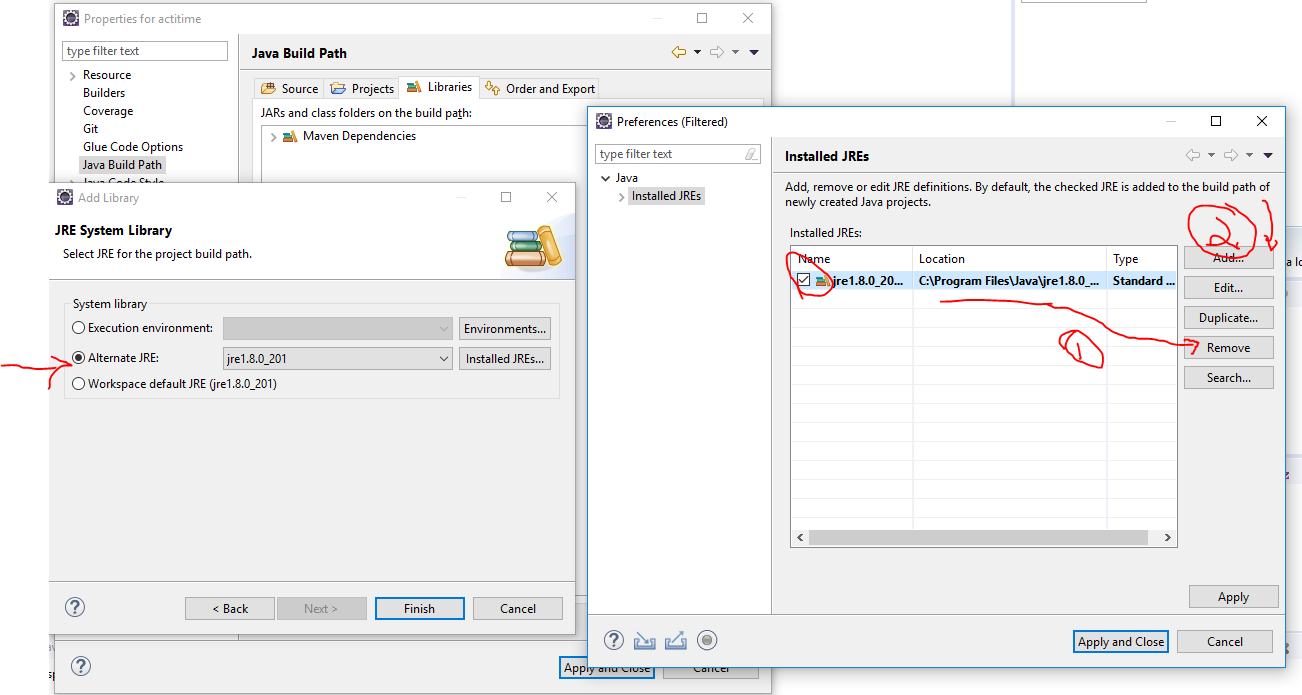
3. 

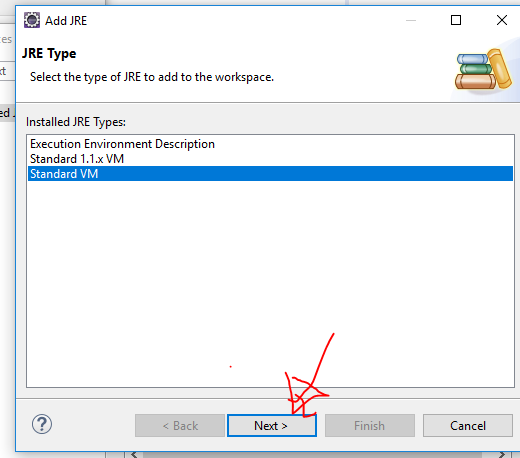
4. Update Compiler and Runtime (JRE) Environment : Right click on project and go to properties

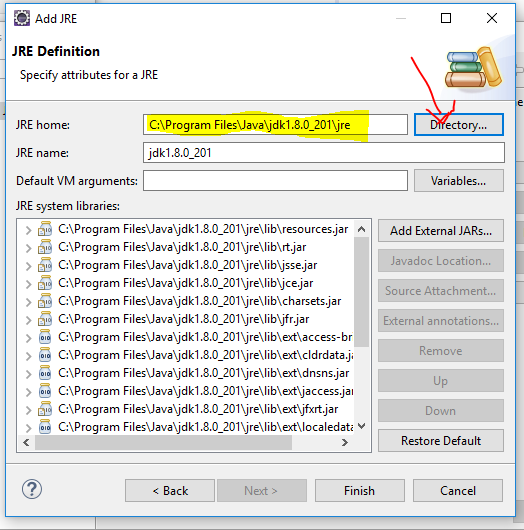
5. 

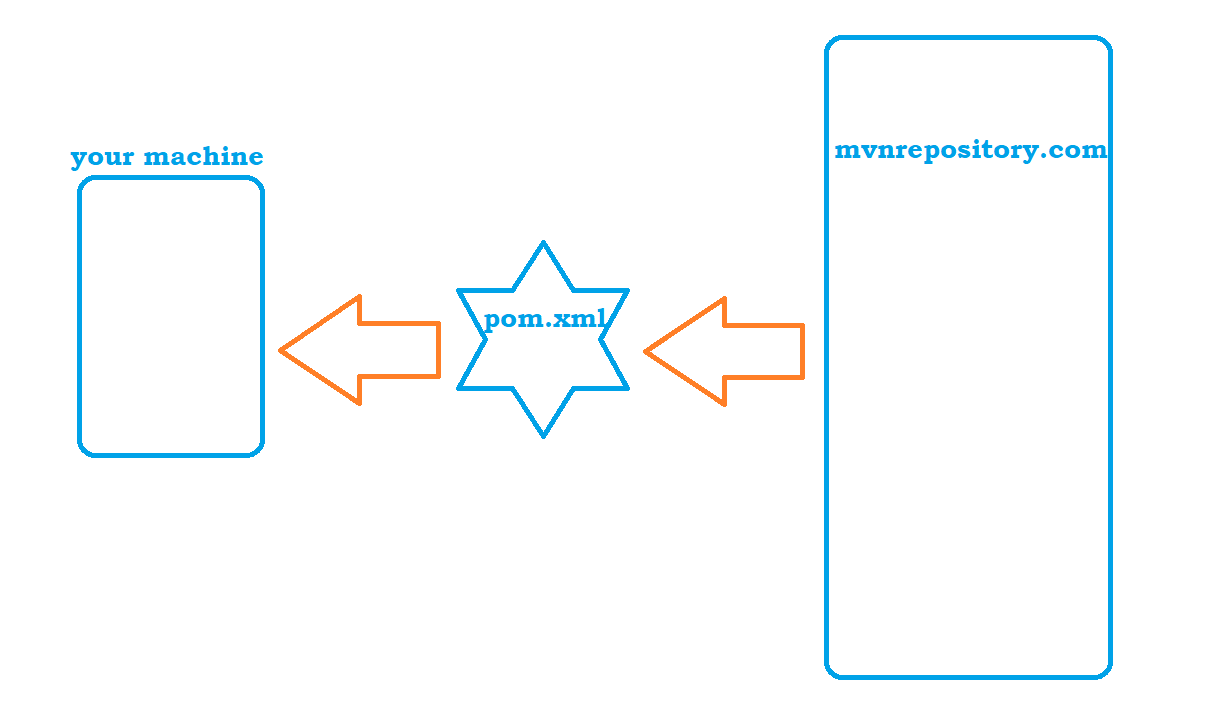




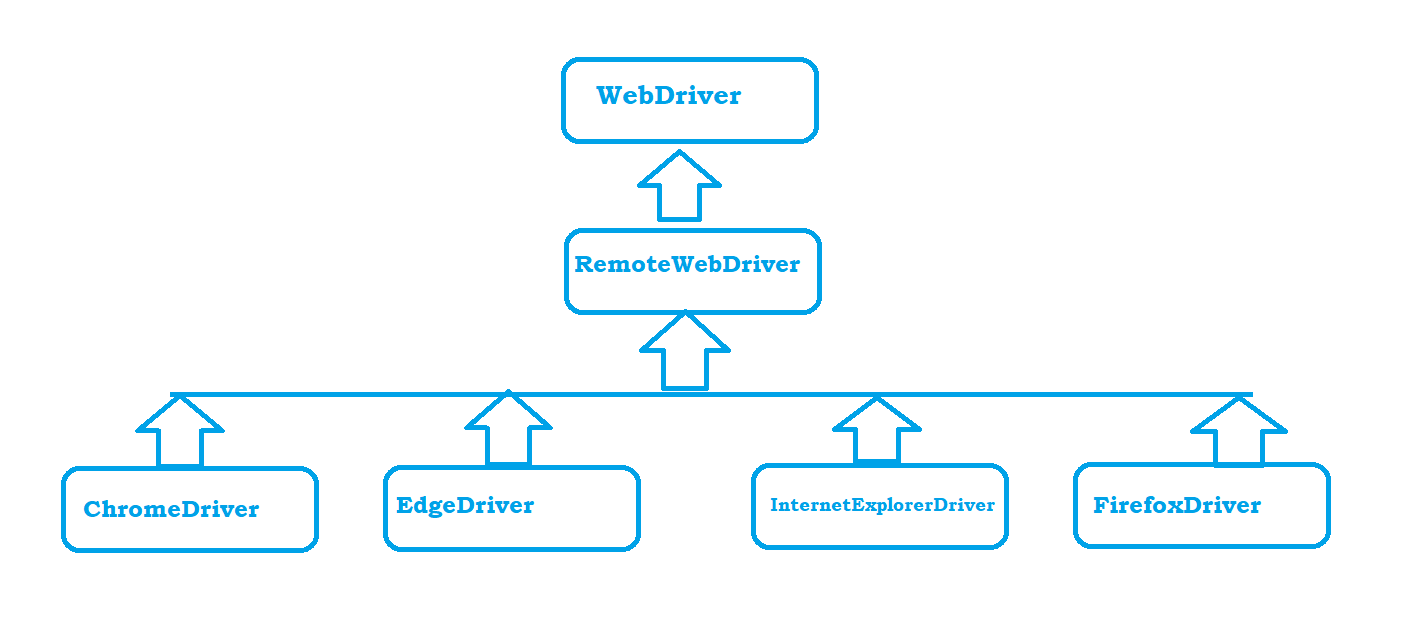




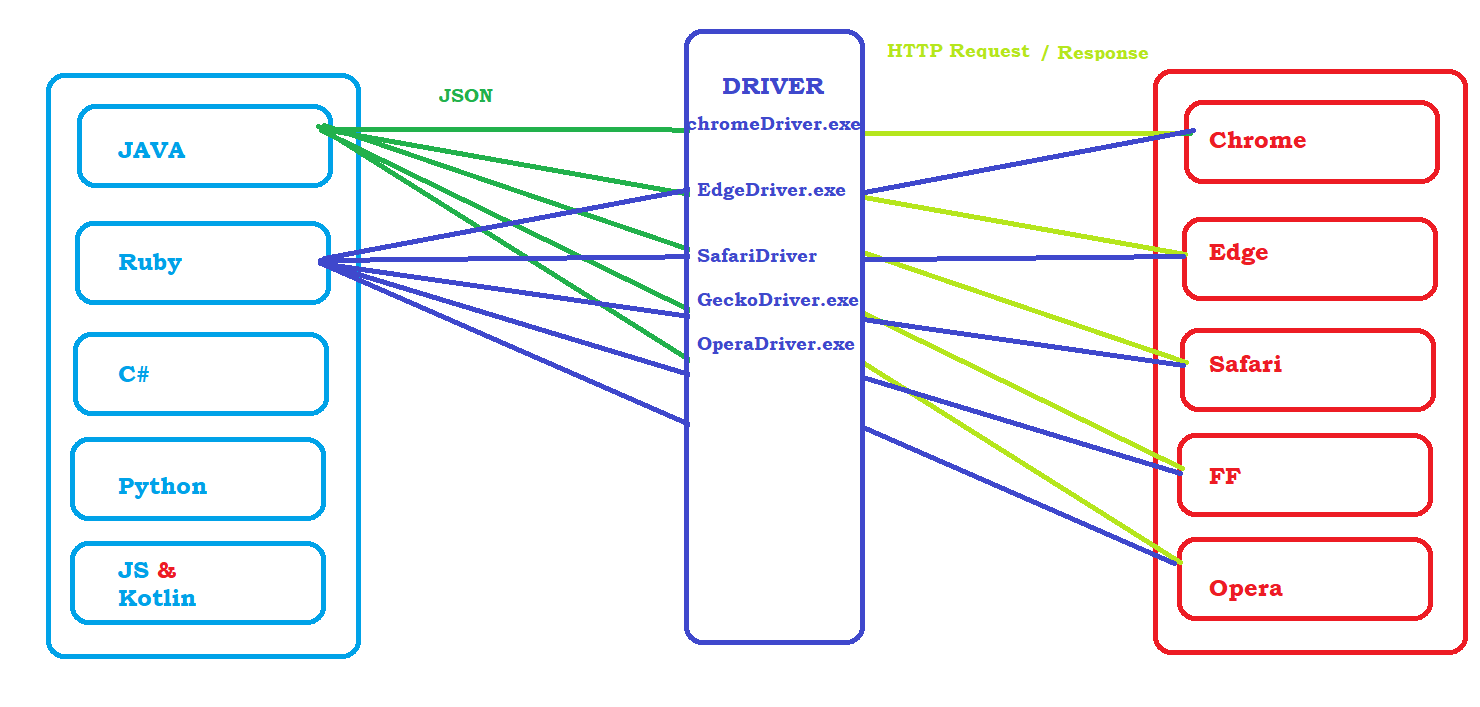




High level Architecture of Selenium



Work flow Architecture:



Expressions :

1. CSS

|  |  |  |
| --- | --- | --- |
| 1 | htmltag[attribute or property=’value’] | input[placeholder='Username'] |
|  |  | input[type='checkbox'] |
| 2 | Htmltag#idValue | input#keepLoggedInCheckBox |
| 3 | #idValue | #keepLoggedInCheckBox |
| 4 | HtmlTag.className | input.textField |
| 5 | .className | .textField |
| 6 | Expression > childtag name | a#loginButton > div |

1. Xpath
   1. Finding Element with Attributes
      1. //htmltag[@attribure='Value'] -> //input[@placeholder='Username']
   2. Logical Operators in Xpath
      1. AND
         1. //htmltag[@attribute1=’value1’ and @attribute2 = ‘value2’]
         2. //input[@type='text' and @placeholder='Username']
      2. OR
         1. //htmltag[@attribute1=’value1’ or @attribute2 = ‘value2’]
         2. //input[@type='text' or @placeholder='Username']
      3. NOT
         1. //htmltag[not( @attribute1=’value1’)]
         2. //input[not(@type='text')]

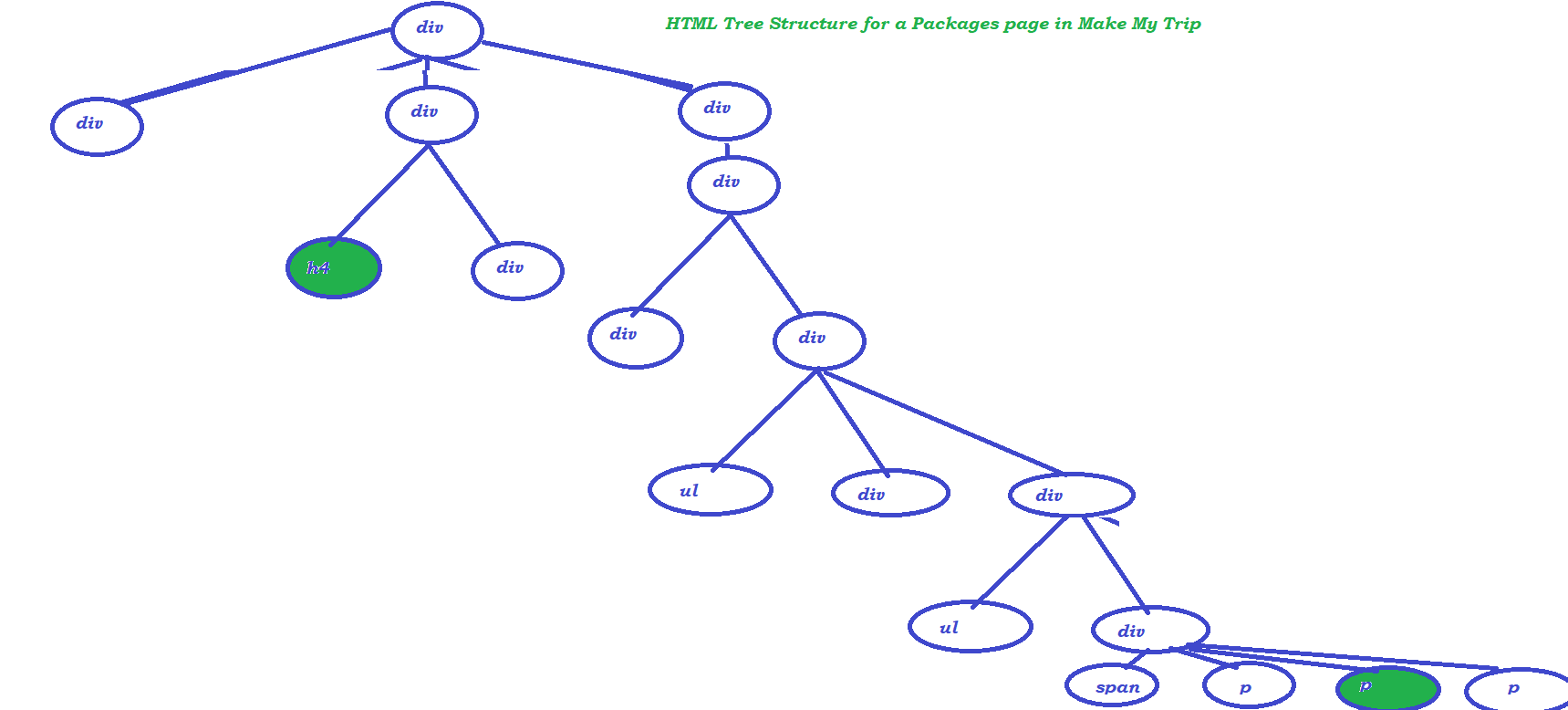
//td[(@class='wd day' or @class='current day' or @class='we day') and text()='3']

//td[not(@class='past day') and text()='4']

* 1. Xpath Using Functions
     1. text()
        1. //htmltag[text()=’ExactValue’] ---- > //td[text()='4']
        2. //label[text()='Keep me logged in']
     2. contains(arg1, arg2)
        1. arg1 can be attribute or text function
        2. arg2 can be partial text which is present on the element
        3. //htmltag[contains(arg1, arg2)]
           1. //label[contains(text(),'Keep')]
           2. //img[contains(@src,'timer')]
     3. starts-with(arg1, arg2)
        1. arg1 can be attribute or text function
        2. arg2 can be text which starts with
           1. //span[starts-with(text(),'Watch')]
           2. //span[starts-with(text(),'iPhone 13')]
  2. Traversing from parent to Child
     1. //a[@id='loginButton']/div
     2. //td[@id='loginButtonContainer']//div[text()='Login ']
  3. Traversing from Child to Parent***NOTE****: Whenever we are playing with dependent and independent elements we have to traverse from child to parent*
     1. Step1 : write a xpath to child element
     2. Step2 : put the complete child xpath in a square bracket and specify parent html tag
     3. //tr[th[text()='Directed by']]//a
     4. //tbody[tr[th[text()='Body']]]//td[@class='nfo']

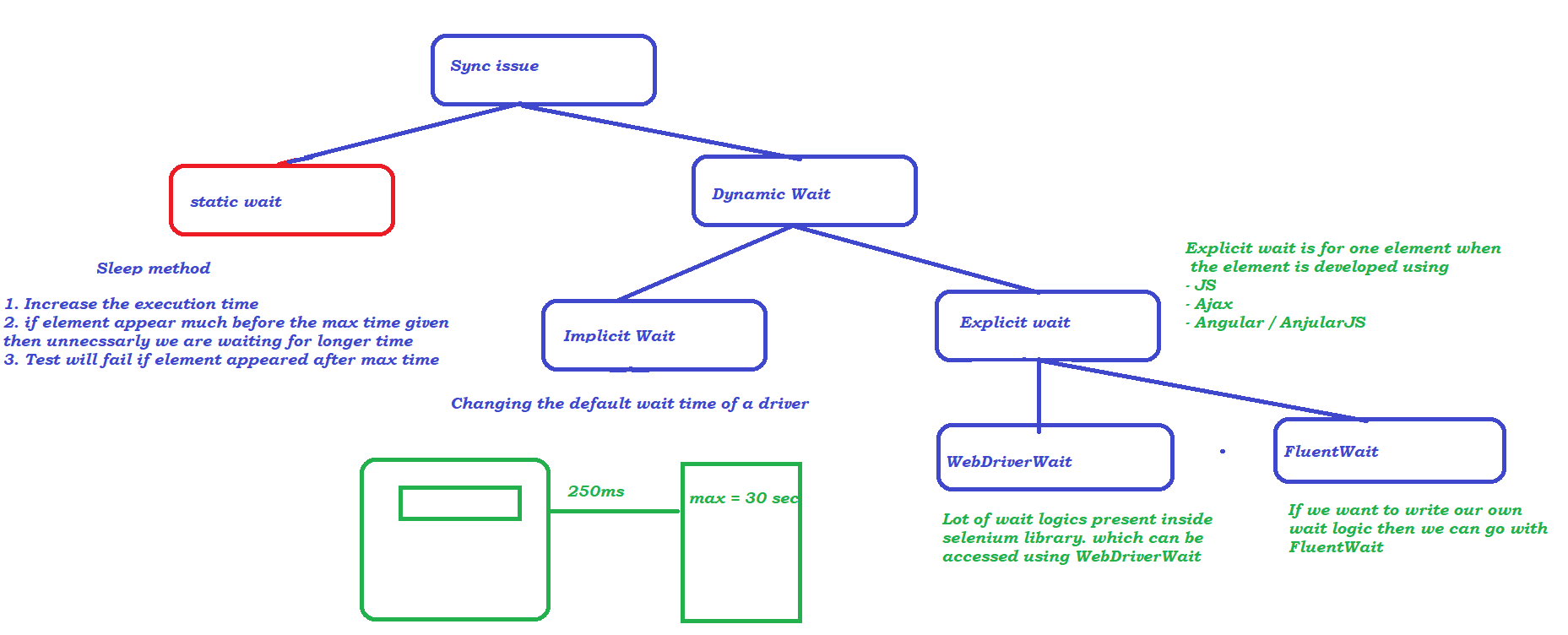
MMT - //div[div[h4[text()='Udaipur & Mount Abu - Free Cancel...']]]//p[contains(@class,'price-current')]

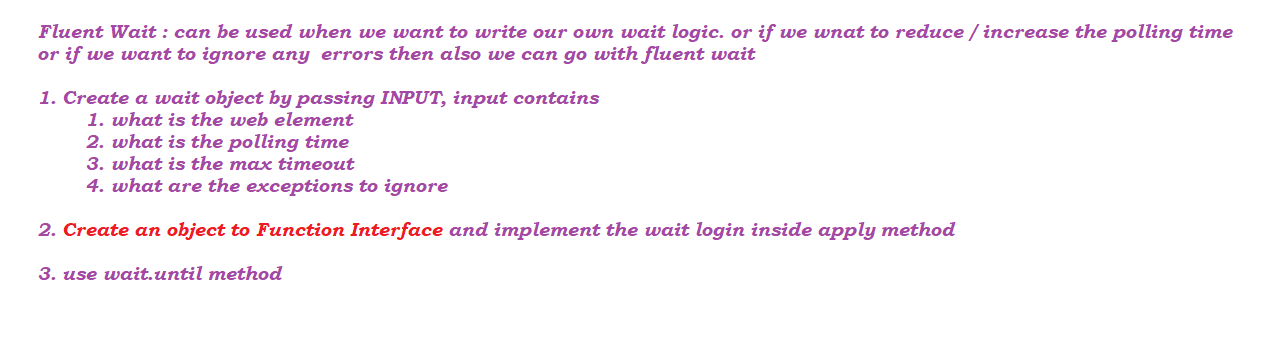
Using AXES Function : //h4[text()='Udaipur & Mount Abu - Free Cancel...']/ancestor::div[@class='itemCard packageCard']//p[contains(@class,'price-c')]

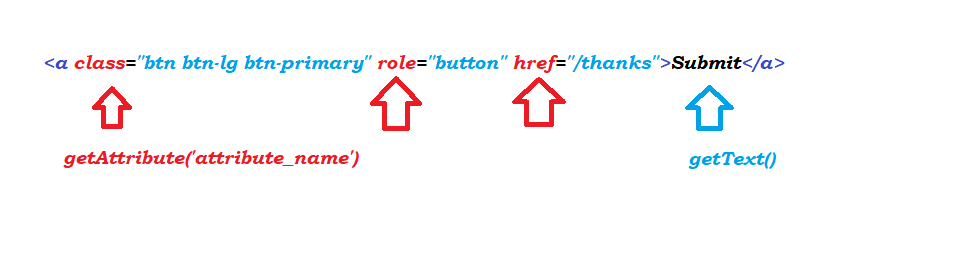


* 1. Axes Functions
     1. Following Sibling
        1. xpathOfaElement/following-sibling::siblingTag
        2. //th[text()='Directed by']/following-sibling::td
     2. Preceding sibling
        1. //td[a[text()='Technology']]/preceding-sibling::th
     3. Following
        1. //table[tbody[tr[th[text()='Display']]]]/following::table
     4. Preceding
        1. //table[tbody[tr[th[text()='Display']]]]/preceding::table
     5. Parent
        1. //th[text()='Display']/parent::tr/parent::tbody/parent::table/following::table
     6. Child
        1. //th[text()='Display']/parent::tr/parent::tbody/parent::table/child::tbody/child::tr/child::th/following-sibling::td
     7. Ancestor
        1. //th[text()='Display']/ancestor::table
     8. //th[text()='Display']/parent::tr/parent::tbody/table

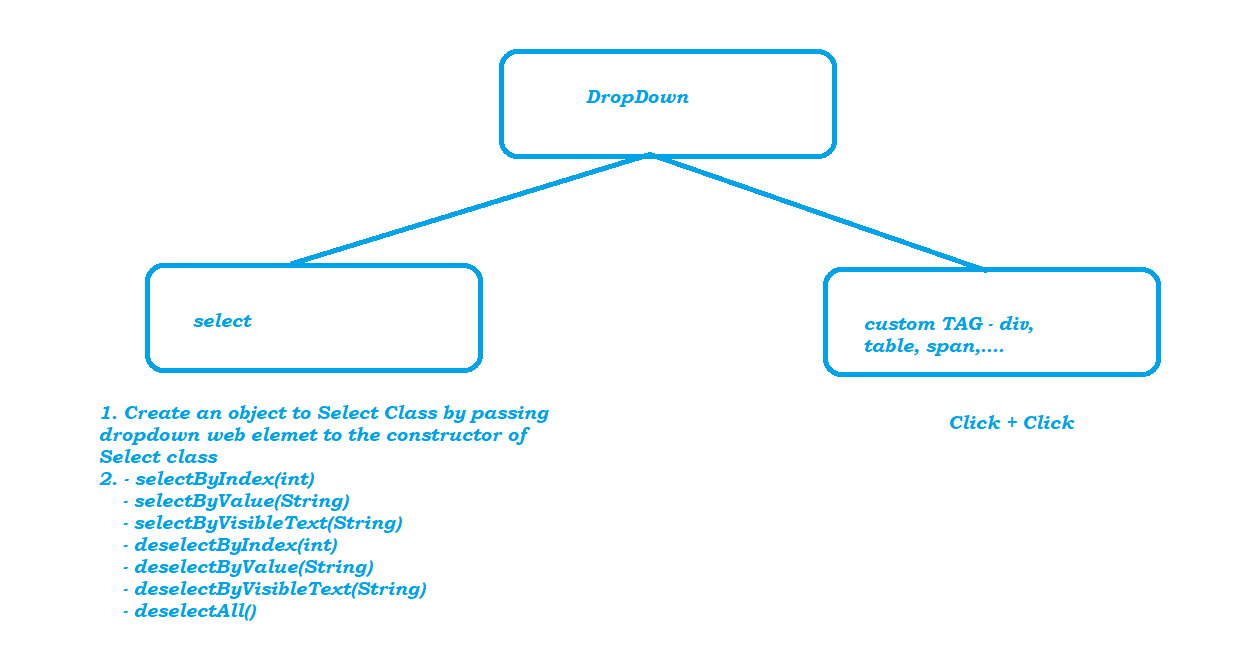
1. Auto Suggestions
2. Sync issue or synchronization issue



3. Tool tip information



4. Drop Downs



5. Browser Operations

* maximizing the browser
* clicking on back
* forward
* refresh
* navigate.to()

6. Automate End-End Test

* Execute the test case Manually couple of times
* Understand the data required to automate
* Identify the reusable functions
* Identify the places to keep validation statements
* Start with writing reusable functions
* Write a Test by calling the function written

7. Executing Test on a different Browser

* We downloaded the driver executable
* Before creating a Browser Object we set the executable path using System.setProperty(key, value)

OR

we can automate the above process by WebDriverManager

8. Actions

9. End To End Test

10. PopUps

11. Data driven testing

12. TestNG

13. Selenium Grid

14. POM

15. BDD-Cucumber

16. GIT – Jenkins