+91-9945042504

Name –

Exp / Automation Background –

**JAVA (work exp / knowledge / new)**

**Expecting from this course**

**Course Content**

**Pre – Java**

1. **Basics of Java**

* Oops Concept( Abstraction, encapsulation, polymorphism & inheritance)
* JDK – Java development kit
* JRE – Java Runtime Environment
* JVM – Java Virtual Machine
* Editor – **eclipse**, netbeans, intellij,....

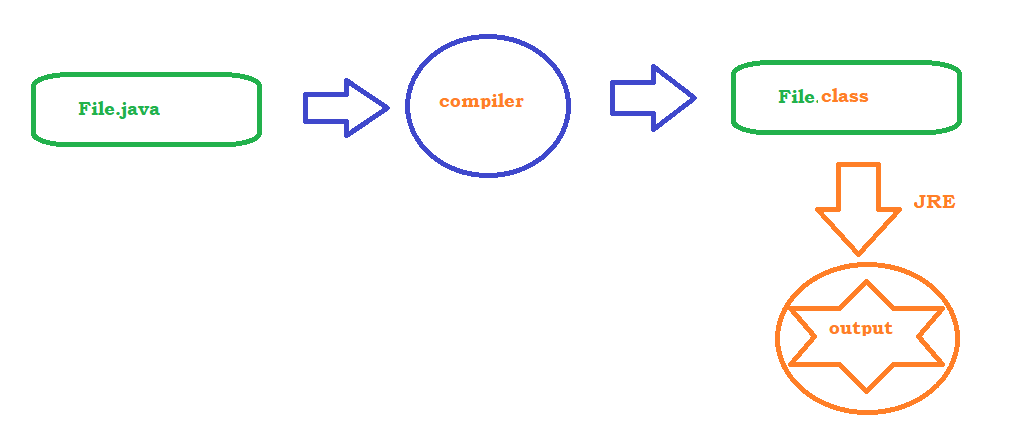
1. **Building blocks of Java programming**
   * **class**
     1. variables
     2. methods
     3. constructors
     4. static initialization blocks
     5. instance initialization blocks
   * **interface**
   * **abstract class**
   * **packages**
   * **looping statements**
     1. for
     2. while
     3. do,while
     4. foreach
   * **conditional statements**
     1. if
     2. if,else
     3. if,elseif, else
     4. switch
2. **Access Specifiers / Access Modifiers (Non functional Access Specifiers)**
   * private
   * default
   * protected
   * public
   * static
   * final
   * abstract
   * synchronized
3. **Data types**
   * primitive data types ->
     1. byte
     2. short
     3. int
     4. long
     5. float
     6. double
     7. char
     8. Boolean
   * derived data types ->
     1. Array
     2. Objects
     3. String
4. **String** 
   * functions available in string class
5. **Threads**
   * Multitasking – Thread
   * Multitasking – Runnable
6. **File Handling – java.io.\***
   * Reading File (properties file and text file )
   * Writing File
7. **Exception Handling**
   * try
   * catch
   * throw
   * throws
   * finally
8. **Collections – java.util.\***
   * List
     1. ArrayList
     2. LinkedList
     3. Vector
   * Set
     1. HashSet
     2. TreeSet
     3. LinkedHashSet
   * Map
     1. HashMap
     2. TreeMap
     3. LinkedHashMap
9. **Generics**

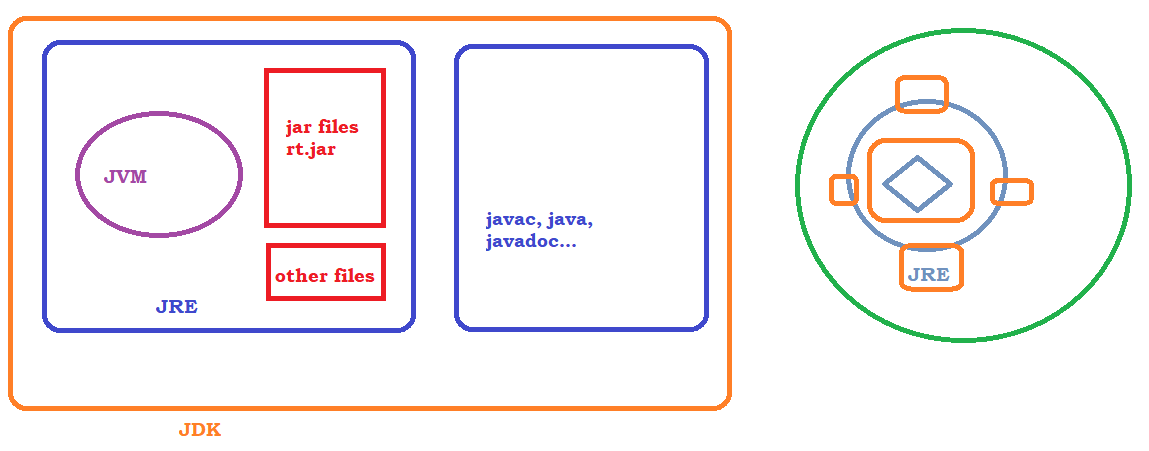
**Selenium**

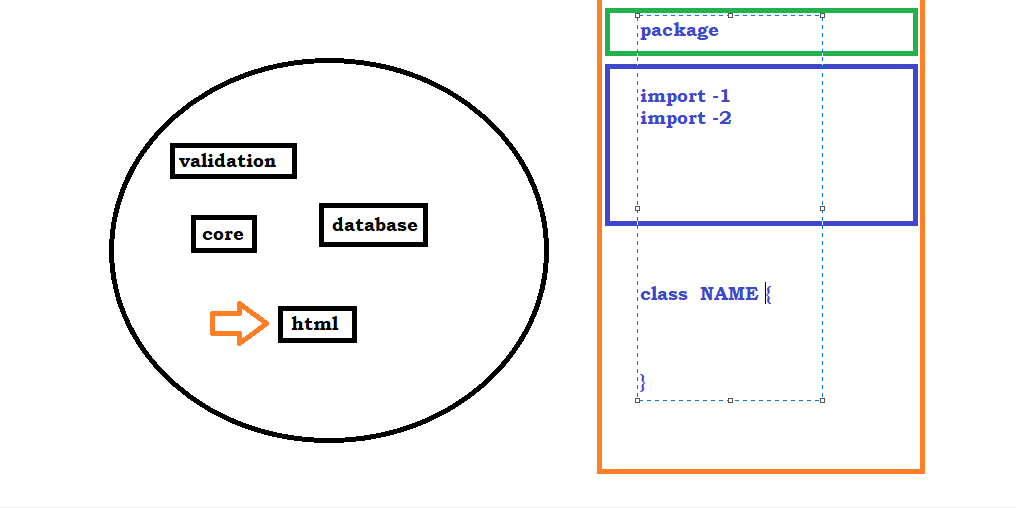
* Drawbacks of manual testing
* Why Automation is needed???
* What are the different tools we have in market, Why Selenium is popular ?
* History of Selenium.
* Versions of Selenium
  + Selenium IDE ( Only for beginners )
  + ~~Selenium RC~~
  + Selenium Webdriver / Selenium 2.0 / 3.0/4.0 alpha
    - identify the elements (8 ways)
    - performing operations ( type, click, select, read text....)
    - Handling sync issues
      * static
      * dynamic
    - Actions
    - Select
    - switchto()
    - dropdowns
    - alerts
    - popups
    - Reusable functions
    - Automate end-to-end test
    - Execute tests with multiple set of test data
  + Selenium Grid
* Frameworks
  + TestNG
    - annotations
    - execute the tests
    - groups
    - skip
    - parallel execution
  + Page Object Model
* **Maven**/ Gradle – Build Automation Tool
* **Introduction to Appium**

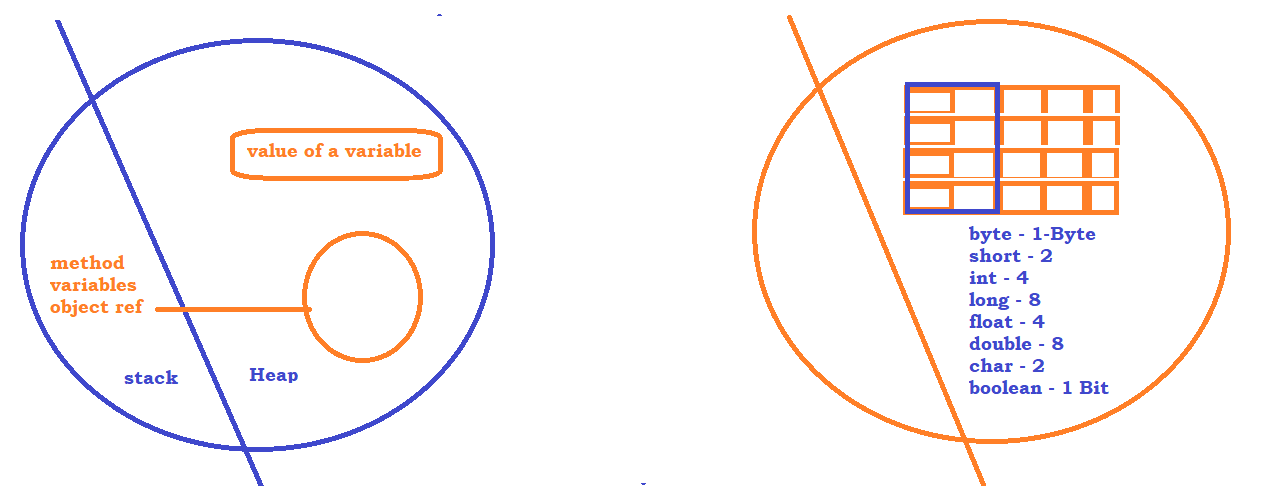
**---------------------------------------------------------------------------------------------------**

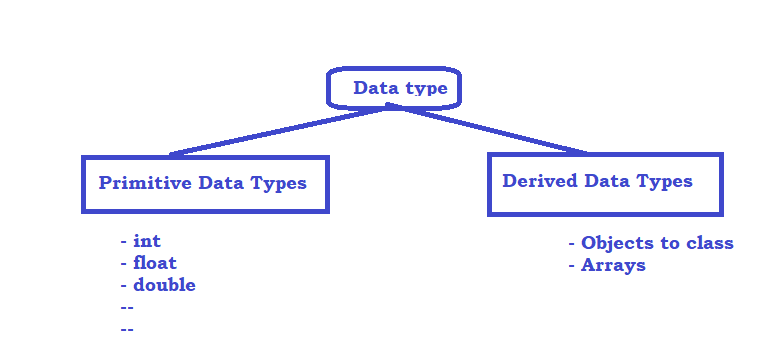
**JAVA –**

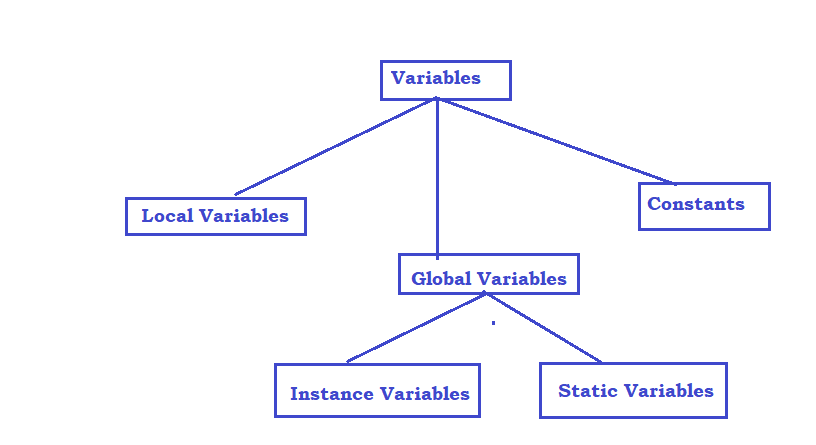
****

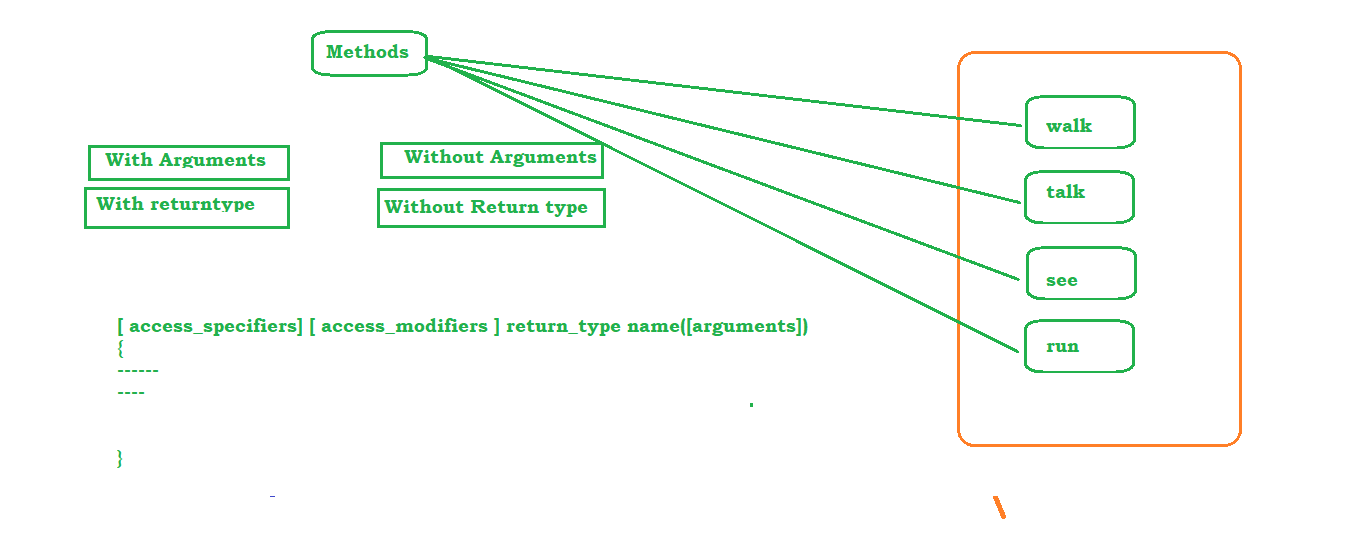
****

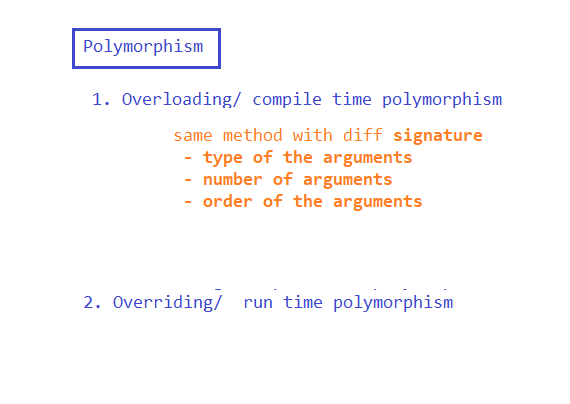
****

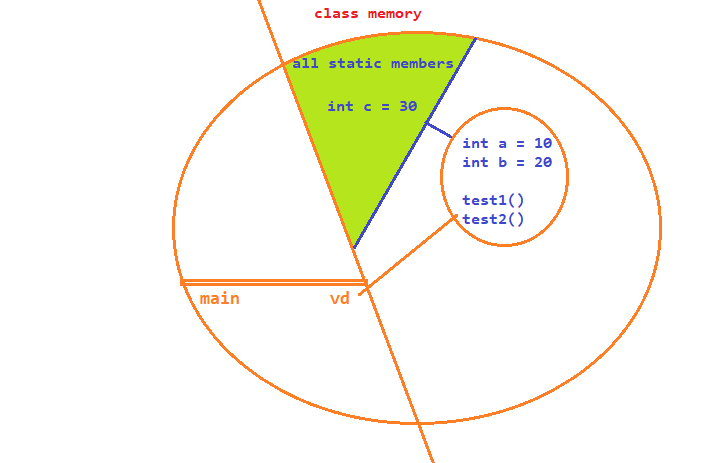
****

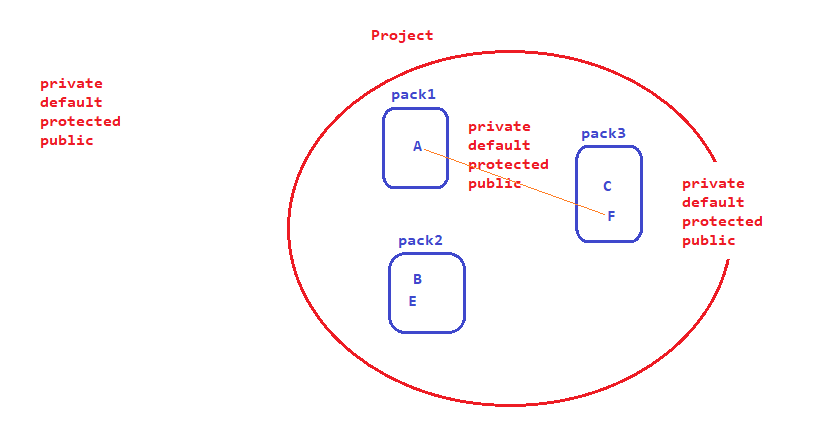
****

****

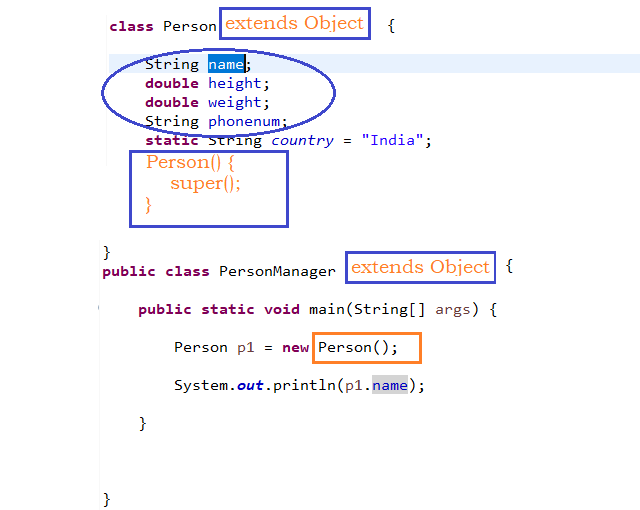
****

****

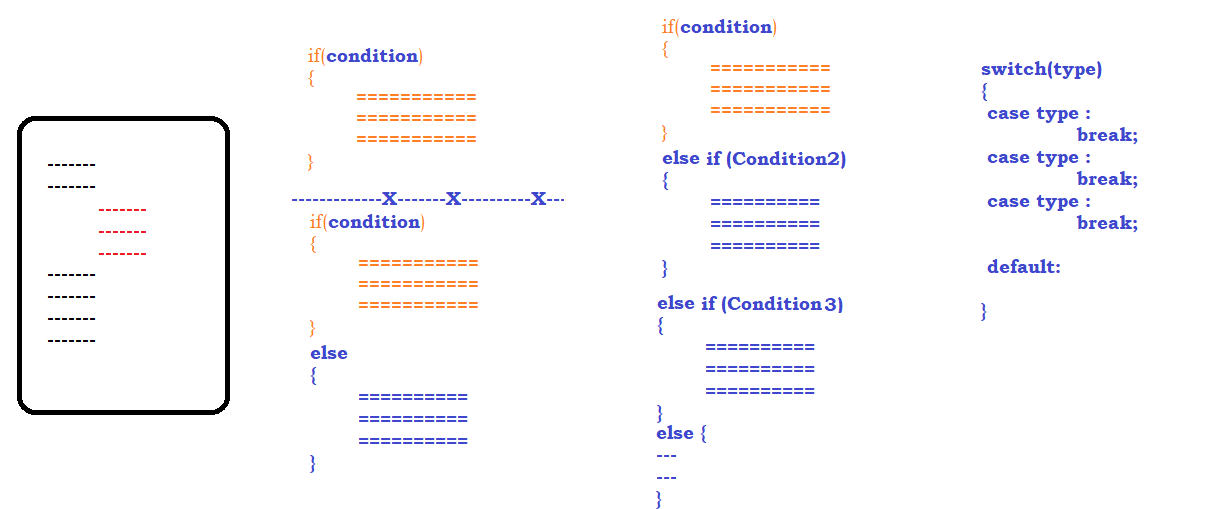
****

****

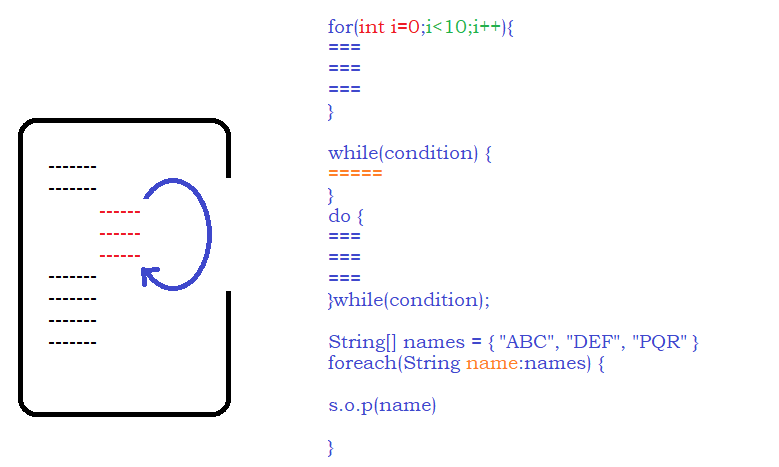
**Constructors :**

****

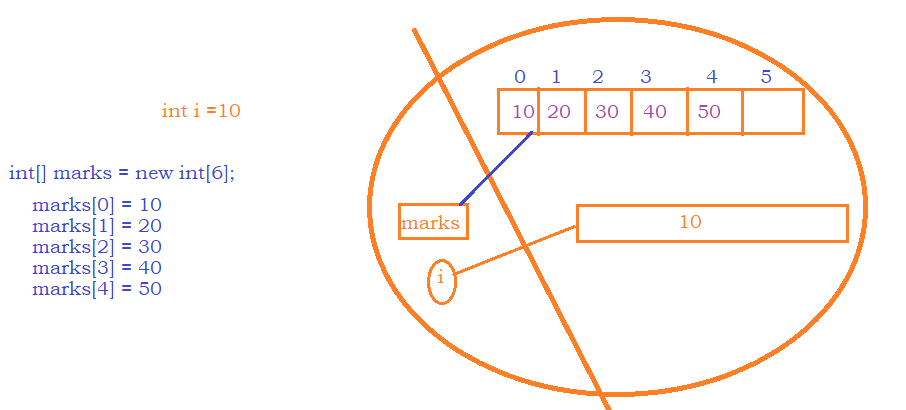
**Looping / Conditional :**

****

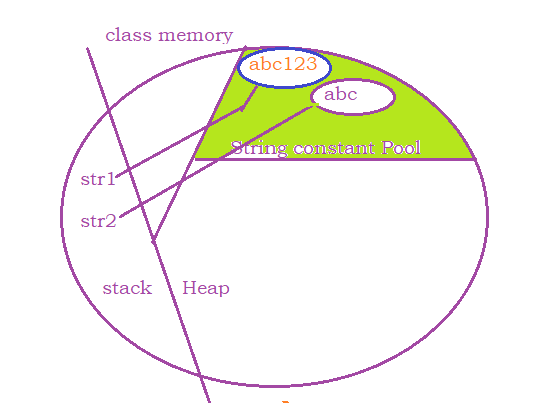
**Looping:**

****

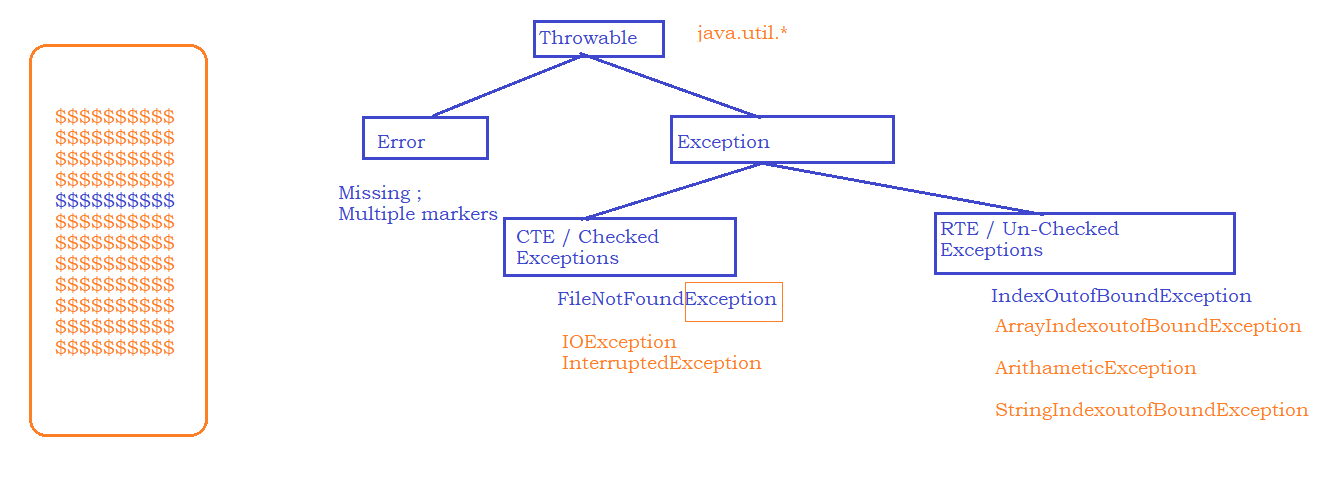
**Array:**

****

**Strings :**

****

**Exceptions :**

****

**Exception Handling ????**

**try {**

**------**

**------**

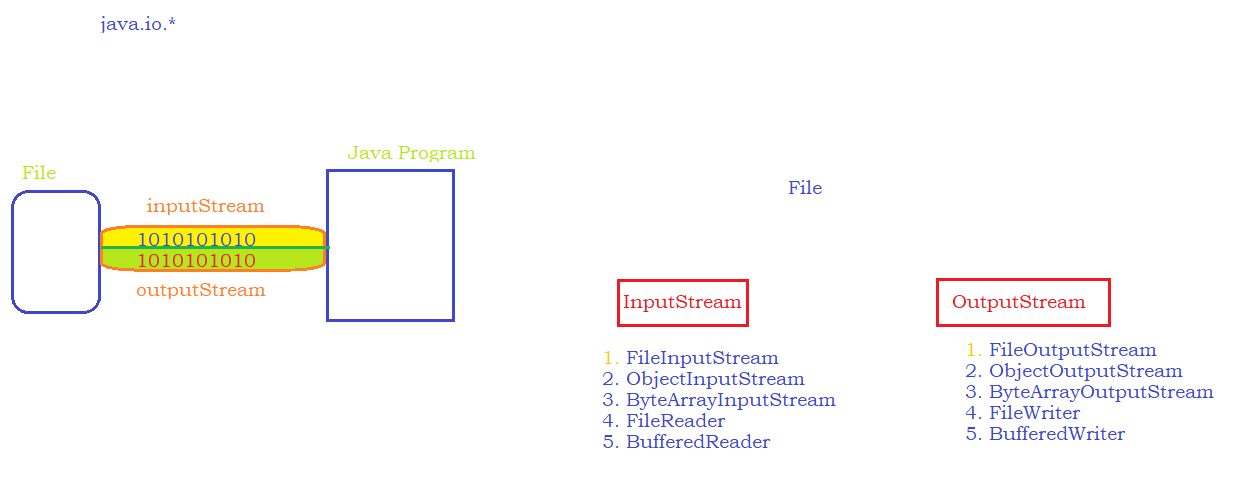
**------**

**} catch(ExceptionClass ex) {**

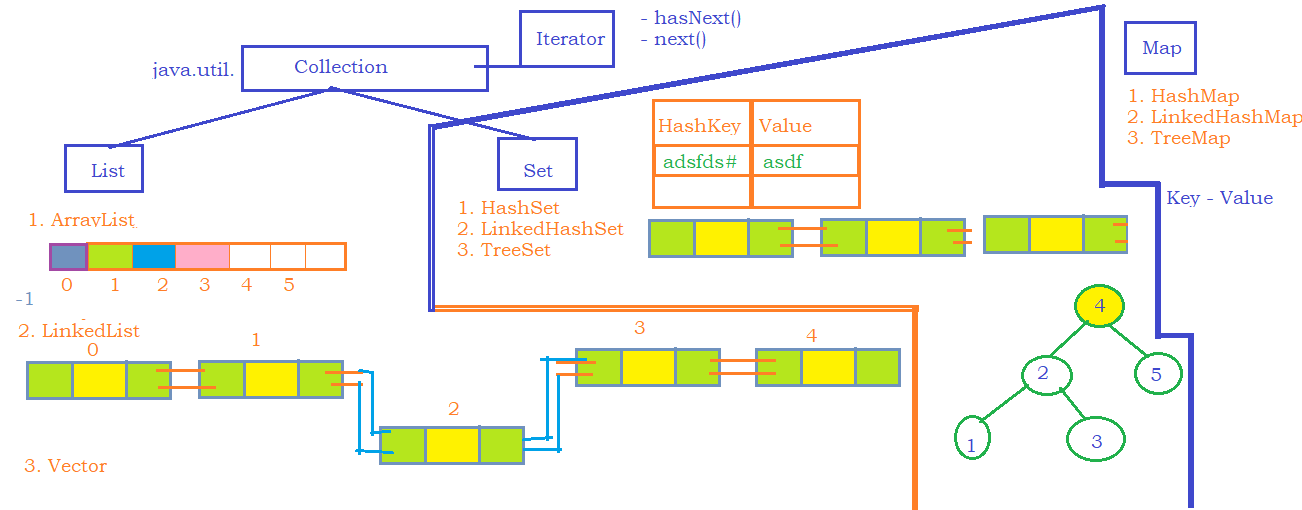
**}**

**ExceptionClass – Every Exception has corresponding class and we should use the class to handle the exception.**

**Files:**

****

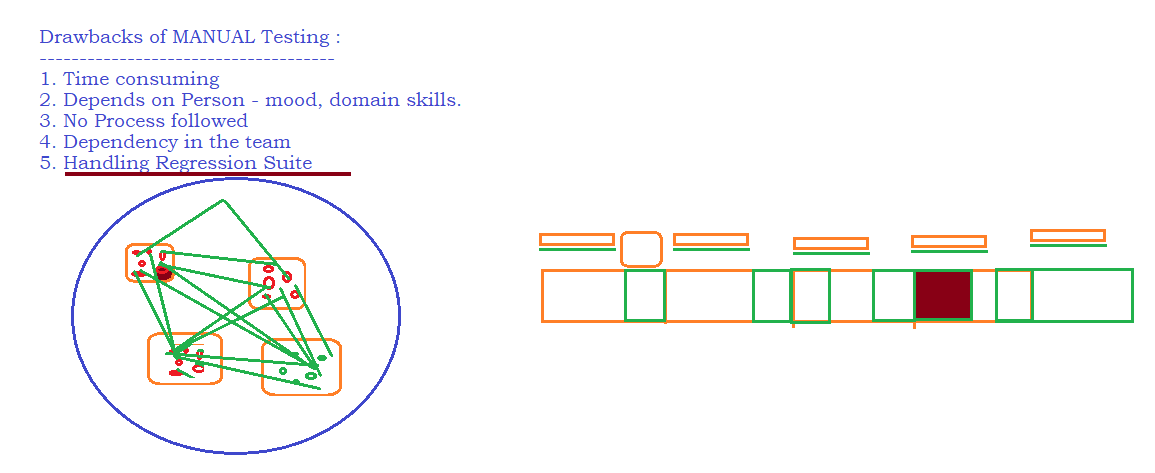
**Collections:**

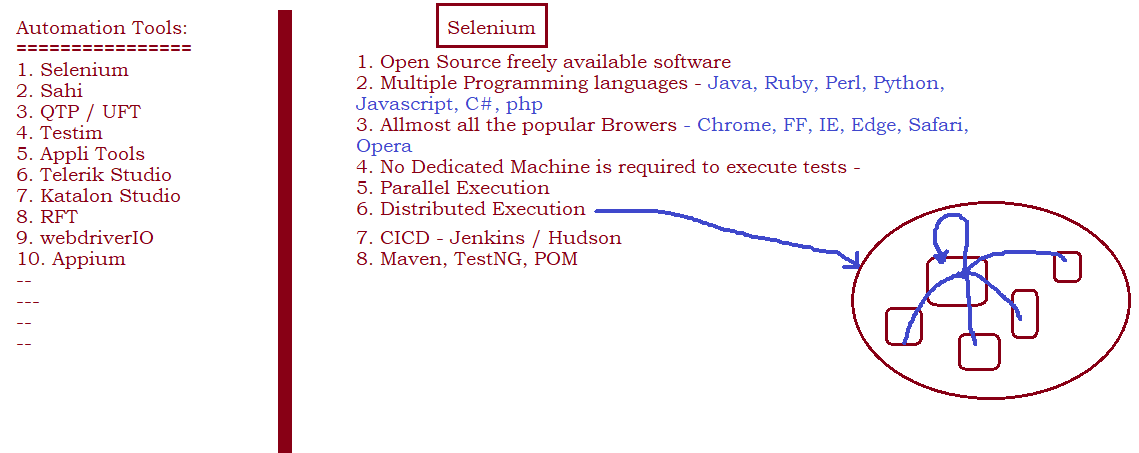
****

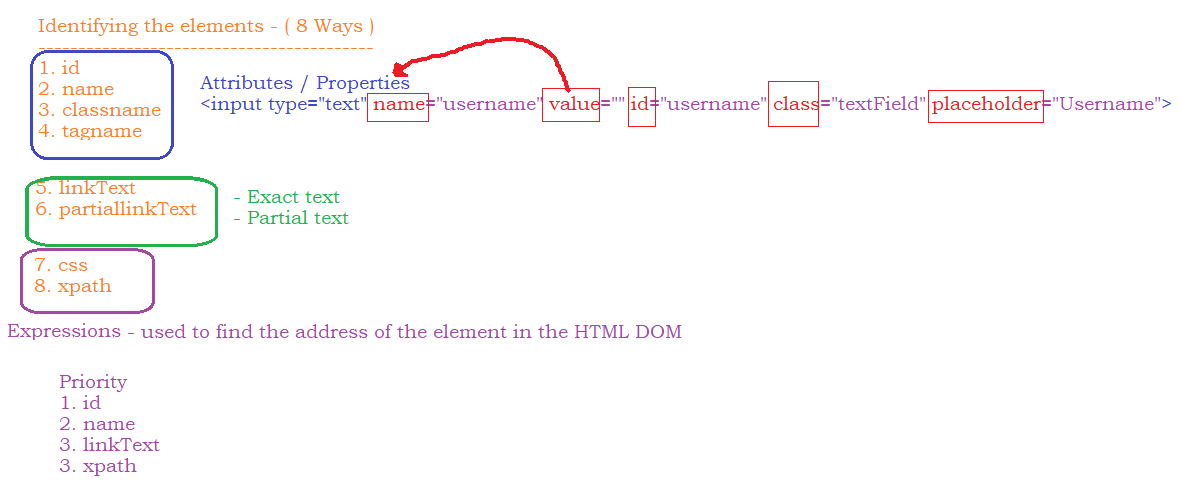
**==================X===============X===================X==============**

**Selenium :**

**Drawbacks of manual testing:**

****

****

****

**CSS – Selector**

1. htmltag[property=’value’] =>other than id or class

input[placeholder='Username']

1. htmltag#valueOfID OR #valueOfID

input#username

1. htmltag.valueOfClass OR . valueOfClass

input.textField

1. Parent to child

htmltag[property=’value’] > childTag

#loginButton > div

**Xpath**

1. //htmltag[@property=’value’]
   1. Username text box - //input[@name='username']
   2. Checkbox on Actitime-Login Page - //input[@id='keepLoggedInCheckBox']
   3. Redbus - //a[@id='redBus Bus Hire']
2. **Xpath using Functions:**
   1. text()
      1. //htmltag[text()=’Exact text on application’]
      2. Ex – RedBus ->
         1. //a[text()='BUS HIRE ']
         2. //button[text()='Search Buses']
   2. contains(arg1, arg2)
      1. arg1🡪 property or function
      2. arg2🡪 corresponding value
      3. Ex – RedBus
         1. //a[contains(text(),'BUS HIRE')]
      4. Ex – Actitime
         1. //img[contains(@src,'timer.png')]
         2. //td[contains(text(),'Please')]
      5. Ex – gsmarena/apple page
         1. //span[contains(text(),'6 Alumi')]
   3. starts-with(arg1,arg2)
      1. arg1🡪 property or function
      2. arg2🡪 corresponding value
      3. Ex – gsmarena/apple page
         1. //span[starts-with(text(),'iPhone')]
         2. //span[starts-with(text(),'iPad Pro')]
3. **Xpath using LOGICAL OPERATORS**
   1. OR
      1. //input[@name='username' or @name='pwd']
   2. AND
      1. Ex- Redbus calendar
         1. //td[(@class='wd day' or @class='we day' or @class='current day') and text()='9']
   3. NOT
      1. //td[text()='13' and not(@class='past day')]
4. **Parent to Child**
   1. //xpath for parent/immediate child
      1. Ex – actitime - //a[@id='loginButton']/div
   2. //xpath for parent// not immediate child
      1. //td[@class='keepLoggedIn']//div[text()='Login ']
5. **Child to parent in absolute way**

Step1 - write xpath to independent element

//**th [text()='Produced by']**

Step2 – put the complete independent element expression inside square bracket //**[th [text()='Produced by']]**

step3 – specifying the parent html tag xpath

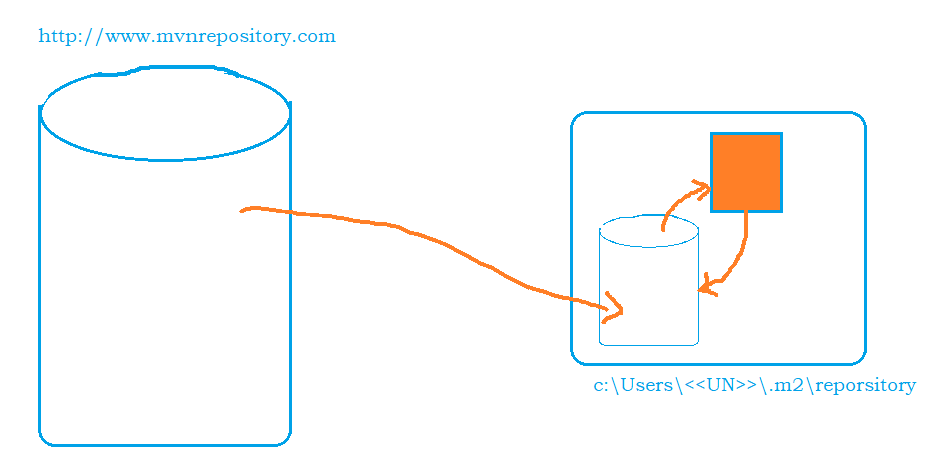
//**tr[th [text()='Produced by']]**

step4 – traverse to required child item

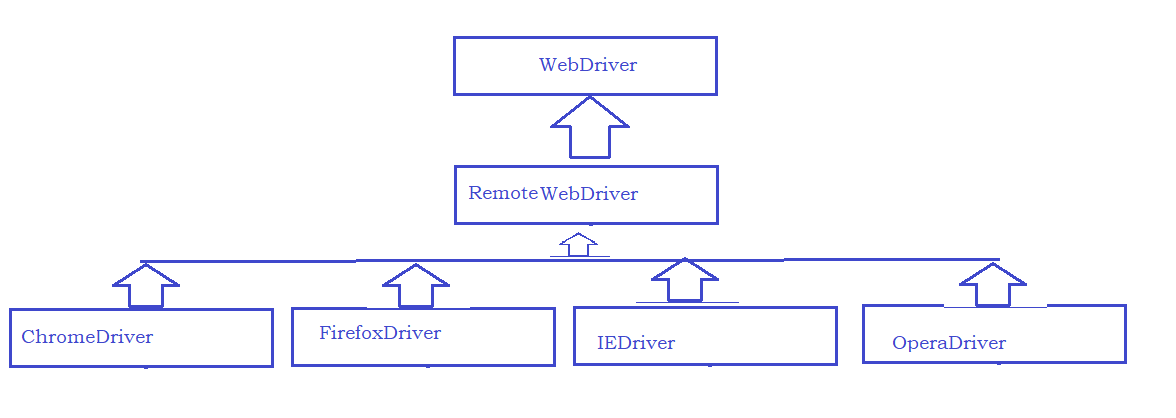
//**tr[th [text()='Produced by']]/td**

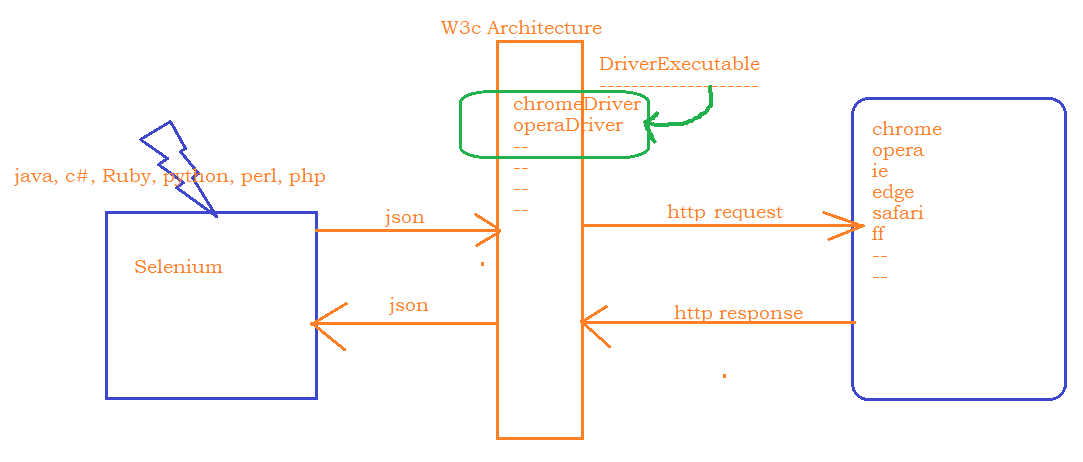
1. **Axes Functions :**
   1. **following-sibling**
      1. **Syntax :**
         1. **//xpath for independent element/following-sibling::siglingTag**
      2. **Example:**
         1. **//th[text()='Produced by']/following-sibling::td**
   2. **preceding-sibling**
      1. **Example:**
         1. **//li[@class='toclevel-1 tocsection-5']/preceding-sibling::li/a[@href='#Cast']**
   3. **following – Find till the end of the page** 
      1. **Example:**
         1. **//li[@class='toclevel-1 tocsection-5']/following::a**
   4. **preceding – Find till the beginning of the page**
      1. **Example:**
         1. **//li[@class='toclevel-1 tocsection-5']/preceding::a**
   5. **parent**
      1. **//th[text()='Produced by']/parent::tr/td**
   6. **child**
      1. **//a[@id='loginButton']/child::div**
   7. **ancestor - Child to parent in Relative way**
      1. **absolute - //tbody[tr[th[text()='Display']]]//td[@class='nfo']**
      2. **ancestor - //th[text()='Display']/ancestor::tbody//td[@class='nfo']**
      3. **axes function - //th[text()='Display']/parent::tr/parent::tbody//td[@class='nfo']**

**Selenium –Setup**

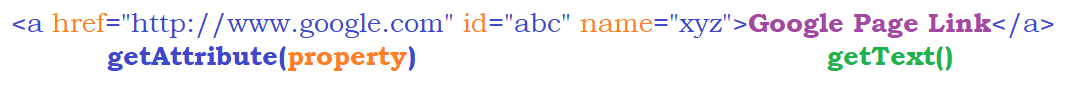
1. **Maven setup –** 
   1. **Set Environment Variables –** 
      1. MAVEN\_HOME - D:\maven\apache-maven-3.6.1
      2. M2 - D:\maven\apache-maven-3.6.1\bin
      3. PATH - D:\maven\apache-maven-3.6.1\bin
2. **Eclipse Project**
   1. **Change compiler settings**
   2. **Change JRE Settings**
      1. RigthClick on project-> properties->Java Build Path-> Select the JRE1.5 and click on remove
      2. Add Library->JRE System Library->Select Alternate JREs Radio -> Installed JREs
      3. Remove what ever you see in Installed LIB window-> ADD -> Standard JVM -> Browse jre present in jdk location (c:\Program files\Java\JDK1.8\jre
      4. ****
   3. ****

**b. pom.xml**

****

****

**Reading Runtime values from Application:**

****