INDIA'S NO1 REALTIME TRAINING INSTITUTE

RAMESHSOFT

SELENIUM WITH JAVA IN REALTIME

100% REALTIME TRAINING

100% PLACEMENTS

TRAINER NAME : RAMESH ANAPATI

CERTIFIED & REAL TIME EXPERT

INDIA'S NO1 REALTIME TRAINING INSTITUTE

RAMESHSOFT

TILL NOW

300+

STUDENTS GOT PLACED

100% REALTIME TRAINING

100% PLACEMENTS

TRAINER NAME : RAMESH ANAPATI

CERTIFIED AND REAL TIME EXPERT

Opposite Of Vindu Tiffin Center Between Canara Bank And Axis Bank 3rd Floor
Near Umesh Chandra Statue SR Nagar Sarala Apartments,HYDERABAD

PH : 9177791456, [redacted]

RAMESHSOFT

JAVA WITH SELENIUM

IN REAL-TIME WITH PRACTICAL APPROACH

BY

Mr. RAMESH ANAPATI

Real Time Expert- M.Tech.

Address: Near Umesh Chandra Statue, Opp.Andhra Bank Lane, above Sai Sudha Opticals, 2nd floor,SR Nagar. Phone:+91 91777 91456 /

Visit: **www.rameshsoft.com/**

E-mail :**rameshsoft.selenium@gmail.com**

Videos:**Youtube.com/Rameshsoft**

RAMESHSOFT

SOFTWARE SOLUTIONS

OUR STUDENTS RECENTLY PLACED ON SELENIUM

AHMED

(SD SOFTECH PVT LTD)

RAFEEQ

(EDREEZ SOFTWARE PVT LTD)

MADHUPRIYA

(CIGNITI TECHNOLOGIES)

SWATHI

(COGNIZANT)

MRUDALA

(SOFTSOL TECHNOLOGIES)

GANESH

(PURPLETALK)

MANDEEP

(SOCTRONICS)

VISHWAJIT

(TCS)

SIBASYS

(COGNIZANT)

SRIKANTH.R

(INFOSYS)

SHRAVAN KUMAR

(IBM)

SANGAMESH

(TECH MAHINDRA)

PRAHALAD

(INFOBRAIN)

HARISH

(PRDC)

KIRAN

(CYBAZE)

THAKIL

(UHG)

RAMESHSOFT

MASTERS INJAVA WITH SELENIUMIN REAL TIME TRAINING

Course Content:

1.Core java

2.Selenium core

3.Selenium advance

4.Selenium Real time frameworks implementation

5.Selenium certification training

6. OCA certification training

7.Selenium real time Web Project

8.Technical discussions on new enhancements & features

9.Technical &Manager,HR Rounds discussions

10.Placement Assistance.

[redacted]

www.rameshsoft.com

Ph. +91 91777 91456 [redacted]

Frameworks

- Frameworks in Real Time we will discuss in detailed in classroom.

- More than 30hrs will discuss.

  Note: This notes is not complete notes

Agile Methodology

* Agile is software development life cycle model (SDLC).
* Agile is a combination of iterative and increment process models.
* Agile methods break the product into small incremental builds. These builds are provided in sprints.
* Agile is a methodology that promotes continuous iteration of development and testing throughout the software development lifecycle of the project and both development and testing activities are concurrent.

Methodologies of Agile Testing :-

1. Scrum
2. Feature Driven Development (FDD)
3. Crystal Methodologies
4. Extreme Programming
5. Dynamic s/w development method (DSDM)
6. Lean s/w development.

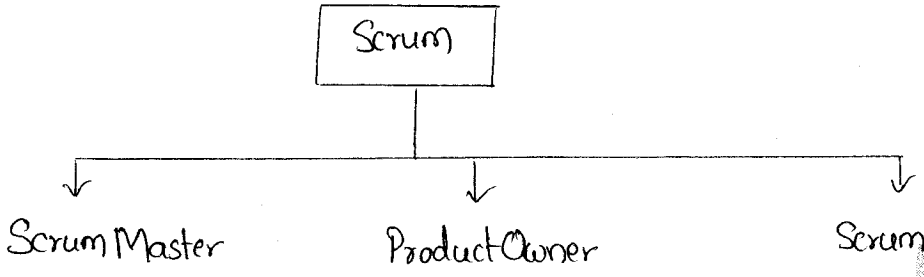
Scrum :-

Scrum is an Agile Development Method which concentrates on how to manage tasks within a team based development environment.

Scrum itself is a Simple Framework for effective team collaboration on complex projects.

Scrum consists of three roles.

1. Scrum Master
2. Product Owner
3. Scrum Team



1. Scrum Master:-

* Scrum Master is responsible for setting up the team, sprint meeting and removes obstacles to progress.
* Scrum Master coach the team, the Product Owner and the business on the Scrum process.
* Scrum Master also look to resolve impediments and distractions for the development team and QA team.

2. Scrum Team (QA + Dev t -):-

* Scrum Team manages its own work and organizes the work to complete the sprint or cycle.
* Scrum Team contains 7+/-2 people. If the project is too big, break team into scrum team.

* Team will have requirements people, designers, coders (developers), testers everyone.
* Team itself should self sustainable, should have all the life cycles possible within that particular iteration or the product.
* Team is a self organized, self managed.
* Team makes a commitment how much they produce within that sprint.

3. Product Owner :—

* The process of Scrum starts with Product Owner.
* Product Owner gets the input from end users, customers, stakeholders and comeup with the product backlog.
* Product backlog are list of features (or) user stories prioritized by the Product Owner in the sequence of busines requirements and own by Product owner.
* Product backlog is a repository where requirements are tracked with details on the number of requirements to be completed for each release.

Sprint :—

Sprint is referred to the fixed period of time the team commits to work in course of developing the product.

* Sprint duration is 2-4 weeks, but once it is fixed for the project it cannot be change.

* The length of the sprint is decided by the team and the Product Owner.
* Before each sprint, the team selects what it will commit to deliver by the end of sprint and the team creates a task level plan for how they will deliver.
* The changes are welcome in the Product backlog but not in the sprint.
* During the sprint, what the team committed to deliver doesn't change and end-date of the sprint doesn't change.
* If something major comes up, the product owner can direct the team to terminate the sprint permanently and starts a new one. (It happens very rare cases).
* During the sprint every day there is a Scrum meeting or daily standup call.

Agile Meetings :-

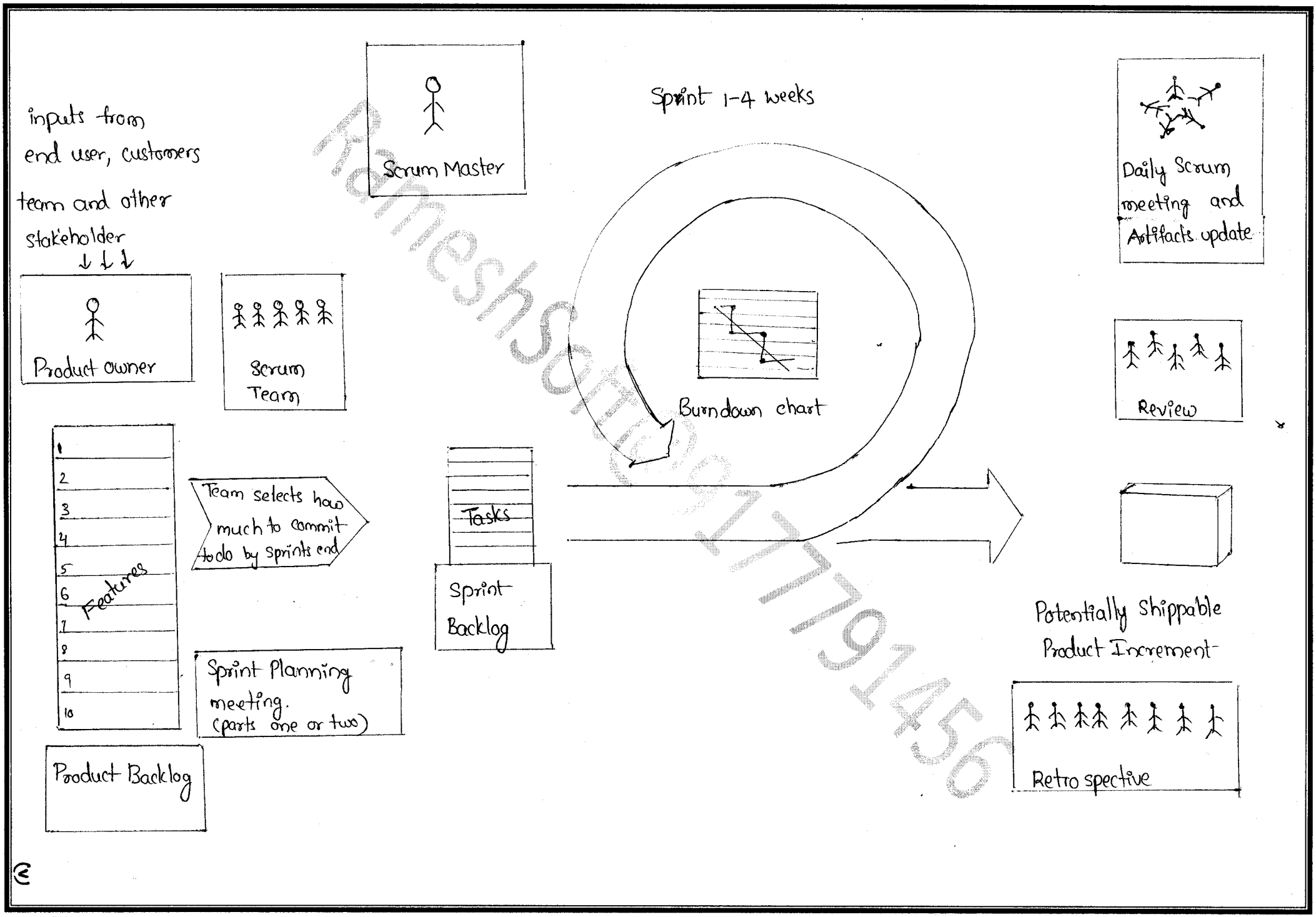
① Product Planning :-

Purpose of the product planning meeting is to discuss long term product vision and duration.

This meeting can be held once every 6 months to continuously evaluate product-level backlogs and priorities.

② Release Planning :-

purpose of the Release Planning meeting is to have features and themes reviewed and prioritized.



RAMESHSOFT @91777 91456 MASTERS IN JAVA WITH SELENIUM REAL TIME TRAINING

The ultimate goal of the meeting is to produce a high level release plan with delivery dates, no of iterations & user stories will be delivered.

③ Sprint Planning:-

Attendees Required:- Development team, Scrum Master, Product Owner.

When:- Usually an hour per week of iteration.

Purpose:- Sprint planning sets up the entire team for success throughout the sprint, coming into the meeting, the Product owner will have a prioritized backlog. They discuss each item with the development team and the group collectively estimates the effort involved.

* The total backlog is become as sprint backlogs.

④ Daily Stand-up meeting:-

purpose of the daily standup meeting is to ensure team members synchronize work on daily basis.

* It is a 15 minutes meeting in each team member is answering the three questions.

a) what we have done since last standup meeting.

b) What is our agenda today & next business day.

c) What are the impediments, if any, I am facing?

Attendees required: Development team, Scrum Master, Product owner.

optional:- Stake holders

Duration: not morethan 15 minutes.

⑤ Sprint Review Meeting:-

In this meeting we will show a demo on working software what we have developed in the sprint.

Attendees required:- Scrum team, Scrum Master, Product Owner.

Optional:- Project stakeholders.

When:- At the end of sprint or mile stone.

Duration:- 30-60 minutes.

* Sprint Review is a 'Product Review'.
* In this meeting it is decided to accept or reject the working software.
* After the meeting feedback is noted from the Product Owner and Product backlog is updated.

⑥ Sprint Retrospective Meeting:-

purpose of this meeting is, In the sprint what went right and what went wrong. And what are the areas that we need to improve.

Attendees Required:- Development team, Scrum Master, Product Owner.

When: At the end of sprint review meeting.

Duration: 60 minutes.

* The team, Product Owner (PO), Scrum Master (SM), meet at the end of each sprint to review their way of working and look for ways to improve their effectiveness.

*It is like a 'Process Review'.

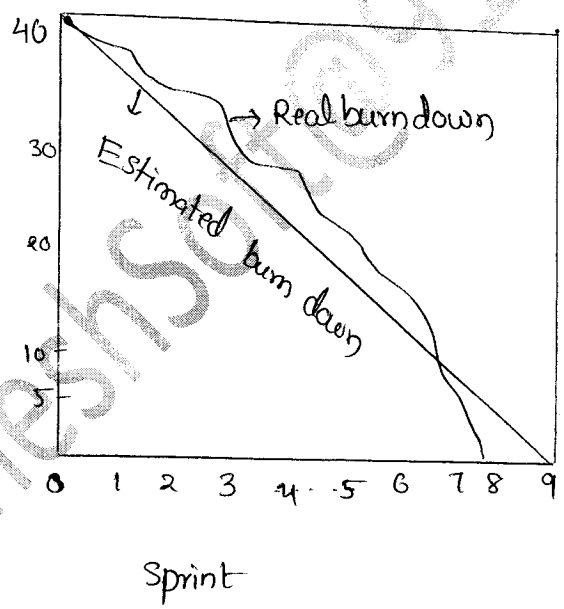
* Process Review means what went wrong, what went right and where should improve.

Burn down Chart (BC):

Chart showing how much work remaining in the sprint calculated in hours remaining.

* Burn down chart is maintained by the Agile Project Manager daily.

* These charts enable the team to successfully self-manage and deliver what they committed by the end of sprint.



Testing

"Testing is a process of checking the application whether it is working properly or not according to client requirements (or) client expectations".

* It is the process of making our application bug-free (or) defect free.
* Testing is a process of checking the quality of the product or project.
* Testing avoids the issues faces by the customer. Once the customer gets the negative feedback when he is using the application, he never comes back to use the application.

Q: What is the difference between Manual & Automation Testing?

ex: Let us consider a scenario, in this scenario we need to execute daily 50 times. Being a manual test Engineer to execute this scenario each & every time, human involvement is compulsory.

Here we are executing the same scenario multiple number of times.

The following problems may encounter

1) need to execute same test case multiple number of time
2) Time
3) More human efforts are required.

→ Human involvement is more in manual testing. i.e, each and every time we need to test the functionality manually.

→ In order to avoid those problems we can go for automation testing.

Automation testing is one-time activity i.e write the code once and

execute number of times.

Q: When do we automate the application?

whenever the application is stable i.e it is not changing then we can automate the application.

Q: When do you automate a Testcase?

whenever the testcase functionality is stable then we can automate.

Q: If the functionality keep on changing, what do you prefer i.e manual testing or Automation Testing?

Manual testing as functionality keep on changing.

Q: Difference between error, defect and bug?

An error is a mistake in coding found by developers.

A defect is a mismatch in between expected value and actual value of a software found by testers. Some times defect is also known as issue or flow.

A bug is a problem in s/w during utilization, found by customer or users.

Java

Features Of Java :-

Features of a language are nothing but the set of services or facilities provided by the language vendors to the industry programmers.

1. Simple :

It is simple because of the following factors :

→ It is free from pointer due to this execution time of application is improve.

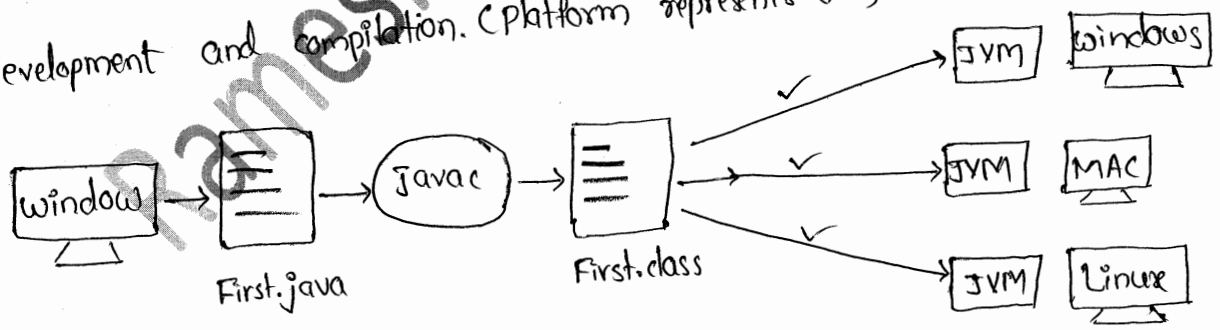
→ It have Rich set of API (application protocol interface)

→ It have Garbage Collector which is always used to collect un-referenced Memory location for improving performance of Java program.

→ It contains user friendly syntax for developing any applications.

2. Platform Independent :

A program or technology is said to be platform independent if and only if which can run on all available operating systems with respect to its development and compilation. (platform represents O.S)



3. Architectural Neutral :

Archetecture presents processor.

A language or technolog is said to be Architectural neutral which can run

on any available processors in the real world without considering there architecture and vendor irrespect to its development and compilation.

4. Portable:

If any language supports platform independent and architectural neutral feature known as portable. The languages like c, cpp, pascal are treated as non-portable language. It is a portable language.

Portability = platform independent + architecture

5. Multithreaded:

A flow of control is known as thread. When any language execute multiple thread at a time that language is known as multithreaded language. and it is multithreaded language.

6. Distributed:

Using this language we can create distributed application. RMI and EJB are used for creating distributed applications. In distributed application multiple client system are depends on multiple server systems so that even problem occured in one server will never be reflected on any client system.

Note: In this architecture same application is distributed in multiple server system.

7. Networked:

It is mainly design for web based applications, J2EE is used for developing network based applications.

8. Robust:

Simply means of Robust is strong. It is robust or strong programming language because of its capability to handle Run-time Error, automatic garbage collection, lack of pointer concept, Exception Handling. All these points makes it robust language.

9. Dynamic:

It support Dynamic memory allocation due to this memory wastage is reduce and improve performance of application. The process of allocating the memory space to the input of the program at a run-time is known as dynamic memory allocation, To programming to allocate memory space by dynamically we use an operator called 'new'. 'new' operator is known as dynamic memory allocation operator.

10. Secure:

It is more secured language compare to other language. In this language all code is covered into byte code after compilation which is not readable by human.

11. High Performance:

It have high performance because of following reasons.

→ This language uses Bytecode which is more faster than ordinary pointer code so performance of this language is high.

→ Garbage Collector, collect the unused memory space and improve the performance of application.

→ It have no pointers so that using this language we can develop an application very easily.

→ It support multithreading, because of this time consuming process can be reduced to execute the program.

12. Interpreted:

It is one of the highly interpreted programming languages.

13. Object Oriented:

It supports OOP's concepts because of this most secure language., for this we need to read oops concepts in the next pages.

How to download Java:

open google → enter 'download Java' → click on 'Java SE downloads/oracle technology network/oracle' link → click on 'Java' download' icon → accept license agreement → If your OS is window x64 then download windows 64-bit JDK or if your os is windows x32 bit then download windows 32 JDK → once click on, you will get '.exe' file → copy that file from downloads and paste in a folder or desktop.

How to install Java:

Double click on downloaded JDK file → click next until close.

*How to check whether java installed successfully or not?

way1: Go to my computer → 'c' drive → program files → check for java folder → jre files.

way2: Open command prompt → type below commands

→ java

=

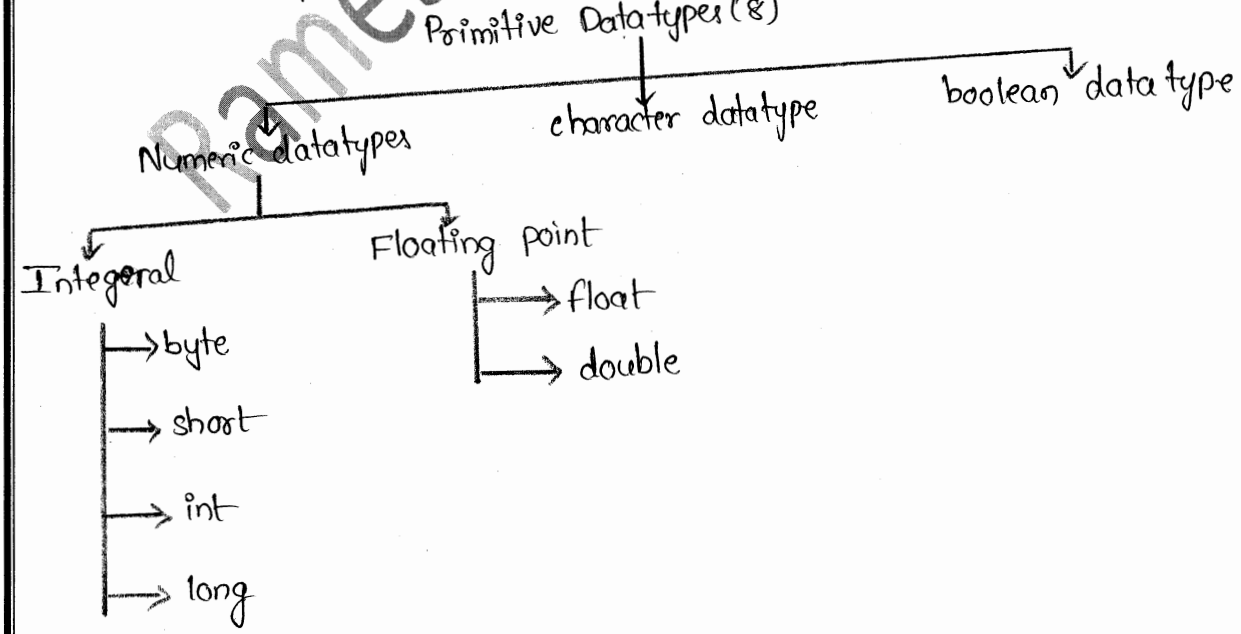
→ java -version

How to set java path :-

Right click on my computer → properties → Advanced system settings → environment variables → click new for user variables → enter variable name as JAVA_HOME → Enter variable value as path of Jdk i.e, c:\program files\ java → click on 'ok' → under system variables select 'path' variable → click "edit" → at the end of existing value put ';' and write path of jdk/bin; → click 'ok'.

Data Types

* Data types are used to represent data in main memory of the computer
* In Java every variable has a type and it is strictly defined.
* There are 8 primitive datatypes are available.



① byte :

size : 8 bits (1 byte)

max-value : +127

min-value : -128

Range : -128 to +127 (-2^7 to 2^7-1)

ex: byte b=10; ✓

byte b=148; ✗ compile time error

byte b=true; ✗ CE

byte b="Rameshsoft"; CE

→ whenever we declare byte variable 1 byte or 8 bits of memory is going to be allocated.

② short :

whenever we declare short variable 2 bytes of memory is going to be allocated

size : 16 bits (2 bytes)

Range : -2^{15} to $2^{15}-1$ ($-32,768$ to $+32,767$)

ex: short s=130; ✓

short s=32769; ✗ CE

short s=true; ✗ CE.

③ int : The most commonly used data type in java is 'int' datatype.

→ whenever we declare int variable, internally 4 bytes (32-bits) of memory is going to allocated.

size : 4 bytes (32 bits)

Range : -2^{31} to $2^{31}-1$ [$-2,147,483,648$ to $2,147,483,647$]

④ long: If we want to hold bigger than int then we can go for long data type.

size: 8 bytes (64-bits)

range: -2^{63} to $2^{63}-1$

Floating point Datatypes:-

If we want to represent real numbers then we should go for floating point data types.

① float: If we want to declare the real value with 5 to 6 decimal places of accuracy is required then we should go for float.

* Whenever we declare float, internally 4 bytes of memory is going to be allocated.

size: 4 bytes (32-bits)

range: -3.4e38 to 3.4e38

② double: If we want to declare the real values with 14-15 decimal places of accuracy is required then we should go for double.

Whenever we declare double, internally 8 bytes of memory is going to be allocated.

size: 8 bytes (64 bits)

range: -1.7e308 to 1.7e308

boolean data type:-

By using boolean data type we can represent boolean values i.e either true or false.

→ size and range is not applicable.

ex: boolean b=true; ✓

boolean b=10; ✗ CE

boolean b = false; ✓

boolean b= 49.0 ✗ CE

char data type:

If we want to represent characters (alphabets) then we should go for char data type.

→ when we declare char data type 2 bytes (16 bits) of memory is going to be allocated.

size: 2 bytes (16 bits)

range: 0 to 65535

Variables in Java

A variable is an identifier whose value is changing during execution of the program.

→ A variable is a value, whenever we declare a variable some memory will be allocated in the Heap area.

→ Based on the requirements, variable values will keep on change.

Types of variables:

1. Non-static variables
2. Static variables
3. Local variables
4. Object type or reference type variables.

Non-static variables:

If the value of a variable is going to be change object to object such type of variables we called as non-static variables.

* When non-static variables will be created?

Instance variables will be created at the time of object creation and destroyed at the time of object destruction.

* When do we declare a variable as non-static?

whenever the value of a variable is going to be change object-to-object then declare that variable as non-static.

* When the non-static variable is going to be destroyed?

Instance or non-static variables are destroyed at the time of object destruction.

Access scope: we can access non-static variables throughout our program

* Where do we declare non-static variables?

After starting the class immediately we will declare non-static variables directly.

* How to define non-static variables:

syntax: datatype variableName = variableValue;

ex: int a=10;

'a' is variable having '10' has value and of type 'integer'.

Default values:

Initialization of variables is optional, if we don't perform initialization of variables JVM is going to assign the default values.

Datatype	Default value
int	0
float	0.0f
double	0.0d
long	0L
string	null
char	nothing
boolean	false

→ Non-static variables are also called as instance variables or object level variables.

→ Non-static variables will be stored in heap area as the part of object

Example:

```
public class TestDemo{
  int x; int y=10;
  public static void main(String[] args){
     TestDemo testDemo = new TestDemo();
     System.out.println(testDemo.x); //0
     System.out.println(testDemo.y); //10;
  } }
```

Static variables:-

If the value of a variable is not varying from object to object then declare that variables as static variables.

→ For static variables at class level a single copy will be created and shared across every object of that class.

→ Whenever we declare variable as static we can access that variables without creating object and we can access by directly class name and by object.

* When static variables will be created? and destroyed?

static variables will be created at the time of class loading and destroyed at the time of class unloading.

* How to define the static variables:

Syntax: static datatype variableName = variableValue;

Access Scope: we can access static variables throughout our program.

* Where do we declare static variables?

After starting the class immediately we will declare static variables directly with static keyword.

Default values:

Initialization of static variables is optional. If we don't perform initialization of variables JVM is going to assign the default values.

→ static variables are also called as class level variables.

→ static variables will be stored inside method area.

Example:

```
public class TestDemo{
 static String x;
 static String y="RameshSoft";
 public void main(String[] args){
  System.out.println(x); //null
  System.out.println(y); // RameshSoft
 }}
```

Local variables:

whenever we declare variables inside the method or constructors such type of variables we can call it as local variables.

* When local variable will be created! destroyed!

local variables will be created at the time method or constructor creation and method or constructor execution is done local variables will be destroyed.

* When do we declare a variable as local?

Whenever you need the values for certain portion of area to meet requi-
-rements then declare that variables as local variables.

Access Scope: we can access local variables within that method or constructor but not outside of that.

* Where do we declare local variables?

we need to declare local variables within the method or constructors and blocks

Default values:

For local variables compulsory we must and should perform initialization otherwise we will get compile time error, i.e for local variables jvm wont provide any default values.

Example :

```
public class TestDemo{
 public static void main(String[] args){
   static string x;
  System.out.println(x); // CE }
   }
```

→ local variables will be stored in stack memory.

→ local variables are also called as temperary variables or automatic varia-
-bles or stack variables.

Object type or Reference type variables:—

A variable that refers a class name is known as reference variable.

A reference variable cannot work like an object until it is converted into an object.

ex: Demo demo; —(i)

demo = new Demo(); —(ii)

Here Demo is class Name

In (i) 'demo' is called a reference variable (not object)

(ii) reference variable is converted into an object

Example:

```
public class Demo{
  int x=10;
  public static void main(String[] args){
   Demo demo;
   //System.out.println(demo.x); → NullPointerException
   demo = new Demo();
   System.out.println(demo.x); // 10
    }
 }
```

→ Default value of demo is 'null'.

Access Modifiers :

The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.

There are four types of Java access modifiers.

1. private 2. default 3. protected 4. public

private :-

private is a modifier applicable for variables, methods and nested classes. when we declare a variable or method as private, we can access that variable within that class only and when you try to access outside of class we will get compile time error.

Example :

```
public class Test {
  private int a=10;
  private void greetings(){
    System.out.println("Welcome to Rameshsoft @ 9177791456");}
  }
public class PrivateDemo {
  public static void main(String[] args){
   Test test=new Test();
   System.out.println(test.a); // CE
   System.out.println("----");
   test.greetings(); // CE
  }
}
```

Note: A class cannot be private or protected except nested class.

default:

If you don't use any modifier, it is treated as 'default' by default. The default modifier is accessible only with in package.

In this example, we have created two packages. 'modifiers' and 'modifiers1'. we are accessing the 'Test' class from outside its package since 'Test' class not public, so it cannot be accessed from outside the package.

Example1: package com.modifiers;

```
class Test{
 private int a=10;
 private void greetings(){
 System.out.println("welcome to Rameshsoft @91777 91456");}
 }
 public class PrivateDemo{
   public static void main(String[] args){
     Test test=new Test();
      System.out.println(test.a);// CE
      test.greetings();//CE
   } }
```

Example2: package com.modifiers1;

```
import com.modifiers.*;
public class TestDemo{
  public static void main(String[] args){
    Test test=new Test();//CE because it is default class, so we
                          cannot access inoutside package

    PrivateDemo private = new PrivateDemo();
  } }
```

protected:

The 'protected' access modifier is accessible within package and outside the package, but through inheritance only (ie in child class only).

Example:
```
package com.modifiers;
public class ProtectedDemo {
    protected int a=10;
    protected void hello(){
     System.out.println("welcome to RameshSoft");
    }
```

```
package com.modifiers1;
import com.modifiers.*;
public class ProtedDemo1 extends ProtectedDemo
{
  public static void main(String[] args) {
     ProtedDemo1 pdemo = new ProtedDemo1();
      pdemo.hello();
  }
}
```

public:

when we declare a variable or method or class as public, we can access those from within the package and outside the package. simply, we can tell every where we can access.

Example:
```
package com.modifiers;
public class PublicDemo {
    public int x=10;
```

```
    public void greetings(){
        System.out.println("Welcome to RameshSoft @91777 91456");
    }
}
```

```
package com.modifiers1;
import com.modifiers.*;
public class Demo{
 public static void main(String[] args){
    PublicDemo publicDemo = new PublicDemo();
    System.out.println(publicDemo.a);
    publicDemo.greetings();
    }
}
```

Observe the following table:

Modifier	Same class	Same package	Subclass (other package)	Universe/outside package
private	Yes			
default	Yes	Yes		
protected	Yes	Yes	Yes	
public	Yes	Yes	Yes	Yes

Methods:

Collection of business statement, that are grouped together to perform an action.

There are two types of methods.

1. Non-static methods
2. static methods.

Non-static method:

* Whenever you want to access the method through object, then declare that method as non-static method.
* method basically contains the business logic.
* method is a function or an action, which perform a specific task.

Syntax: Access specifier (1) void/return type (2)(3) methodname() (4)

{ (5)

statements; (6)

} (7)

(1) Access specifier: It defines the access type of the method and it is optional to use.

(2)(3) return type/void: method may return a value.

If we declare a method as void then that method should not return anything.

(4) methodname: method name can be anything, but it should be very meaningful.

* Every method name should starts with small letter.

* If the method name is having multiple words, then first word should start with lowercase letter and the next words should starts with uppercase letters.

* special characters are not allowed.

* spaces are not allowed in between the method name.

ex: primeNumberDemo()

⑤ Indicates beginning of the method

⑥ Collection of business statements

⑦ closing a method

```
ex:  public void helloDemo()
     {
      System.out.println("Hello Welcome to RameshSoft");
     }
```

Static methods:

* If we declare a method as static, then we can access that method without creating object.

* If the logic of the method is not going to be changed then declare that method as static method.

```
ex: public static void hello()
    {
    ==
    }
```

How to access static variables and static methods:

we can access static variables and static methods in 3 ways.

1. directly by calling their names
2. by using class name
3. by creating object

1. Directly by calling their names:

```
ex: public class TestDemo {
      int a=10;
      static int b=20;
      public void hello() {
       System.out.println("Hello!");
      }
      public static void display()
      {
       System.out.println("welcome to Rameshsoft");
      }
      public static void main(String[] args) {
        system.out.println("The value of b is:"+b);
        display();
      }
    }
```

output: The value of b is : 20
Welcome to Rameshsoft

2. By using class name:

```
ex: public class TestDemo {
      int a=10;
      static int b=20;
```

```
    public void hello()
    {
    System.out.println(" Hello");
    }
    public static void display(){
    System.out.println("Display");
    }
    public static void main(String[] args){
     int c = TestDemo.b;
     System.out.println(c);
     TestDemo.display();
     }
   }
```

output: 20
Display

3. By creating object:

creating object:

Syntax: classname/interfacename objectvariablename = new classname();

ex:

```
public class TestDemo{
 int a=10;
 static int b=20;
 public void hello()
 { System.out.println("Display"); }
 public static void main(String[] args) {
  TestDemo testDemo =new TestDemo();
    int c = testDemo.a;
    System.out.println(c);
    testDemo.hello();
    testDemo.display(); } }
```

output : 10
hello
Display

Note : Always recommended approach is by using class name.

How to access non-static variables and non-static methods :

we can access non-static variables and methods by using only creating an object.

ex: TestDemo testDemo = new TestDemo();

In the above program, display() method is a static method and hello() method is a non-static method.

static vs non-static :-

* Inside non-static area i.e non-static method, we can access both static variables and methods and non-static variables and methods.

* Inside static area i.e static method, we can access only static variables and methods.

* we cannot access non-static variables or methods inside static method. If we want to access, it will show compile time error as 'cannot make reference non-static fields into static area'.

Ex:

```
public class TestDemo{
  int a = 10;
 static int b=20;
  public void hello(){
```

```
System.out.println("a value is : "+a);
System.out.println("b value is :"+b);
}
public static void display()
{
 hello(); // Error area
 System.out.println("a value is:"+a); // error area
 system.out.println("b value is :"+b);
}
public static void main(String[] args){
 System.out.println(TestDemo.b);
 TestDemo.display();
 TestDemo testDemo=new TestDemo();
  testDemo.a;
  testDemo.hello();
 }
}
```

output: Compile time error

In the program, we accessed non-static method hello() non-static variable 'a' inside display(), which is a static method that is why it is showing. error.

OOPS Concepts:

1. Class
2. Object
3. Data Hiding
4. Data Abstract
5. Polymorphism
   i) overloading
   ii) overriding
6. Inheritance (is-A relationship)
7. Encapsulation

class:- A class is a collection of business implementation of business functionality in the form of business method.

A class is a collection of data members and methods.

A class is a collection of objects. It is a logical entity.

Object:-

An object is an instance of class.

Object is a blue print of a class.

Data Hiding:-

Data hiding is a software development technique specifically used in object-oriented programming to hide internal object details (data members).

* Simply data hiding means restricting the access and hiding data.
* we can achieve data hiding by using 'private' modifier.

ex: public class TestDemo {

    String username;
    private String password;
    = }

Data Abstraction:

Data Abstraction means hiding internal implementation & just highlight the set of service is called Data abstraction.

Encapsulation:

Binding or wrapping code and data together into a single unit is known as encapsulation.

* A java class is the example of encapsulation.

Tightly encapsulated class:

A class is said to be tightly encapsulated class if and only if it contains all private variables and it doesn't care about methods whether they are public or private.

```
//example for tightly
//encapsulated class
public class Test
 {
  private int a;
  private int b;
  private string s;
  public void m1()
    {
     =
    }
 }
```

```
//example for Encapsulation class
//(not tightly encapsulated class)
public class Test
 {
 int a=10;
 private int b=20;
  =
  =
 }
```

Inheritance :

By using inheritance, we can extend the functionalities of existing class into a newly derived class.

* The main advantage of inheritance is 'code reusability'.
* we can implement inheritance by using 'extends' keyword.
* minimum two classes are required to perform inheritance.

Ex:

```
class DemoOne
{
int a=10;
public void hello(){
  System.out.println("Hello method");
}
public void display()
{ System.out.println("Display method"); }
}
class DemoTwo extends DemoOne
{
  public void childMethod()
  { System.out.println("child method"); }
}
public class InheritDemo
{
  public static void main(String[] args){

    DemoTwo demoTwo = new DemoTwo();
    System.out.println(demoTwo.a);
    demoTwo.hello();
    demoTwo.display();
    demoTwo.childMethod(); } }
```

INDIA'S NO1 REALTIME TRAINING INSTITUTE

RAMESHSOFT

100% REALTIME TRAINING

100% PLACEMENTS

TILL NOW

300+

STUDENTS GOT PLACED

TRAINER NAME : RAMESH ANAPATI

CERTIFIED AND REAL TIME EXPERT

Opposite Of Vindu Tiffin Center Between Canara Bank And Axis Bank 3rd Floor
Near Umesh Chandra Statue SR Nagar Sarala Apartments,HYDERABAD

PH : 9177791456, 9502695908, 040-48572456

Frameworks

- Frameworks in Real Time we will discuss in detailed in classroom.

- More than 30hrs will discuss.

  Note: This notes is not complete notes

Polymorphism :

when one task is performed in different ways i.e knows as polymorphism.

In java, we use method overloading and method overriding to achieve Polymorphism.

overloading :-

If a class has multiple methods by same name but different parameters it is known as method overloading.

* By using overbading increasing the readability of the program.

Ex:
```
class Demo {
    int a, b;
    public void add (int a, int b)
    {
     int c=a+b;
     system.out.println(c);
    }
    public void add (float a, float b)
    {
     float c=a+b;
     System.out.println(c);
    }
    public void add(double a, double b)
    {
    double c= a+b;
    system.out.println (c);
    }
    public void add( )
    {
    system.out.println ("no arguments add() method");
    }
}
```

```
public class OverloadingDemo {
  public static void main(String[] args) {
    Demo d=new Demo();
     d.add();
     d.add(10,20);
     d.add(10.50f, 25.60f);
     d.add(10.0, 50.0);
    }
}
```

Automatic promotion in overloading:

while performing overloading method resolution if there is no method with specified argument type compiler won't raise any error immediately.

→ First compiler promotes that argument to the next level and checked for matched method.

If the match is available then it will be considered. otherwise, once again promotes that argument to the next level until all possible promotion, still matched method is not available then only compiler raises error.

This promotion of argument type is called automatic promotion in overload-ing.

The following are various possible automatic promotion in java.

byte → short → int → long → float → double

char ↗

**Note:— In overloading method resolution always taken care by the compiler based on reference type. It is also considered as static polymorphism (or) compile time polymorphism.

Overriding :-

Through inheritance, whatever the parent class has by default available to every child class. whenever child class is not satisfied with parent class implementations, then child class has the ability to override that method with their own content or with their own implementations. This is called method overriding.

* In overriding, method resolution always taken care by Runtime object. This is also called Runtime polymorphism or Dynamic polymorphism.

Rules to follow while overriding :-

① In overriding method names and arguments must be matched i.e, method signatures should be same.

② while overriding the return types must be same. This rule is applicable until 1.4 version. But from 1.5 version onwards co-varient return types are allowed. According to this child method returntype need not be same as parent method return type its child is also allowed.

ex: ① Object ✓

↓

Object | String | StringBuffer | Integer ...

ex: ② ✓ Number ↓ Integer | × ③ String ↓ Object | × ④ double ↓ int

Note: Co-varient return type concept is not applicable for primitive types.

③ private methods are not visible in child classes, Hence overriding concept is not applicable for private methods. But based on our requirement, we can define exactly same private method in child class, it is valid but it is not overriding.

④ Parent class final methods cannot be overridden in child classes. But a non-final methods can be overriden as final.

⑤ we should override Parent class abstract methods in child class to provide implementation.

⑥ A non-abstract method can be overridden as abstract to stop availability of Parent class method implementation to the child classes.

⑦ A non-final method can be overridden as final method. But a final method cannot overridden as non-final method.

method Hiding:—

→we can't override a static method as non-static.

If both parent and child class methods are static then we won't get any compile time error. It seems to be overriding is possible but it is not overriding, it is method hiding.

```
Ex:   public class Demo{
          public static void m1()
              {System.out.println("Parent static method");
              }
      }
      class childDemo extends Demo
          public static void m1()
              {
```

```
        System.out.println("Child static method");
        }
    }
```

Here m1() static method is overridden in child class, and it is not overriding. but it is method hiding.

Example for overriding:

```
class Vehicle{
  public void run()
    {
    System.out.println("vehicle is running");
    }
}
class Bike extends Vehicle
{
    public void run()
      {
      System.out.println("Bike is running safely");
      }
      public static void main(String[] args)
       {
        Bike bike = new Bike();
          bike.run();
       }
}
```

Output: Bike is running safely.

Can we overload main() ? Explain with an Example:

Yes, we can oveload main() method. but Jvm will always call the original main method, it will never call our overloaded main methods

Example:

```
public class MainMethodOverloadDemo{
    public static void main (String[] args){
        System.out.println (" standard main method");
        }
    public static void main(int[] args)
    {
    System.out.println(" int[] array main()");
    }
    public static void main()
    {
    System.out.println("no parameterized main()");
    }
}
```

After execution the output is: "standard main method."

i.e Jvm will call only original main method, not overloaded methods

So, if we want to execute overloaded methods, we need to call those overloaded methods from original main method.

```
i.e  public static void main (string[] args) {
            main(new int[]{1, 2, 3}); //calling overloaded int[] arg main()
            main(); // calling overloaded main() method
     }
```

Q: can we inherit main() ? Explain with example

Yes we can inherit the main().

Example: // Child.java

```
public class Child extends Parent
  {

  }

class Parent {

  public static void main(String[] args)
    {
    System.out.println("Main method from Parent class");

    }
```

Output: Main method from Parent class

***

Note 1:- A non-abstract method can be overridden as abstract to stop availability of Parent class method implementation to the child classes.

Note 2:- illegal combinations with abstract

private & abstract — illegal combination
abstract & final — illegal combination
abstract & static — illegal combination

Abstract

* abstract is a keyword which is applicable for method and classes but not for variables.

* If you want to restrict the object creation of class, simply declare that class as abstract.

*If we declare a class as abstract, then we cannot create object for that class. and if we try to create object for abstract class, we will get compile time error as 'abstract' type cannot be instantiated'.

```
ex: abstract public class Test {
      int a, b;
      public static void main(String[] args)
       {
        Test test = new Test(); // CE
       }}
```

* Eventhough a class is abstract, we can access the data of that class using inheritance concept.

```
ex: abstract class Test1
      {
       int a=19, b=30;
       public void hello()
        {
        System.out.println(a);
        System.out.println(b);
        }
      }
      public class Test extends Test1
        {
```

```
    public static void main(String[] args) {
        Test test = new Test();
        System.out.println(test.a);
        System.out.println(test.b);
        System.out.println("welcome to Rameshsoft");
        test.hello();
    }
}
```

output: 19
30
welcome to RameshSoft

19
30

Abstract method:—

If we don't know the implementation and just we have a service requirement, then declare that method as abstract method.

* Every abstract method should ends with ';'
* Abstract method doesn't contain any body.

How to declare abstract method:

abstract accessspecifier returntype methodname;

ex: abstract public void m1();

* If a class has atleast one abstract method, then declare that class as abstract.
* whenever the class has abstract method, then child classes are responsible to provide the implementation for abstract methods.

RAMESHSOFT

SOFTWARE SOLUTIONS

OUR STUDENTS RECENTLY PLACED ON SELENIUM

AHMED

(SD SOFTECH PVT LTD)

RAFEEQ

(EDREEZ SOFTWARE PVT LTD)

MADHUPRIYA

(CIGNITI TECHNOLOGIES)

SWATHI

(COGNIZANT)

MRUDALA

(SOFTSOL TECHNOLOGIES)

GANESH

(PURPLETALK)

MANDEEP

(SOCTRONICS)

VISHWAJIT

(TCS)

SIBASYS

(COGNIZANT)

SRIKANTH.R

(INFOSYS)

SHRAVAN KUMAR

(IBM)

SANGAMESH

(TECH MAHINDRA)

PRAHALAD

(INFOBRAIN)

HARISH

(PRDC)

KIRAN

(CYBAZE)

THAKIL

(UHG)

* If the child class also doesn't provide any implementation then declare that class as abstract and again another child is responsible to provide implementation.

* while implementing abstract method in child classes, remove abstract keyword and provide the implementation.

Ex:

```
abstract class TestTwo{
  abstract public void m1();
  abstract public void m2();
    public void m3()
    { system.out.println("m3");
    public void m4()
    { system.out.println("m4"); }
}
class TestThree extends TestTwo{
   public void m1()
    {
     System.out.println("m1"); }
    public void m2(){
     system.out.println("m2"); }
    }
   public class AbstractDemo{
      public static void main(String[] args){
          TestThree test = new TestThree();
            test.m1();
```

```
        test.m2();
        test.m3();
        test.mu();
    }
}
```

Abstract class vs Concrete class:-

Abstract class contains both implemented methods and non-implemented methods (abstract methods), so we cannot create object for abstract class.

* Abstract class is also called as "partially implemented class".

Concrete class contains only fully implemented methods, so we can create object for concrete class.

note: we can create constructor and execute the constructor for abstract classes by using inheritance and super() constructor call.

Interface

Interface is a collection of pure abstract methods i.e, it contains only abstract methods.

* From client point of view, an interface defines the set of services what he is offering and from service provider point of view, an interface defines the set of services what he is expecting.

* An interface contains services or requirements in the form of abstract methods.

Syntax: interface interfacename
① ②
{
= abstract methods
}

① → it is a keyword inorder to convey that it is an interface.

② → interface name can be anything, but it should be very meaningful and every interface name should start with uppercase alphabet only.

* How to implement interface:

As interface contains all abstract methods, we should implement that interface

* child classes are responsible to provide the implementation of interface.
* whenever we implement the interface by using "implements" keyword.
* whenever we implement abstract methods in the child class, we should implement that method by removing 'abstract'

```
Ex: interface ATM
    {
    void cashWithDraw();
    void miniStatement();
    void fastCash();
    void balanceEnquiry();
    void pinChange();
    }
    abstract class ATMImplement implements ATM
    {
      public void cashWithDraw()
          { System.out.println("cashWithDraw"); }
```

```
public void miniStatement()
 {
System.out.println("mini statement");
 }
public void pinChange()
 {
System.out.println("pin change");
}
public void fastCash()
 {
System.out.println("fast cash");
 }
}
class TestImpl extends ATMImplement
{
  public void balanceEnquiry()
   {
   System.out.println("Balance enquiry");
   }
}
public class InterfaceDemo{
  public static void main(String[] args)
   {
    TestImpl  test = new TestImpl();
     test.balanceEnquiry();
     test.fastCash();
     test.miniStatement();
     test.pinChange();
     test.fastCash();
```

(or)

we can create object by using interface reference, so that we can access only interface specific methods.

```
ATM atm = new TestImpl();
    atm.balanceEnquiry();
    atm.fastCash();
    atm.miniStatement();
    atm.pinChange();
  }
}
```

Note: The Java compiler adds public and abstract keywords before the interface method and public, static and final keywords before data members (variables).

ex:

```
interface Printable{
    int MIN = 5;
    void print();
}
```

→ (compiler) →

```
interface Printable {
public static final int MIN = 5;
public abstract void print();
}
```

interface Vs Abstract class Vs Concrete class:-

Interface:-

→ An interface must contain only method signatures and static data members.

→ An interface contains only abstract methods i.e, it doesn't have definitions of the methods declared in it.

→ As it contains abstract method, we can not create object for interface

→ we cannot create constructor for interface.

→ Interface never talks about implementations, but an interface 'extends' another interface.

Abstract class:-

→ Abstract class talks about implementation, but not completely i.e, abstra-ct class contain both implemented methods and non-implemented method (abstract method).

→ we can create and we can execute the constructor for Abstract class through child class.

→ we cannot create object for abstract classes.

Concrete class:-

→ Concrete class is a fully implemented class we can create object for concrete d class.

→ we can create and execute constructor for concrete class.

→ Concrete class talks about only implementations.

**
Note:- In interface from java 8 we can write the following methods.
In Java 8, we can able to define 'default methods' and 'static methods'. this feature is not available before java 8.

But before java 8, in interface we can only write abstract methods.

Structure of Java Program:

If we want to develop any application (or) write any program in any programming language, we follow a standard format which released by language developers.

As a part of java programming James Grosling has released the following standard format for writing java applications.

```
comment lines ; —(i)
package details; —(ii)
class <classname>
(iii)   (iv)
{
  Data members —(v)
  user-defined methods — (vi)
  public static void main(String[] args)
   (x)    (ix)   (viii)   (vii)      (xi)
  {
    block of statements; —> (xii)
  }//end of main()
}// end of class.
```

Explanation:—

In above structure

i) comment lines: By using comment line we can increase the readability

of the program.

* whatever we write between the comment lines, the code won't compiled. i.e compiler simply ignores the code which we write between the comment lines.

→ There are two types of comments.

① single line comments

② multi line comments.

Single line comments starts with // (double slash)

multi lines comments starts with /* (some text lines) */.

ii) Package is a collection of predefined classes, interfaces & sub packages. A sub package interms contains collection of classes, interfaces and sub-sub packages, etc. In our java program if we use any predefined classes and interfaces then it is the responsibility of the java programmer to specify in which package the classes, interfaces present otherwise we get compile time error.

* In java programming out of so many number of packages, one of the package is imported by default known as "java.lang.*;"

iii) class: class is a keyword used for developing user/programmer defined datatype.

iv) <classname>: represent a java valid variable name treated as the name of the class. Programatically each & every class name can be treated as user defined data type.

v) Data members represents either instance or static, and they will be selected based on the name of the class.

vi) User-defined methods represents set of methods which are either non-static or static and they meant for performing the operations either once or each & every time.

Each & every user defined method contains business logic which will solve the requirement of client problem.

vii) Each & every java program starts executing from main(). Hence main() is also known as program driver.

viii) Since main() of java not returning anything and hence whose return type is void.

ix) Since main() of java executes only once throughout the life of the java program and hence it's nature must be static.

x) Since main() of java accessed by all the java programmers and hence whose access specifier must be public.

Access specifier always makes us to understand where to access the data & where not to access the data.

xi) Block of statements represents set of executable statements which are internally calling the data members & methods of the class.

xii) The file naming convention in java programming is that whichever class name is containing main(). Such class name must be given with an extension '.java'.

Java Source file structure:

① A Java Program can contain any number of classes and atmost one class should be declared as public.

② If we declare any class as public then name of the program and name of the public class must be matched otherwise we will get compile time error.

③ If there is no public class then we can use any name for java program.

ex: RameshSoft.java
Ramesh.java
A.java
B.java
C.java

```
class A
{ }
class B
{ }
class C
{ }
```

④ Only one class should be declared as public when we have multiple java classes. when we are trying to declare two classes as public we will get compile time error.

```
public class A
{
==
}

public class B
{
==
}
```

compile time error.

⑤ The class which we declared as public, the public class name and java source filename should be same otherwise we will get compile time error.

⑥ The class which we declared as public, in that public class only always recommended to write main( ) method.

Q: what happens with the following line?

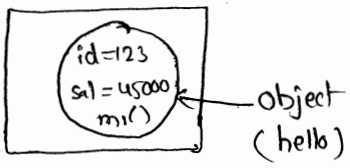
```
BClassDemo bclassDemo = new BClassDemo();
```

* whenever we use "new" keyword or 'new' operator in java, object is going to be created.
* The moment we create an object, all the non-static variables and methods are going to be participated in the object.
* whenever we create object, internally some memory will be allocated.

Note:
* whenever we create object, only non-static variables and methods are going to be participated in object creation but not static variables and methods.

ex:
```
class Hello
{
static string name = "Rameshsoft";
 int id = 123;
 int sal = 45000;
 public void m1()
 {
 }
public static void main(String[] args)
 {
 Hello hello = new Hello();
 System.out.println(hello.id); } }
```

heap area



* Jvm always looks for main() to execute the program. i.e whenever we run the java program jvm always looks for main().
* If main() is there then it executes the program and gives the corresponding output.
* If main() is not there it will not execute the program simply it throws Exception

Packages:-

package is a collection of classes and interfaces.

* By using packages, we can avoid naming collopsions.
* we can get the visibility if we maintain packages.

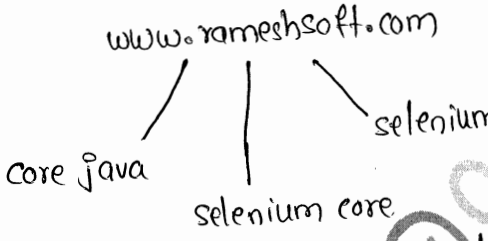
syntax: package packagename;

↓ keyword ↓ name of the package, but it should be meaningful.

How to define a package name:-

domainnameinreverse.modulename.submodulename;

Ex:



① package core java → not recommended way

② package com.rameshSoft.corejava → Hightech city level. (it is recommended)

* we need to write classes in packages and we need to write packages at a file source level.
* we can write only one package in one class and a class may have any number of import statements.

ex: package pack1; → invalid

package pack2; → invalid

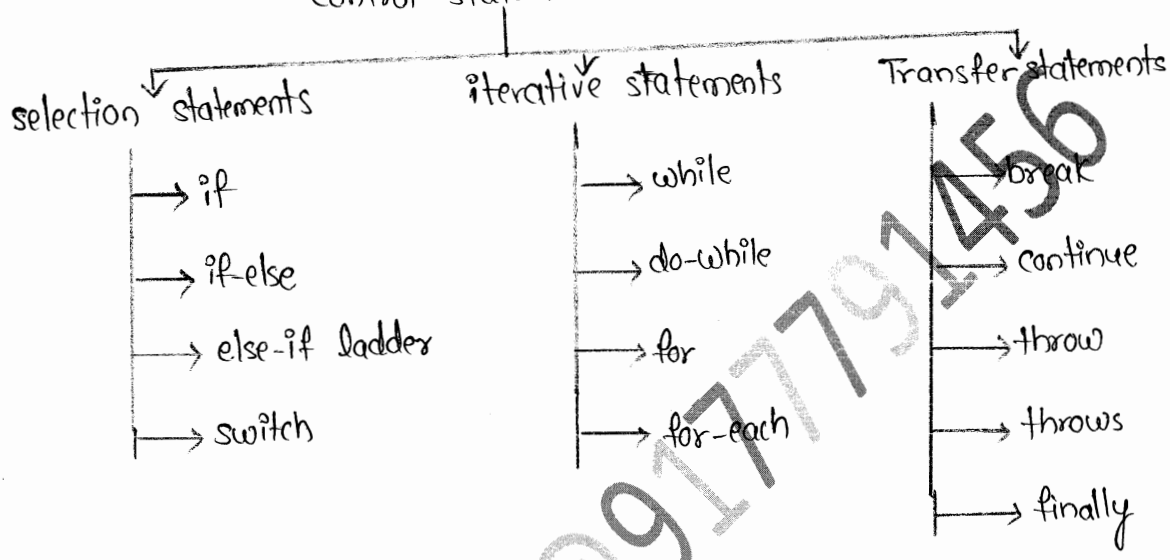
import java.util.*; → valid

import java.Date.*; → valid

Control Statements

control statements decides in which order the statements are going to be executed.

control statements are three types.



Selections statements :

1. if

syntax :

```
if (condition)  ← ①
{
    statements; →②
}
    statements; } →③
```

→ by using 'if' we can check the condition

→① condition should be always boolean type, i.e it can be either 'true' or 'false'.

→ if condition is true, if-block is going to be executed, if condition is false, if-block is not going to be executed.

→ ② whenever the condition is true, this block is to be executed.

→ ③ This one is going to be executed whether condition is true or false

2. if-else :-

Syntax :
```
if (condition)
{
    statements ; } —②
}
else
{
    statements ; } —③
}
    statements ; } — ④
```

→① condition should be always, boolean, i.e that can be either true or false

→② if the condition is true then this if-block is going to be executed

→③ if the condition is false then it is going to be executed else block.

→④ if the condition is true/false this statements are going to be executed.

3. switch statement

→ by using switch, we can perform operations on choice-based or option-based.

→ The condition in switch may be of type integer, String, byte, short, int long, float, double, Enum and boolean.

→ by using 'break' statement in switch, we can come out from the loops.

→ if no case matched with the condition then 'default' case is going to be executed.

→ Fall through inside switch: if there is no break from the matching case, all the rest of the cases are going to be executed until break finds then it is called fall through inside switch mechanism.

```
Syntax:  switch (condition)
         {
          case condition1:
                        = statements;
                        break;

          case condition2:
                        = statements;
                        break;
           case condition3:
                        = statements;
                        break;
                :
           case conditionn:
                        = statements;
                        break;
           default:
                        = statements;
                        break;
         }
```

Iterative statements:

If we want to execute the statements repeatedly and continuously then we can go for iterative statement or looping statements.

1. while loop: while loop is the best suitable if the number of iterations are unknown in advance.

```
syntax: initialization section;
        while (condition)
                 ↳ boolean
        {
          =
```

= statements;
increment/decrement;
}
≡ statements;

* initialization and increment/decrement sections are mandatory. If we don't write increment/decrement section, we are going to get an error as 'StackOverflowException'.

2. do-while loop:- If we want to execute the loop body atleast once irrespective of condition then we can go for do-while.

Syntax: initialization section;
do
{
≡ statements;
increment/decrement;
}
while (condition)

* In 'do-while', first the loop is going to be executed and then it checks for the condition.

** Difference between while and do-while:

In while, first the condition is going to be executed and if the condition is true, then loop body will be executed.

In do-while, first loop body is going to be executed once and then it is going to check the condition. If the condition is true, it will execute the loop body and if the condition is false, simply it comes out of the loop.

3. for- loop

If the number of iterations are known in advance then we can go for 'for-loop'

Syntax1: for(initialization ; condition ; increment/decrement)
{
= statements;
}

syntax2: initialization section;
for(; condition ; increment/decrement)
{
= statements;
}

syntax3: initialization section
for(; condition ;)
{
= statements;
increment/decrement;
}

Syntax4: for(initialization ; condition ;)
{
= statements;
increment/decrement;
}

Syntax 5: for(initialization ; ; increment/decrement)
{
= statements;
}

Note: In 5th syntax, if we don't specify the condition in for loop, by default

that condition will always become true.

Transfer statements:

① break: If we want to come out of the loop based on condition, we can go for break statement.

→ by using break, we can stop or break the execution.

→ we can use the 'break' inside switch and loops.

```
ex: public class Demo {
      public static void main(String[] args) {
        for(int i=1; i<=10; i++)
         {
         System.out.println(i);
         }
        if(i==5)
        {
         break;
        } } } }
```

output: 1 2 3 4

2. continue:

By using continue, we can skip the current iteration and we can continue the loop body normally.

* continue statement skips the current iteration without breaking the loop execution.

```
ex: public class Demo
      {
        public static void main(String[] args)
         {
```

```
for(int i=1; i<=10; i++)
 {
  if(i==5)
   {
    continue;
   }
  System.out.print(i);
 }
}
```

Output: 1 2 3 4 6 7 8 9 10

Note:

* we can use break only inside the switch and loops.
* we can use continue only inside the loops.
* we cannot use break and continue directly inside 'if' block.

Initialization Of Variables

Initialization of variables can be performed in 4 ways.

1. Direct initialization
2. Through keyboard
3. Through Object
4. Through constructors.

Direct initialization:

```
ex: public class Demo{
        int a=10;
        int b;      } direct initialization
        b=20;
        public static void main(String[] args)
        {
        System.out.println(a); } //error becoz a & b are non-static variables, we
        system.out.println(b);   cannot access directly.
        Demo demo = new Demo();
        System.out.println(demo.a);
        system.out.println(demo.b);
        } }
```

Through keyboard:-

If you want to read the values from keyboard, java provided one predefined class 'called' "Scanner" class.

* This Scanner class is having lot of methods to read the values.

Ex: Scanner scanner = new Scanner(System.in);

```
System.out.println("Enter a String");
String s1 = scanner.next();
System.out.println("String value is :" + s1);
```

datatype	method used to read
String, char	next()
Integer	nextInt()
Float	nextFloat()
Byte	nextByte()
short	nextShort()
long	nextLong()

3. Through Object:

→ we can perform initialization of variables through object

→ by using '.' operator we can perform the initialization of variables.

syntax: ObjectreferenceVariable name.variable name = variableValue;

ex:
```
TestDemo testDemo = new TestDemo();
testDemo.a = 10;
testDemo.b = 20;
```

Through Constructor:-

Constructors are used to initialize the variables.

* Constructor name should be same as class name.

* If the class is declared as public, then constructor also should be declared as public.

syntax: access modifier Constructorname()
{
=
}

* If our program doesn't have any constructors, the compiler will generate or create a constructor called default constructor means a constructor with no parameters.
* If our program has atleast one constructor then compiler won't generate any default constructor.

this() & super():-

* Inside constructor by default the first keyword is either this() or super().
* super() and this() are constructor calls and we can use these calls only inside constructors.
* By using super() constructor call, we can call parent class constructor i.e, super() call refers to the parent class constructor.
* By using this() constructor call, we can call current class constructor i.e this refers to the current class constructor.

this & super:-

→ this and super are keyword

→ By using this, we can refer current class variables.

→ we can use 'this & super' any where except static area

Q: When the constructor is going to executed?

whenever we create object then 'constructor is going to be executed by JVM.

note: If you want to perform initialization of variables then create constructor and perform operations.

Q: write a program for constructor?

```
public class ConstructorDemo {
  int a;
  String name;
  int b;
  public ConstructorDemo(int a1, String str, int b1)
   {
   a=a1;
   name=str;
    b=b1;
   }
public ConstructorDemo()
{
System.out.println("Default constructor");
}
public static void main(String[] args)
{
     ConstructorDemo cd=new ConstructorDemo();
     ConstructorDemo cd1=new ConstructorDemo(10,"hello",20);
     ConstructorDemo cd2=new ConstructorDemo(30,"java",60);
  }}
```

→ we can overload the constructors.

→ For constructors returntype is not allowed even 'void' also.

Example1:

```
public class Test {
 int a;
 public Test(int a)
 {
 this.a=a;
 }
}
class Test1 extends Test
{
 int a;
 String str;
Test1 ( int a)
 {
 this.a=a;
 super(10);
 this();
 }
Test1 ( int a , String str) {
 this.a=a;
 this.str=str;
 this (10); }
 Test1 ( ) { }
 System.out.println("Default"); }
 public static void main(String[] args)
 {
 Test1 test1 = new Test1 (10, "Java");
 }
}
```

Example 2:

Constructor overloading:

we can overload the constructor based on our requirement.

```
public class Test {
 int a;
 String str;
 float b;
public Test ( int a1 , String str1){
 a=a1;
 str = str1;
 }
public Test() {
System.out.println("Default");
 }
public Test( float b1)
 {
  b=b1;
 }
public static void main(String[] args)
{
 Test test1 = new Test (10, "Hello");
 Test test2 = new Test();
 Test test3 = new Test (10.5f);
 }
}
```

Coding Standards :-

* Whenever we are implementing programs, coding standards are very important.
* let us discuss one by one.

① Coding standard for classes :

* class name should be very meaningful and every class name should starts with capital letter.

ex: public class PrimeNumber
{
=
}

by seeing the class name we should be in a position to understood what class that class is having inside of it.

* Every class name should start with capital letter and if it having multiple words each word should start with capital letter.

ex: PrimeNumberDemo
Test
HelloTest
String
Object
:

② Coding standards for package :

package names should be very meaningful. when you are giving package names, there is one real time naming convention is there.

syntax: domain name in reverse . module name . sub module name

ex: package pack1 ; → not recommended

package com.rameshSoft.selenium ; ✓ recommended.

③ Coding standards for interface:

Every interface name should start with capital letter and it is having multiple words, each word starts with capital letter.

ex: ATMDemo

Serializable

Runnable

④ Coding standards for methods

every method name should starts with alphabet and if it contains multiple words, every word should starts with capital letter. except first word.

ex: testDemo()

gmailLoginTestDemo()

getName()

⑤ Coding standards for variables:

Every variable name should starts with lower case and if it contains multiple words, each word should start with capital letter.

ex: id ;

name ;

rollNumber;

collegeName;

→ each and every variable name 1st letter must starts with an alphabet.

(The length of the variable should not contain morethan 32 characters).

→ no special symbols are allowed except '_' (underscore)

→ no keywords to be used as variable name.

⑥ Coding standards for constants:

→ should contain only uppercase letters and it is having multiple words, these words separated with underscore ('_') symbol.

ex: MAX_PRIORITY;

NORM_PRIORITY;

MAX_PRIORITY

*Path Vs classpath:

path:- path variable represents the location where binary executables are available.

→ if we are not setting the path the java & javac commands cannot work

E:/Ramesh > javac ↵
javac not recognized as external or internal command.

E:/Ramesh > set path = c:\Programfiles\java\jdk1.7.0\bin;

E:\Ramesh> javac ↵ it will work now

class path:- class path can be used to specify the location where required '.class' files are available.

→ if we are not setting the classpath then the program won't compile and won't run.

Jdk vs Jre v/s Jvm:-

① jdk (java development kit): JDK provides environment to develop and run java applications.

② Jre (Java Runtime Environment): provides environment to run the java applications.

③ Jvm (Java virtual machine): It is responsible to execute java programs.

Object class :-

For any java class the Parent class is 'Object' class.

→ Every class in java whether we extends or not bedefault 'Object' class is the Parent class either directly or indirectly.

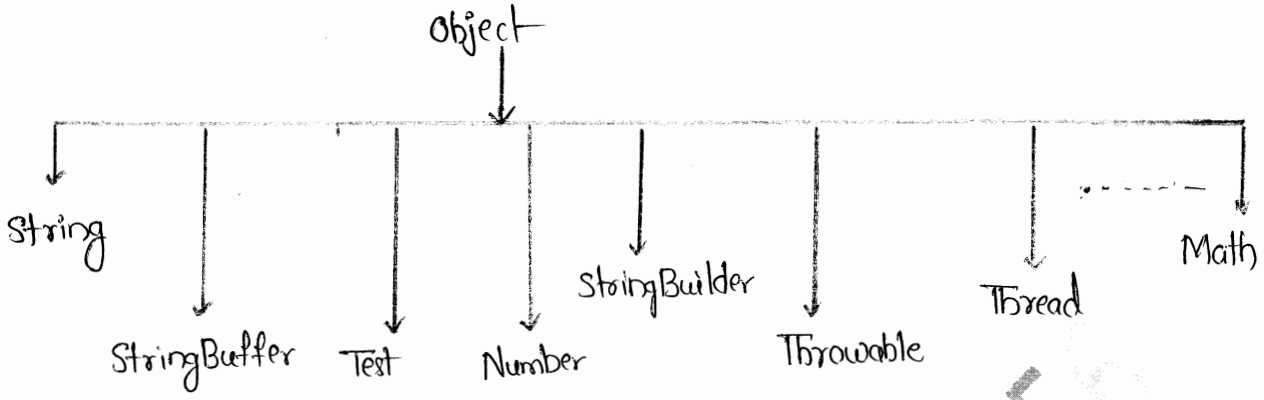
```
public class Test          public class Test extends Object
{                    ⟹    {
=                          =
}                          }
```

→ The object class contains the following 11 methods, these are by default available to every child class.

1. public String toString()
2. public native int hashCode()
3. public boolean equals(object o)
4. protected native object clone()
5. public final native void notify()
6. public final native void notifyAll()
7. public final void wait()
8. public final void wait(long milliseconds)
9. public final void wait(long milliseconds, int ns)
10. public final native class getClass()
11. protected void finalize()

→ Object class is available in java.lang package and this package is not required to import explicitly because all classes and interfaces

available in this package are by default available to every java program.



String (c) :

String is a class which is present in java.lang package. (ie java.lang.String)

→ If the content is fixed and won't change frequently then we should go for String ie String is like a immutable type

String class Constructors:

① String s = new String();

creates an empty String object

② String s = new String (StringBuffer sb);

creates an equivalent String object for the given StringBuffer

③ String s = new String (char[] ch);

creates an equivalent String object for the given char[] array.

ex: char[] ch = {'a', 'b', 'c', 'd'}

String s = new String (ch);

System.out.println (s); "abcd".

④ String s = new String (byte[] b)

creates an equivalent String object for the given byte[] array

Methods in String class:

① public char charAt (int index):

returns the character locating at specified index.

ex: String s = "Ramesh";

System.out.println(s.charAt(2)); // m

System.out.println(s.charAt(9)); // RE: StringIndexOutOfBoundsException

② public string concat(String s);

to perform concatination and we can use '+' and '+=' operators for to perform concatination.

* String s = "Ramesh";

s = s + "Soft";

System.out.println(s); // RameshSoft

* String s = "Ramesh";

s += "Soft";

System.out.println(s); // RameshSoft

* String s = "Ramesh";

s = s.concate("soft");

System.out.println(s); // RameshSoft

③ public boolean equalsIgnoreCase(String s);

By using this method we can perform content comparison and it ignorer case.

ex: String s = "Ramesh";

System.out.println(s.equalsIgnoreCase("ramesh")); ✓ true

System.out.println(s.equalsIgnoreCase("Ramesh")); ✓ true.

④ public int length():

by using this method, return number of characters parent in a String.

String s = "Ramesh";

System.out.println(s.length()); //6

⑤ public String subString(int begin):

It returns the substring from begin index to end of the String.

String s = "Ramesh";

System.out.println(s.subString(3)); //esh

⑥ public String subString(int begin, int end):

returns the string from begin index to end-1 index.

String s = "Rameshsoft";

System.out.println(s.subString(1, 4)); // ames

⑦ public String toLowerCase():

to convert all upperCase to lower case

String s = "RAMESHSOFT";

System.out.println(s.toLowerCase()); rameshsoft

⑧ public String toUpperCase();—

to convert all lower case letters to uppercase.

ex: String s = "ramESHSoft";

System.out.println(s.toUpperCase()); //RAMESHSOFT

⑨ public string replace(char old, char new):—

It replaces with old value with new value

ex: String s = "abcd";

System.out.println(s.replace('d', 'e')); //abce

⑩ public String trim():—

It removes the blank spaces present at beginning and ending of the string but not blank spaces present at middle of the string.

⑪ public int indexOf(char ch)

→ It returns the index of first occurence of specified character.

⑫ public int lastIndexOf(char ch):—

→ It returns index of last occurence of specified character.

ex:— String s = "RameshSoft";

System.out.println(s.indexOf('a')); o/p: 1

System.out.println(s.lastIndexOf('s')); // o/p: 6

StringBuffer :-

StringBuffer is a class present in java.lang package, if the content will change frequently then go for StringBuffer.

* Every method present in StringBuffer is synchronized.

StringBuilder :-

StringBuilder is a class present in java.lang package.

If the content will change frequently then go for StringBuilder

* Every method present in StringBuilder is not synchronized.

Q Difference b/w length(), length and size()?

length() :- This is a method, used to return number of characters present in the string.

syntax: public int length()

length :- length is a variable applicable for Array objects.

size() :- This is a method, used to return the size of collection object in the form of integer. i.e public int size()

Ex:-
```
String s = "RameshSoft";
System.out.println(s.length());//10

int[]a = new int[]{1,2,3};
System.out.println(a.length);//3
```

```
ArrayList al = new ArrayList();
al.add(10);
al.add(20);
System.out.println(al.size());//2
```

① write a Java program to reverse a String with API and without API.

i) public class ReverseStringwithAPI

```
{
  public static void main(String[] args)
   {
   Scanner scanner = new Scanner(System.in);
   System.out.println("please enter the String");
   String s = scanner.nextLine();
   String reverse = "";
  reverse= new StringBuffer(s).reverse().toString();
   System.out.println("Reverse of String is:"+reverse);
     }
}
```

ii) public class ReverseStringWithoutAPI

```
{
  public static void main(String[] args)
   {
    Scanner scanner = new Scanner(System.in);
     String reverse = "";
      System.out.println("plz enter the String");
       String s = scanner.nextLine();
     for(int i = s.length()-1 ; i>=0 ; i--)
      {
          reverse= reverse+s.charAt(i);
       } System.out.println("Reverse of String is:"+reverse); } }
```

(2) Write a Java Program for String Palindrom ?

```
public class StringPalindrome{
  public static void main(String[] args){

    Scanner scanner = new Scanner(System.in);

    System.out.println("Please enter a String");

    String str = scanner.nextLine();

    String reverseStr = "";

    char[] chars = str.toCharArray();

   for(int i = chars.length-1; i>=0; i--)
    {
      reverseStr = reverseStr + chars[i];
    }
  System.out.println("Reverse String is :"+reverseStr);

   if(str.equals(reverseStr))
   {
   System.out.println("String is Palindrome");

   }
  else
   {
   System.out.println("String is not palindrome");

    }

   }
}
```

③ Write a Java Program to count the number of words repeated in a String?

```
public class RepeativeWordCountDemo {

  public static void main(String[] args)
   {
     wordCount("cat dog dog cat pen pencil pencil pen");
     wordCount("AAA BBB CCC AAA BBB DDD EEE");
   }
public static void wordCount(String str)
 {
HashMap<String, Integer> hmp = new HashMap<String, Integer>();
 String[] array = str.split("\\s+");
for(String s : array)
 {
   if(hmp.containskey(s))
    {
     hmp.put(s, hmp.get(s)+1);
    } else
    { hmp.put(s, 1);
    }
 }
 Set<String> keys = hmp.keySet();
 for(String key:keys)
  {
    if(hmp.get(key)>1)
      { System.out.println("Duplicate word is:"+key+"occured"+hmp.get(key)+"
                                                          times");
      } } } }
```

④ Write a Java Program for removing whitespaces in the given String?

```
public class RemoveWhiteSpacesDemo
{
  public static void main(String[] args)
  {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Enter the String");
    String str = scanner.nextLine();
  //1st way using replaceAll()
    str=str.replaceAll("\\s", "");
    System.out.println(str);
  //2nd way using replace()
    str=str.replace(" ","");
    System.out.println(str);

  //3rdway without using API
    char[] ch = str.toCharArray();
    StringBuffer sb = new StringBuffer();
    for(int i=0; i<ch.length; i++)
    { if((ch[i]!=' ')&&(ch[i]!='\t'))
        {
          sb.append(ch[i]);
        } }
    System.out.println(sb); }}
```

⑤ Write a Java Program for merge of two Strings?

Rule: if String1 = "AA", String2 = "BBB"

output should be ABABB

```
public class MergeStringsDemo {
 public static void main(String[] args) {
     merge("AA","BBBB");
     merge("XXXX","YYYY");
     merge(" ","VVV");
     merge("BBB"," ");
     merge("GGGGGG","H");
     // merge(null, null); RE: NPE
     }
public static String merge(String s1, String s2)
 {  int i=0;
    int j=0;
    String mergeString="";
  while(i<s1.length()&&j<s2.length()
    {
     mergeString = mergeString+s1.charAt(i)+s2.charAt(j);
      i++;
      j++;
    }
    while(i>s1.length()||j<s2.length())
      {
```

```
    mergeString = mergeString + s2.charAt(j);
      j++;
    }
  while(j>s2.length() || i<s1.length())
   {
   mergeString= mergeString + s1.charAt(i);
      i++;
    }
 System.out.println("Merging of both strings are :"+ mergeString);
 return mergeString;
  }
}
```

⑥ Write a Java Program to check whether given two strings are anagrams or not?

```
public class AnagramsDemo {
    public static void main(String[] args)
      {
        String s1 = "Hello";
        String s2 = "elloH";
    s1 = s1.toLowerCase();
    s2 = s2.toLowerCase();
     char ch1[] = s1.toCharArray();
     char ch2[] = s2.toCharArray();
```

```
boolean status;
Arrays.sort(ch1);
System.out.println(ch1);
Arrays.sort(ch2);
System.out.println(ch2);
status = Arrays.equals (ch1, ch2);
if( status )
   System.out.println(" both are anagrams");
else
 System.out.println(" both are not anagrams");
  }
}
```

⑦ Explain with example how String is immutable?

```
public class ImmutableStringDemo
{
   public static void main(String[] args)
   {
      String s1 = "Hello";
        s1.concate (" 123");
      System.out.println (s1); // Hello
      StringBuffer sb = new StringBuffer("Java");
          sb.append(" selenium");
      System.out.println(sb); // Java selenium } }
```

⑧ Write a Program for finding Number of words in a String and No.of occurences of a specific word in a given String.

```
public class WordOccurenceCount
 {
  public static void main(String[] args){

   String str = " Hai Hello Hello Hai Java Selenium";

   String[] array = str.split(" ");

   System.out.println("No.of words in a String are : "+array.length);

    String search = "Hello";

  System.out.println("length of actual String is: "+str.length());

  System.out.println("length of specific search String is : "+search.length());

  int lengthAfterReplace = str.replaceAll("Hello", "");

int wordCount = (str.length() - $engthAfterReplace)/search.length();

  System.out.print(" Hello word occurence count is : "+wordcount);

    }

}
```

⑨ Explain the difference b/w "==" and equals(); with example.

```
public class EqualsDemo {
    public static void main(String[] args)
    {
      String str1 = "Hello";
      String str2 = "Hello";
      String str3 = new String("Hello");
      String str4 = new String("Hello");
    System.out.println(str1==str2); // true
    System.out.println(str1.equals(str2)); // true
     System.out.println(str1.==str3); // false
     System.out.println(str1.equals(str3)); // true
     System.out.println(str4==str3); // false
     System.out.println(str4.equals(str3)); // & true
       // for StringBuffer & StringBuilder
        StringBuffer sb1 = new StringBuffer("Java");
        StringBuffer sb2 = new StringBuffer("Java");
         StringBuffer sb3 = sb2;
       System.out.println(sb1==sb2); // false
       System.out.println(sb1.equals(sb2)); // false
       System.out.println(sb3==sb2); // true
         System.out.println(sb3.equals(sb2)); // true  } }
```

(10) Write a Java Program for converting String to int and int to String?

```
public class StringIntConversions {

    public static void main(String[] args)

    {
      String s = "10";

      int x = 20;

// String to int

    int x1 = Integer.valueOf(s);

    int x2 = Integer.parseInt(s);

    // int to String

    String s1 = String.valueOf(x);

    String s2 = Integer.toString(x);

System.out.println(s1);

System.out.println(s2);

System.out.println(x1);

System.out.println(x2);

    }

}
```

Eclipse:-

Eclipse is an IDE (Integrated Development Environment) mean by using eclipse we can write and run the programs.

*Eclipse is an open source tool.

How to download eclipse:-

open google → type download — eclipse Luna → click on "Java IDE EE Developer" link → click on 'windows x64 bit → click on download button → eclipse will be downloaded in the form of zip file → extract zip file.

Workspace:- workspace is a place where we are going to write or maintain all our Java programs or java projects. simply it is a location to save our project.

→ whenever you open eclipse, it will asks about the workspace.

prerequisites to use eclipse:-

→ In order to use eclipse, we need Java to be installed in our machine, other-wise it will show error when we try to open.

Advantages:-

* Eclipse is the big assistance to the programmer i.e programmer is not responsible to write each and every thing because it has bunch of shortcuts.

ex: Syso + ctrl + space → System.out.println(" ");

main + ctrl + space → public static void main(String[] args) { }

* In eclipse at the time of writing code itself, the code is going to be compiled.

ex: If we forget to give semicolon (;) at the end of statement, immediately we are going to get compile time error saying that "syntax error: please insert ';' to complete the statement.

* If we use eclipse latest versions, we are not required to configure tools like ANT, MAVEN etc...

Eclipse Modules:—

whenever we are working with eclipse we need to focus on 3 areas.

① Project explorer / package explorer

② Console

③ Eclipse editor

① Project Explorer :— It is a place where we can see our projects.

↦ whenever we create new project all the projects are going to reside under project explorer / package explorer.

② Console :— Console is a place where we can see the output.

③ Eclipse editor :- It is a place where we can write our java programs

Note:— select 'perspective' as 'Java'.

How to create a project:—

we can create project in 2 ways in Eclipse.

way1: File → new → Java project → enter project name → next → finish.

way2: Right click on project explorer area → new → project → Java proje-ct → next → enter project name → next → finish.

whenever we create project, by two folders are going to be created.

1. src folder
2. JRE System Library

src folder :- It is a folder or package, where we are going to write all our Java programs.

JRE system Library :- Java Runtime Environment System Library is used to run our programs.

How to create a package :-

It is always recommended to create packages under 'src' only.

Select 'src' → right click → new → package → enter package name → finish.

How to create a class :-

we need to always create classes under packages.

select package → right click → new → class → enter class name → finish

How to download Selenium:-

open firefox browser (any browser) → go to google.com → enter 'download selenium' → click on download selenium (www.seleniumhq.org/download) → click on 'download for Java" → ok → once you downloaded it will come with zip folder and we need to extract that zip folder.

Selenium integration with eclipse:-

If we want to use selenium functionality in eclipse, we need to configure or we need to add selenium jar files.

* we can add selenium jar files in 2 ways.

① Select the project → right click → build path → configure build path → Java build path → libraries → add external Jars → add all corresponding selenium jar files → click on ok.

② Right click on project → properties → Java build path → library → add external jars → select all selenium jar files → click on 'ok'.

Selenium

Introduction:-

Selenium is an open-source and a portable automated software testing tool for testing web applications. It has capabilities to operate across different browsers and operating system.

Selenium is not just a single tool but a set of tools that helps testers to automate webbased applications. more efficiently.

* Selenium is a portable software testing Framework for web applications.
* Selenium was created by Jason Huggins in 2004. An engineer at thought works, he was working on a web application that required frequent testing.
* Selenium is a web application testing framework that allows you to write tests in many programming languages like java, c# , groovy, perl , PHP Python and Ruby. Selenium deploys on windows. Linux and MAC OS.
* It is an open-source project released under the Apache 2.0.
* There are 4 selenium tools available in selenium suite

1. Selenium RC
2. Selenium IDE
3. Selenium WebDriver
4. Selenium Grid.

Among all the modules, Selenium webDriver is very popular in the market.

Selenium WebDriver :—

→ webDriver in Selenium is an interface.

→ Selenium webDriver is also known as Selenium 2 and used for web as well mobile application testing.

→ it is freeware software testing tool and mostly used as a regression testing tool for web and mobile applications.

→ Selenium WebDriver software testing tool don't require selenium server for running test.

→ WebDriver is using native automation from each and every supported language for running automation scripts on browsers.

→ WebDriver supports web as well as mobile application testing so you can test mobile applications (iphone or android).

→ Supporting latest version of almost all browsers.

→ webDriver controls the browser itself.

→ It provides a user friendly API which you can understand and explore easily as a result it will help to read & maintain your script easier.

Selenium Grid :—

Selenium Grid is a tool used to run parallel test across different machines and different browsers simultaneously which results in minimized execution time.

Selenium WebDriver :-

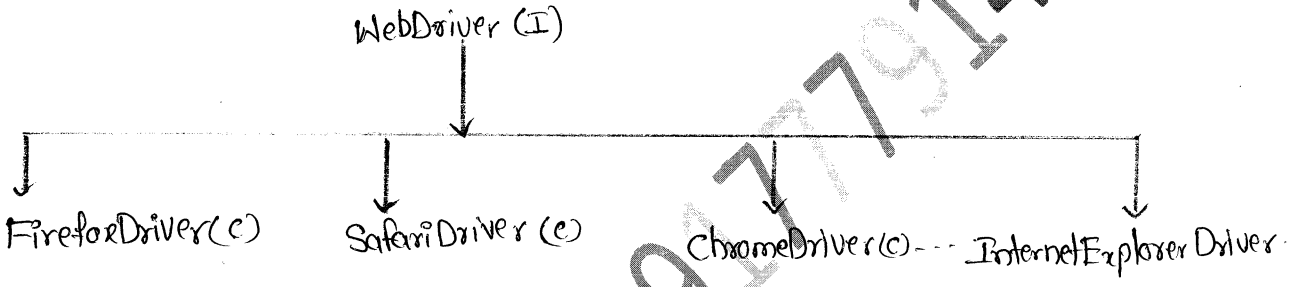
→ By using Selenium WebDriver we can automate the web applications.

→ As WebDriver is an interface, it needs to be implemented.

The following are the WebDriver implemented classes.

1. FirefoxDriver 2. ChromeDriver 3. InternetExplorerDriver

4. SafariDriver ...etc



Pseudo code for WebDriver interface:

```
interface WebDriver
{
  void get(String url);
  WebElement findElement(-);
  List<WebElement> findElements(-);
  String getTitle();
  String getPageSource();
  string getCurrentUrl();
  string getWindowHandle();
  Set<String> getWindowHandles();
}
```

```
public class FirefoxDriver implements WebDriver
{
    public void get(String url)
    {
    =
    }
=
}
public class ChromeDriver implements WebDriver
{
    public void get(String url)
    {
    =
    }
    public String getTitle()
    {
    =
    }
=
}
public class InternetExplorerDriver implements WebDriver
{
=
}
public class SafariDriver implements WebDriver
{
=
}
```


Methods present in WebDriver interface:

getTitle():- whenever we call this method, it always returns the title of the page in the form of String.

Syntax: `public String getTitle()`

ex: WebDriver driver = new FirefoxDriver();

driver.get("https://www.gmail.com");

String title = driver.getTitle();

System.out.println(title);

get():- By using this method, we can enter url and it doesn't return anything

syntax: `public void get(String url)`

ex: WebDriver driver = new FirefoxDriver();

driver.get("https://www.google.com");

close():- Whenever we call close(), current window(or) focused window is going to be closed.

syntax: `public void close()`

ex: driver.close();

quit():- whenever we call 'quit()' all append windows will be closed.

Syntax: public void quit()

click():- by using click(), we can perform clickable operations on web elements.

syntax: `public void click()`

ex:- WebElement element = driver.findElement(By.id("next"));

element.click();

clear() :- whenever we call clear() on web element, it will clear that particular text field.

Syntax: `public void clear()`

```
ex: WebElement ele_username = driver.findElement(By.id("Email"));
    ele_username.clear();
```

sendkeys() :- by using sendkeys(), we can pass the test data to the webel-ements. On webelements only we can call this sendkeys().

Syntax: `public void sendkeys(String testdata)`

```
ex: WebElement ele_username = driver.findElement(By.id("Email"));
    ele_username.sendkeys("rameshsoft.selenium@gmail.com");
```

findElement() :- by using this method, we can identify only one webelement and it return that webelment.

Syntax: `public WebElement findElement()`

```
ex: driver.findElement(By.id("Email")).sendkeys("rameshsoft.selenium@gmail.com");
```

findElements() :- whenever we call findElements(), it is going to identify all the corresponding web elements in the form of List.

Syntax: `public List<WebElement> findElements(By arg);`

```
ex: List<webElement> links = driver.findElements(By.tagName("a"));
```

getCurrentUrl() : whenever we call this method, it will return the url of the current page in the form of String.

syntax: `public String getCurrentUrl()`

```
ex:- driver.get("https://www.gmail.com");
     String url = driver.getCurrentUrl();
```

getPageSource(): whenever we call getPageSource() we will get the source code of the page in the form of String.

syntax: `public String getPageSource()`

```
ex: driver.get("https://www.google.com");
    String pageSource = driver.getPageSource();
    System.out.println("source code of the page is:"+pageSource);
```

getWindowHandles():- whenever we call this method, it will return all the windows names in the form of set of Strings.

Syntax: `Set<String> getWindowHandles()`

```
ex: driver.get("https://www.gmail.com");
    Set<String> windowNames = driver.getWindowHandles();
    System.out.println("Window names are :"+windowNames);
```

manage():- by using this manage(), we can perform managing operations like maximizing the window, time initializations and performing operations on keys.

Syntax: `public Options manage()`

ex: driver.manage().window().maximize() → to maximize window

driver.manage().deleteAllCookies(); → to delete all the cookies.

navigate(): – if we want to perform navigation operations we can use navigate().

syntax: `public Navigation navigate();`

→ by using navigate(), we can enter the url

ex: driver.navigate().To("https://www.gmail.com");

→ by using navigate(), we can perform navigations like refresh, forward backward etc...

ex: driver.navigate().forward();

switchTo(): – by using switchTo(), we can switch to windows, frames, alerts ...etc.

syntax: `public TargetLocator switchTo()`

ex: driver.switchTo().window(-);

driver.switchTo().frame(-);

driver.switchTo().alert();

How to perform operations on Web element :-

we can perform operations on web elements in the following way

step① : Identify or find the correspond web element

In a web page, each and every object called as web element.

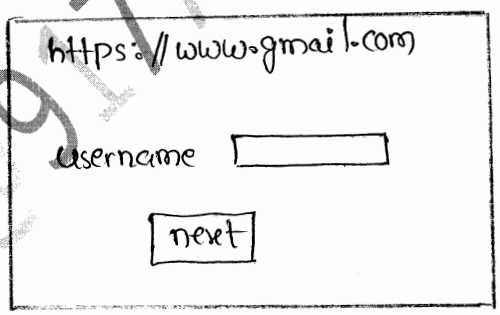
Each web element contains some properties or attributes and based on those properties or attributes we can identify web elements.

In order to identify or find a web element, webDriver has one method called 'findElement()' method and this method always ask how you want to find the element (based on id, name, xpath etc..).

ex:

html code for gmail page is

```
< input id="Email" type='email' ... >
  ①     ②     ③     ②     ③
```



① – tagname

② attribute name/property name

③ attribute value/property value.

ex: driver.findElement(By.id("Email"));

step②: Perform the corresponding action on that web element.

Once we identify that webelement, we can perform the corresponding action on that web element like data typing actions (sendkeys), clear and clickable actions.

```
ex: driver.findElement(By.id("Email")).clear();
    driver.findElement(By.id('next')).click();
```

WebElement Identifications

There are many ways of identifying web elements.

1. id
2. xpath → absolute xpath
   → relative xpath
3. CSSSelector
4. name
5. class name
6. tag name
7. link Text
8. partial linktext

* if the webelement is having 'id' always identify that webelement with 'id' as it is unique always.

① By using "id" :-

*Identification of web element using "id" is always recommended because most of the times "id" is unique.

Q How to get or find web element attributes?-

Right click in a web page → inspect element (Q) → click on icon ( ) identify the web element → it will display properties.

* Eventhough if the webelement has name, class name, etc, always prefer 'id' if it has.

ex: username [ ]

html code : <input id="Email" name="Email" type="email" ..... />

driver.findElement(By.id("Email")).sendkeys("rameshsoft.selenium@gmail.com");

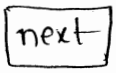
② By using 'name':

Identify web element using "name" is not recommended because there may be a chance of existing duplicate names.

ex: driver.findElement(By.name("Email")).sendkeys("rameshsoft.selenium@gmail.com");

③ By using "class name" :-

Identifying webelement using 'class name' is also not recommended because there may be a chance of existing duplicate class names.



identifying the 'next' button using class name.

ex: driver.findElement(By.className("rc-button rc-button submit")).click();

④ By using "linktext" :-

we can identify the webelement by using 'linktext' as well and it is recommended.

*If we want to identify the webelement using linktext, we need to observe the following things

i) compulsory it should starts with anchor tag (<a/>)

ii) that webelement must and should contain text

ex: Electronics

driver.findElement(By.linktext("Electronics")).click();

⑤ By using 'tagname' :- we can identify the webelement by using tagname.

ex: <input id='Email' type='email' ....>

driver.findElement(By.tagName("input")).click();

⑥ By using 'partial link text' :—

we can identify a web element by using 'partial LinkText' if and only if it should start with <a> tag and contain text.

ex: consider the above html code

driver.findElement(By.partialLinkText("Ele")).click();

⑦ By using xpaths:

There are two types of xpaths.

i) Relative xpath

ii) Absolute xpath

i. Relative xpath :—

→ Relative xpath always starts from exact match onwards.

→ Relative xpath always starts with ("//") double slash.

ex: //input[@id='email']

xpath using single attribute:

//*[@propertyname/attribute name='attribute value/property value']

(or)

//tagname[@propertyname/attribute name='attribute value/property value']

ex: <input id='Email' type='email' name='Email' ></input>

xpath: ① //*[@id='Email'] or //input[@id='Email']

② //*[@name='Email'] or //input[@name='Email']

note: Always recommended to write xpaths using tagname.

xpath using multiple attributes:-

syntax:

//tagname[@ attribute name = 'attribute value'][@attribute name = 'attributevalue']

(or)

//*[@attribute name = 'attributevalue'][@ attribute name = 'attributevalue']

ex: //input[@id = 'Email'][@name = 'Email']

xpath using 'AND' operator:-

we use 'AND' operator between two attributes in xpath

The web element will be identified by using 'AND' in xpath if only if those two attributes exist.

syntax:

//tagname[@Attribute name = 'attribute value' AND @attribute name = 'attributevalue']

(or)

*

ex: //input[@id = 'Email' AND @name = 'Email']

xpath using 'OR' operator:-

syntax:

//tagname[@attribute name = 'attribute value' OR @attribute name = 'attributevalue']

(or)

*

ex: //input[@id = 'Email' OR @name = 'Email']

xpath using 'contains' function:-

syntax: //tagname[contains(@attribute name, 'attribute value')]

ex: <input id = 'rameshsoft' name = 'ramesh'>

xpath → //input[contains(@id, 'mesh')]

xpath using 'starts-with' function:-

syntax: //tagname[starts-with(@attributename, 'attributevalue')]

ex: <input id='rameshsoft' name='ramesh' >

xpath → //input[starts-with(@id, 'rame')]

//input[starts-with(@name, 'rames')]

xpath using 'ends-with' function:-

syntax: //tagname[ends-with(@attributename, 'attributevalue')]

ex:

xpath → //input[ends-with(@id, 'soft')]

xpath using text :-

Syntax: //tagname [text()='text']

ex: RameshSoft

xpath → //a[text()='RameshSoft']

xpath using 'text with contains()':

Syntax: //tagname[contains(text(), 'textvalue')]

ex: RameshSoft

xpath → //a[contains(text(), 'ameshSoft')]

xpath using child and index:-

Identify the highlighted web element

```
<div id="rameshsoft">
  <div>
    <label>
    <input>
    <input>
  </div>
```

xpath → //div[@id='rameshsoft']/div/input[1]

If the web element doesn't have any attributes come from Parent

```
<div id="rameshsoft">
    <div>
      <label>
      <input>
      <input>
      <label>
    <div>
      <input>
    </div>
```

xpath → //div[@id="rameshsoft"]div[1]/input[2]

* identify all 'input' elements in below html code:

```
<div id="rameshsoft">
  <div>
    <label>
    <input>
    <input>
    <label>
```

```
<div>
   <input>
</div>
```

xpath → //div[@id='rameshsoft']/div/input

xpath using 'following' :-

By using 'following' we can identify all the webelements from the matching webelement onwards.

syntax:

① //tagname[@attribute name='attribute value']/following::*;

The above syntax is used to identify all the following webelements.

② //tagname[@attributename='attributevalue']/following::tagname

The above syntax is used to identify all the following with that tagname.

ex: //*[@id='Email']/following::*;

//*[@id='Email']/following::label[3]

xpath using 'preceding' :-

Syntax:

① //tagname[@attributename='attribute value']/preceding::*;

② //tagname[@attributename='attributevalue']/preceding::tagname;

ex: //*[@id='Email']/preceding::*;

//*[@id='Email']/preceding::input

//*[@id='Email']/preceding::input[3]

xpath using 'following-sibling' :-

By using 'following-sibling' we can identify the immediate sibling of the webelement.

syntax:

① //tagname[@attributename='attributevalue']/following-sibling::*;

② //tagname[@attributename='attributevalue']/following-sibling::tagname;

ex: //label[@id='Email']/following-sibling::*;

//*[@id='Email']/following-sibling::input[1];

xpath using preceding-sibling :-

syntax:

① //tagname[@attributename='attributevalue']/preceding-sibling::*;

② //tagname[@attributename='attributevalue']/preceding-sibling::tagname;

ex: //input[@id='Email']/preceding-sibiling::*;

//input[@id='Email']/preceding-sibling::label;

ii. Absolute xpath:-

Absolute xpath start with single slash('/') and it starts from Root <html> onwards.

ex: <html>

<tbody>

<input id='a'—>

<a ----->

```
<input id='b' - - - >
<input - - - - >
<a - - - >
<input - - - >
</tbody>
</html>
```

Relative xpath: //input[@id='b']

Absolute xpath: /html/tbody/input[1]/input[1]

(8) Css Selector :-

Css Selector has more advantage than xpath i.e css is much more faster than xpath.

* In internet explorer, css selector works faster, when compared with to xpath.

syntax: tagname[attribute name = 'attribute value']

ex: <input id='Email' name='Email' type='email' >

driver.findElement(By.cssSelector("input[id='Email']"));

driver.findElement(By.cssSelector("input[name='Email']"));

Special characters in css selector :-

In css selector we have two special characters

1. '.' (dot) operator
2. '#' (hash) symbol

1. '.' (dot) operator :-

dot (.) always refers to class name

syntax: tagname.classname or .classname

ex: <input id='Email' class='email'>

driver.findElement(By.cssSelector("input.email"));

(or)

driver.findElement(By.cssSelector(".email"));

2. '#' (hash) symbol :

hash (#) always refers to 'id'.

syntax: tagname#id (or) #id

ex: <input id='Email' ---->

driver.findElement(By.cssSelector("input#Email"));

(or)

driver.findElement(By.cssSelector("#Email"));

css using nth-child():-

In this case we have some id or class name and other attributes for different elements, we can go with nth-child().

syntax: css=tagname:nth-child(n).

n- represent child number.

ex:
```
<ul>
<li> C </li>
<li> C++ </li>
<li> C# </li>
```

```
<li> Java </li>
<li> Python </li>
<li> Ruby </li>
</ul>
```

WebElement cssele = driver.findElement (By.cssSelector ("li : nth-child (n)"));

Here if we can see the ul li parent child structure. we have only tagnames which are common to everyone. Here if we want to locate at 'Java' we will put n=4.

WebElement cssele = driver.findElement (By.cssSelector("li : nth-child (4)"));

CSS Selector using 'contains' (*)

Syntax: tagname[attributename * = 'attribute value']

ex: driver.findElement (By.cssSelector("input [id * = 'Email']"));

CSS selector using 'start-with' (^):-

^ symbol used for start-with method.

syntax: tagname[attributename ^ = 'attributvalue']

ex:- driver.findElement(By.cssSelector("input [id ^ = 'user']"));

CSS selector using 'ends-with' ($):-

'$' symbol acts as end with method.

Syntax: tagname [attributename $ = 'attributevalue']

ex: driver.findElement (By.cssSelector ("input [id $ = 'login']"));

CSS using single attribute:-

Syntax: tagname[attribute='attributevalue']

ex:- driver.findElement(By.cssSelector("input[id='user-login']"));

CSS Selector using multiple attributes:-

syntax: tagname[attribute1='attributevalue1'][attributename2='attributevalue2']

ex: driver.findElement(By.cssSelector("input[id='user-login'][name='log']"));

Firebug and Firepath:-

These are plugins to firefox browser in order to find and verify xpaths, css...

Download & install firebug:-

open google → type download firebug → click on https://getfirebug.com/downloads/ → click on appropriate version → add to Firefox → install now → ok.

Download & install firepath:-

open google → type 'download firepath' → click on Firepath::Add ons for Firefox link → Add to firefox → install now → ok → restart.

Note:-

After installation of firepath & firebug we can see an 'Ant' symbol in the window if it is installed successfully.

Q:- If the web element doesn't have any attributes then how to identify the web element?

<input>

Ans: * we can identify through their parents.

* we can identify through following & preceding sibling.

Ex: <input id="1" name="a">

<div id="2"---->

<input id="3" name="b">

<input id="4" name="c">

<input>

</input>

1) //*[@id='1']/following::input[3]
↳ index no.

2) //*[@id='1']/input[3]

3) //*[@id='4']/following-sibling::input

Note: index number starts from 1, 2...n

Automate the following test case (Gmail Login Test Case):

Test Case name	Expected value	Actual value
Gmail Login test case with correct username and wrong password.	1. Open firefox browser 2. enter url as www.gmail.com 3. enter username into 'username' text field 4. click on 'next' button 5. enter password 6. click on 'signIn' button 7. close the browser	Firefox Browser should be opened. Gmail page should be opened with username and text field. Data should be populated in username field. password page should be displayed. Data should be populated. It should display error page Browser should be closed.

Program:-

```
public class GmailLoginTestCase{
    public static void main(String[] args) throws Exception
    {
     WebDriver driver = new FirefoxDriver();
     driver.manage().window().maximize();
     driver.manage().deleteAllCookies();
     driver.get("https://www.gmail.com");
```

```
WebElement ele_username = driver.findElement(By.id("Email"));
        ele_username.clear();
        ele_username.sendkeys ("rameshsoft.selenium@gmail.com");
driver.findElement(By.id("next")).click();
        Thread.sleep(5000);
WebElement ele_password = driver.findElement(By.id("passwd"));
        ele_password.clear();
        ele_password.sendkeys("******");
driver.findElement(By.id("signIn")).click();
        Thread.sleep(5000);
        driver.quit();
    }
}
```

Synchronization

"Synchronization" is the process of making our application and our tool (Selenium) to wait each other inorder to get appropriate results".

* when we work with Automation, implementation of synchronization is very important.
* we can implement synchronization in four ways.

1. implicit wait
2. explicitwait
3. Thread.sleep(-)
4. Fluent wait

1) Implicit wait :-

* By using implicitwait, we can achieve synchronization.
* initialize the implicitwait immediately after opening the browser.

Syntax:

```
driver.manage().timeOuts().implicitlyWait(int time, sec/min/hours);
```

ex: driver.manage().timeouts().implicitlyWait (45, TimeUnit.SECONDS);

* whenever we initialize implicitlyWait() on the webDriver, it is applica-ble to all the webelements i.e no need to write implicitlyWait() for each and every webelement.
* If the specified time is 45 seconds, but the element is found in 5 seco-nds, then it simply skips the remaining time, i.e, it won't wait till

the specified time is completed.

* implicitlyWait() always asks you two parameters.

  a) time

  b) time in the form of Seconds/Minutes/Hours.

2. Explicit Wait :-

If you want to handle AJAX kind of calls, then we can go for explicit wait.

* In order to handle AJAX kind of calls, Selenium provided one predefined class called 'WebDriverWait'.

* This class has one non-static method called 'until(-)' and by calling this method, we can implement explicitWait.

Steps to implement explicit wait :-

① create WebDriverWait object by passing the time and WebDriver as an argument.

WebDriverWait wait = new WebDriverWait(driver, long time);

② Call 'until()' method, which is available in WebDriverWait class by passing expected conditions.

```
wait.until(ExpectedConditions.elementToBeClickable(WebElement element));
```

↓ it is a class from org.openqa.selenium.support.ui.WebDriverWait

* In explicit wait, if the element is found before the specified time, simply it skips the remaining time ie it wont wait until the completion of specified time.

ExpectedConditions class

ExpectedConditions is a class from org.openqa.selenium.

The following are some of the methods from ExpectedConditions class.

① public static WebElement presenceOfElementLocated (By locator)

An expectation for checking that an element is present on the Dom of a page.

This doesn't not necessarily mean that the element is visible.

This method returns WebElement.

② visibilityOfElementLocated():

Syntax: public static WebElement visibilityOfElementLocated (By locator)

An expectation for checking that an element is present on the Dom of a page and visible. Visibility means that the element is not only displayed but also has a height and width that is greater than zero.

This method returns WebElement.

Ex:- WebDriverWait wait = new WebDriverWait (driver, waittime)

wait.until(ExpectedConditions.visibilityOfElementLocated(locator));

③ we can also use the below method to check the element to be visible by WebElement.

ex: wait.until(ExpectedConditions.visibilityOf(element));

presenceOfAllElementsLocatedBy(-) :-

Syntax: public static List<WebElement> presenceOfAllElementsLocatedBy(By locator)

This method is, checking that there is at least one element present on a web page.

This returns List of WebElements.

elementToBeClickable(-):-

Syntax: public static WebElement elementToBeClickable(By locator)

This method is for checking an Element is visible and enabled such that you can click it. It returns the WebElement once it is located and clickable

elementToBeClickable(-) :-

Syntax: public static WebElement elementToBeClickable(WebElement element)

An expectation for checking an element is visible and enabled such that you can click.

This method returns WebElement (same WebElement) once it is clickable.

elementToBeSelected(-):-

Syntax: public static boolean elementToBeSelected(WebElement element)

This method is, for checking if the given element is selected.

This method returns true once the element is selected.

ex: WebDriver driver = new FirefoxDriver();

WebDriverWait wait = new WebDriverWait(driver, 50);

wait.until(ExpectedConditions.elementToBeSelected(locator));

invisibilityOfElementLocated(-)

Syntax:- public static boolean invisibilityofElementLocated(By locator)

This method used for checking that an element is either invisible or not present on the DOM.

This method returns true if the element is not displayed or element doesn't exist or stale element.

ex: WebDriver drivewait = new WebDriverWait(driver, waittime);

driverwait.until(ExpectedConditions.invisibilityOfElementLocated(locator));

invisibilityOfElementWithText(-) :-

Syntax: public static boolean invisibilityOfElementWithText(By locater, String text)

This method for checking that an element with text is either invisible or not present on the DOM.

This method takes two arguments 1> locator i.e used to find element

2> text of the element.

This method returns true if no such element, stale element or displayed text not equal that provided.

WebDriverWait wait = new WebDriverWait(driver, waittime);

wait.until(ExpectedConditions.invisibilityOfElementWithText(locator, text));

textToBePresentInElementLocated (-)

Syntax: public static boolean textToBePresentInElementLocated(By locator, String text)

This method for checking if the given text is present in the element that matches the given locator.

This method returns true onee the first element located by locator contains the given text.

frameToBeAvailableAndSwitchToIt()

Syntax: public static WebDriver frameToBeAvailableAndSwitchToIt(By locator)

This method for checking whether the given frame is available to switch to.

If the frame is available is switches the given driven to the specified frame.

It returns WebDriver instance after frame has been switched.

frameToBeAvailableAndSwitchToIt (-)

Syntax: public static WebDriver frameToBeAvailableAndSwitchToIt(int frameLocator)

public static WebDriver frameToBeAvailableAndSwitchToIt(String id/name);

public static WebDriver frameToBeAvailableAndSwitchToIt(WebElement ele)

Example:-

```
public class ExpectedConditions_Demo1 {
    WebDriver driver;
  @BeforeMethod
  public void setup() throws InterruptedException
  {
    driver = new FirefoxDriver();
    driver.get("https://www.gmail.com");

    driver.manage().window().maximize();

  //driver.manage().window().maximize();
    driver.manage().deleteAllCookies();
  driver.manage().timeouts().implicitlyWait(40,TimeUnit.SECONDS);
    }
  @Test
   public  void test() throws Exception
   {
WebElement username = driver.findElement(By.id("Email"));
 username.clear();
 username.sendkeys("rameshsoft.selenium@gmail.com");
driver.findElement(By.id("next")).click();
driver.findElement(By.id("passwd")).sendkeys("******");
driver.findElement(By.id("signIn")).click();
 WebDriverWait wait = new WebDriverWait(driver, 30);
wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath("//div[contains(te
                                                    xt(),'compose')]")));
 driver.findElement(By.xpath("//div[contains(text(),'compose')]")).click(); } }
```

Example 2:

```
public class ExpectedConditionsDemo2{

  @Test
  public void test() throws Exception
  {
   WebDriver driver = new FirefoxDriver();
   driver.get driver.manage().window().maximize();
        driver.manage().deleteAllCookies();
       driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
       driver.get("https://www.quackit.com/html/frames_example.html");
WebDriverWait wait = new WebDriverWait(driver, 60);
wait.until(ExpectedConditions.frameToBeAvailableAndSwitchTo("menu"));
wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//*[text()='ABC']"))
                                                                    ).click();

wait.until(ExpectedConditions.frameToBeAvailableAndSwitchToIt("menu"));

wait.until(ExpectedConditions.presenceOfElementLocated(By.xpath("//*[text()=
                                                    'DEF']"))).click();

       driver.close();
    }
}
```

3. Thread.sleep(-):

* sleep() is a static method which is available in 'Thread' class.
* By using Thread.sleep(-), we can pause the execution for a certain period of time.

Syntax:
```
public class Thread {
    public static void sleep(int time)
    {
    =
    }
}
```

* This 'Thread' class is available in 'java.lang' package.

Syntax: Thread.sleep(int milliseconds);

ex: Thread.sleep(30000); → 30 seconds

* Eventhough if the element is found in 3 seconds, it will wait till the specified time is completed i.e. it won't skip the time.
* If we write 'Thread.sleep(-)' for each and every webelement it kills our application performance. We need to use Thread.sleep(-) in the following scenarios.

→ Page navigations when we perform clickable operations.

→ Transparent window loadings.

→ Frames Handling.

Example :

```
public class SynchronizeDemo {

  public static void main(String[] args) throws InterruptedException {

  WebDriver driver = new FirefoxDriver();

  driver.manage().window().maximize();

  driver.manage().deleteAllCookies();

  driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);

  driver.get("https://www.gmail.com");

  driver.findElement(By.id("Email")).clear();

  driver.findElement(By.id("Email")).sendKeys("rameshsoft.selenium@gmail.
                                               com");

  WebDriverWait wait = new WebDriverWait(driver, 58);

  wait.until(ExpectedConditions.elementToBeClickable(driver.findElement(By.id("
                                                    passwd"))));

  driver.findElement(By.id("passwd")).clear();

  driver.findElement(By.id("signIn")).click();

  Thread.sleep(3000);

  driver.quit();

  }
}
```

Fluent Wait :-

FluentWait is a class from org.openqa.selenium.support.ui, and it is an implementation class of Wait interface.

Each FluentWait instance defines the maximum amount of time to wait for a condition and we can give the frequency which to check the condition.

we can also ignore any exception while polling element such as NoSuchElement Exception in selenium.

Example:-

```
public class FluentWaitDemo {
    public static void main(String[] args) throws Exception
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.get("https://google.com");
        driver.findElement(By.name("q")).sendKeys("selenium by
                                                  ramesh anapati");
        driver.findElement(By.name("btnG")).click();
        Thread.sleep(4000);
    driver.findElement(By.linkText("selenium: selenium WebDriver by
                                    Ramesh Anapati");
FluentWait<WebDriver> wait = new FluentWait<WebDriver>(driver).withT-
        imeOut(30, TimeUnit.SECONDS).pollingEvery(1, TimeUnit.SECONDS);

wait.until(new Predicate<WebDriver>() {
    public boolean apply(WebDriver argo) {
        boolean status = driver.findElement(By.xpath("//*[@id='profile1']/
                                    div[d1/dt/a")).isDisplayed();
        return status; } });
} }
```

Validations:—

Validation is the process of verifying or checking the application.

→ In real time compulsory we need to provide validations.

→ There two types of validations.

① pre-validations

② post-validations.

Pre-validations:—

Before performing any operations on the web element first we need to check whether it is 'displayed' and 'enabled' or not.

we can achieve pre-validation by using isDisplayed() and isEnabled().

isDisplayed():—

* whenever we call isDisplayed() method on the web element, it always checks whether that web element is displayed or not.
* If the web element is displayed it returns 'true', else it returns false.
  i.e, it always returns boolean value.
* we need to call isDisplayed() method on web elements only.

syntax: `public boolean isDisplayed()`

ex: `driver.findElement(By.id("Email")).isDisplayed();`

isEnabled():—

* whenever we call isEnabled() on the web element, it always checks whether that web element is enabled or not.

* If the web element is enabled, then it returns 'true', else it returns 'false' i.e, it always returns boolean value.

syntax: `public boolean isEnabled()`

* we need to call this isEnabled() on web element only.

ex: `driver.findElement(By.id("next")).isEnabled();`

post-validations:-

After performing operations on web elements we need to check whether they are selected or not.

* we can achieve post validation by using isSelected().

isSelected():-

* By using isSelected(), we can check whether the web element (Radio button, checkbox) is selected or not.

syntax: `public boolean isSelected();`

* If the webelement is selected, it returns true, else it returns false.
i.e isSelected() always returns boolean value.

ex: `driver.findElement(By.id("radio")).isSelected();`

Q: Difference between equals() and equalsIgnoreCase()?—

equals():—

* By using equals(), we can compare two values.
* This equals() is case sensitive
* This method returns 'true' if two values are equal, else it returns 'false'.

syntax: `public boolean equals()`

equalsIgnoreCase():—

* By using equalsIgnoreCase(), we can compare two values (string type) values.
* It ignores the case (i.e case insensitive)
* It returns true if two values are equal, else it returns false

syntax: `public boolean equalsIgnoreCase();`

Q: Difference between get() and navigate()?

we can pass URL to the browser in two ways.

1. By using get()
2. By using navigate().To()

get():—

whenever we call get() method by passing url, it will wait until the page is loaded.

This get() is available in 'WebDriver' interface.

Syntax: `public void get(String url)`

ex: driver.get("https://www.gmail.com");

navigate().To():—

Whenever we call navigate().To(), it will not wait until the page is loaded.

syntax: `public Navigation navigate().To(String url)`

ex: driver.navigate().To("https://www.google.com");

navigate():—

By using navigate(), we can perform navigation operations. like refresh, forward and backward ...etc.

```
ex: driver.navigate().forward();
    driver.navigate().back();
    driver.navigate().refresh();
```

Q:— How to get the text of WebElement?

getText():—

* we can get text of a web element by using getText();
* whenever we call getText(), it will return text in the form of a String.

syntax: `public String getText()`

* we need to call getText() on web element

ex: String electronics = driver.findElement(By.xpath("//*[text()='abc']").getText();

INDIA'S NO1 REALTIME TRAINING INSTITUTE

RAMESHSOFT

100% REALTIME TRAINING

100% PLACEMENTS

TILL NOW

300+

STUDENTS GOT PLACED

TRAINER NAME : RAMESH ANAPATI

CERTIFIED AND REAL TIME EXPERT

Opposite Of Vindu Tiffin Center Between Canara Bank And Axis Bank 3rd Floor
Near Umesh Chandra Statue SR Nagar Sarala Apartments,HYDERABAD

PH : 9177791456, 9502695908, 040-48572456

Frameworks

- Frameworks in Real Time we will discuss in detailed in classroom.

- More than 30hrs will discuss.

  Note: This notes is not complete notes

getAttribute();-

By using getAttribute(), we can get attributes of a webelement.

* whenever we pass corresponding attribute name, it is going to give corresponding value for that attribute in the form of string.

syntax: public String getAttribute(String attribute name);

ex: String username_id = driver.findElement(By.id("Email")).getAttribute("id");

String username_name = driver.findElement(By.id("Email")).getAttribute("name");

System.out.println("username id is :" + username_id);

System.out.println("username name is :" + username_name);

☆ Q:- What are web element commands?

1) click()
2) clear()
3) submit()
4) getText()
5) getAttribute()
6) sendKeys()
   - sendKeys(String testdata)
   - sendkeys('keys to send)
7) isEnabled()
8) IsDisplayed()
9) isSelected()
10) getPageSource()
11) getCurrentUrl
12) getTitle()

**Q:- Without using click() method, how can we perform clickable operations?

we can perform clickable operations in following ways.

1) submit()

2) JavascriptExecutor

3) sendkeys() method (on web element we need to call)

4) Action class sendkeys() method.

Using submit();-

ex: WebElement ele = driver.findElement(By.id("next"));

ele.submit();

note: The web element type attribute must be submit then only we can use submit().

<input id="next" type="submit" ----->

using JavascriptExecutor:-

JavaScriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("arguments[0].click();", ele);

(or)

js.executeScript("document.getElementById('next').click()");

using sendkeys():-

ele.sendkeys(Keys.ENTER);

Using Action's class sendkeys();-

Actions actions = new Actions(driver);

actions.sendkeys(Keys.TAB).build().perform();

actions.sendkeys(Keys.ENTER).build().perform();

Handling Radio buttons, check boxes and Check list

we can handle radio buttons and check boxes using clickable operations i.e using click() method.

whenever you are working with Radio buttons and check boxes we need to perform pre-validations and post validations.

we perform pre-validation using isDisplayed() and isEnabled().

post-validations by using isSelected().

Example:

```
public class RadioButtonDemo{
  public static void main(String[] args){
  WebDriver driver = new FirefoxDriver();
  driver.manage().window().maximize();
  driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);
  driver.get("https://www.facebook.com");
WebElement radio = driver.findElement(By.id("u_0_e"));
if(radio.isDisplayed() && radio.isEnabled())
  { radio.click();
  }else{
    System.out.println("unable to click on radio button");
    }
if(radio.isSelected())
{ System.out.println("Radio button is selected"); }
else { radio.click();
       system.out.println("second time performing click operation");}
driver.quit(); } }
```

How to perform keyboard operations:-

by using 'Actions' class we can perform keyboard operations.

steps to perform keyboard operations using 'Actions' class are

step(i) : Create 'Actions' class object

```
Actions actions = new Actions(driver);
```

here 'driver' is an object of WebDriver (I), and we passing as an argument.

step(ii) : perform the keyboard operations.

'Actions' class has one non-static method called sendkeys() and this method always expects keys as arguments.

Syntax: `public Actions sendkeys(keys to send);`

Program:

```
public class KeyboardDemo{
  public static void main(String[] args){
    WebDriver driver=new FirefoxDriver();
    driver.manage().window().maximize();
    driver.manage().deleteAllCookies();
    driver.manage().timeouts().implicitlyWait(56, TimeUnit.SECONDS);
    driver.get("https://www.google.com");
    driver.findElement(By.name("q")).sendkeys("selenium by Ramesh");
    driver.findElement(By.name("btnG")).click();
```

```
        Actions actions = new Actions(driver);
actions.sendkeys(keys.PAGE_DOWN).build().perform();
actions.sendkeys(keys.PAGE-DOWN, keys.PAGE-DOWN).build().perform();
actions.sendkeys(keys.PAGE-UP).build().perform();
actions.sendkeys(keys.HOME).build().perform();
actions.sendkeys(keys.END).build().perform();
    }
}
```

Note: Keys is a Enum class, by using Enum class we can define our own data types.

By class:-

"By" is an abstract class and which is available in org.openqa.selenium package. The following are the static methods present in By class.

1) public static By id(String id)
2) public static By name(String name)
3) public static By className(String classname)
4) public static By cssSelector(String cssSelector)
5) public static By linkText(String linkText)
6) public static By partialLinkText(String partialLinkText)
7) public static By tagName(String tagName)
8) public static By xpath(String xpath)

how to perform "ctrl+", "ctrl+shift" operations

In order to perform "ctrl+", "ctrl+shift" operations, keys "Enum" class has one method called 'chord()' method.

syntax: `public String chord(keys to send)`

Example:

```
public class ControlShiftOperationsDemo {
    public static void main(String[] args) {
    WebDriver driver = new FirefoxDriver();
    driver.manage().window().maximize();
    driver.manage().deleteAllCookies();
    driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);
    driver.get("https://www.flipkart.com");
    Actions actions = new Actions(driver);
    //perform 'ctrl+s' operation
    actions.sendKeys(keys.chord(Keys.CONTROL, "s").build().perform();
    //performing 'ctrl+shift+s' operation
    actions.sendKeys(Keys.chord(Keys.CONTROL, Keys.SHIFT, "s").build().per-
                                                              form();

    driver.quit();
    }
}
```

How to perform Navigation operations

By using navigate(), we can perform navigation operations. like refresh forward and backward etc.

Example:

```
public class NavigationDemo {
    public static void main (String[] args) {
    WebDriver driver= new FirefoxDriver();
    driver.manage().window().maximize();
    driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    driver.navigate().To("https://www.flipkart.com");
    driver.navigate().forward();
     driver.navigate().back();
    driver.navigate().refresh();
     driver.quit();
    }
}
```

How to perform Drop-down operations:

In order to perform Drop-down operations, Selenium provided one predefined class called 'Select' class.

* we can also perform Drop-down operations using click().

1. Using Select class:-

Select is a class from org.openqa.selenium

pseudo code for 'Select' class

```
public class Select
{
   public Select(WebElement element)
     {
     }
   public void deSelectAll()
     {
     }
   public void selectByValue(String value)
     {
     }
   public void selectByVisibleText(String text)
     {
     }
   public void selectByIndex(int index)
     {
     }
   public List<WebElement> getAllSelectedOptions()
     {
     }
```

```
public List < WebElement > getOptions ()
{
}
public void deselectByValue( String value)
{
}
public void deSelectByVisibleText( String text)
{
}
public void deSelectByIndex(int index)
{
}
}
```

In order to handle Drop-down or weblist or checklist we need to follow the following steps.

① Create Select class Object :—

Create Select class object, this object always expect webelement as an argume-nt i.e, on which web list we need to do the operations.

```
Select select = new Select(WebElement element);
```

② Call corresponding method provided by Select class:

Select class provided the below methods in order to handle web list.

i) public void selectByIndex(int index)

ii) public void selectByVisibleText(String text)

iii) public void selectByValue(String value)

Example for Drop-down:

```
public class DropDownDemo{
 public static void main(String[] args) throws InterruptedException {

  WebDriver driver=new FirefoxDriver();

 driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);

 driver.get("https://www.facebook.com");

WebElement ele_day= driver.findElement(By.id("day"));

   Select select=new Select(ele_day);

    select.selectByIndex(5);

    Thread.sleep(3000);

WebElement ele_month=driver.findElemegt(By.id("month"));

    Select select1=new Select(ele_month);

    select1.selectByValue("january");

                (or)

     select1.selectByVisibleText("jan");

WebElement ele_year=driver.findElement(By.id("year"));

     Select select2=new Select(ele_year);

          select2.selectByValue("1992");


    driver.quit(); }}
```

**

2. Using click():-

without using 'Select' class we can perform drop-down operations by using 'click()' with xpath.

ex: driver.findElement(By.xpath("//select[@id='day']/option[67]")).click();

How to perform Mouse over actions :-

In order to perform mousehover, selenium provided one class called 'Actions' class.

→ The 'Actions' class has one non-static method called moveToElement(-);

Syntax: `public Actions moveToElement(WebElement element);`

→ we can also perform mousehover actions using 'JavascriptExecutor':

Using Actions class :-

step① create 'Actions' class object

```
Actions actions = new Actions(driver);
```

* This moveToElement() always expect webelement as an argument i.e, it will ask you on which webelement you want to perform the mousehover.
* we need to pass webDriver as an argument to Actions class object.

step② :- Call the appropriate method to perform the mousehover.

* In order to perform mousehover we have moveToElement(-) in Actions class.

ex: `actions.moveToElement(driver.findElement(By.xpath("//*[text()='Electronics']"))).build().perform();`

Example:

```
public class MouseHoverDemo {
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);
```

```
driver.get("https://www.flipkart.com");
//performing mouse hover on electronics
  Actions actions = new Actions(driver);
WebElement element = driver.findElement(By.xpath("//*[text()='Electronics']
                                                 "));
actions.moveToElement(element).build().perform();
driver.quit();
  }
}
```

How to perform right click operations :-

* we can perform Right click operations by using "Actions" class.
* Actions class has one non-static method called contextClick() to perform right click operations.
* This method always ask webelement as an argument i.e, on which web element we want to perform the right click operations.

Syntax: `public Actions contextClick(WebElement element);`

```
actions.contextClick(element).sendkeys(keys.ARROW_DOWN).sendkeys(
        keys.ARROW_DOWN).sendkeys(keys.RETURN).build().perform();
```

How to perform Auto-suggestions:-

we can perform Auto-suggestion normally by using xpaths.

ex:-

```
driver.get("https://www.google.com");
driver.findElement(By.name("q")).sendKeys("selenium");
driver.findElement(By.xpath("//*[@class='sbsb-b']/li[3]/div/div[2]")
                                                    ).click();
```

How to take Screen Shot in Selenium:

* In order to take screen shot, we need to take support of 'TakeScreen-shot' interface as WebDriver doesn't have direct functionality.
* we need to convert our webdriver into TakeScreenShot(I) and then call a method called 'getScreenshotAs(-)' method.
* Always maintain your screenshots in the drive instead of eclipse
* In order to copy files or images into our local system, we have one predefined method called 'copyFile' to maintain the images in our system driver.

syntax: `copyFile(File src, File destination)`

Example:

```
public class ScreenShotDemo {
    public static void main(String[] args) {
```

```
WebDriver driver = new FirefoxDriver();
driver.manage().window().maximize();
driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);
driver.get("https://www.gmail.com");
File src = ((TakeScreenShot) driver).getScreenShotAs(OutputType.FILE);
FileUtils.copyFile(src, new File("E:\\ScreenShots\\gmail.png");
    }
}
```

How to perform or how to handle Multiple windows:

Selenium always focuses by default on current window and when we are trying to perform any operations on new window selenium will not allow you to perform the operations as it is focusing first window.

Step①: Get the current window name

In order to get the current window name, we have predefined method available in "WebDriver" called 'getWindowHandle()'.

Syntax: `public String getWindowHandle()`

whenever we call getWindowHandle(), it is always going to return current window name in the form of String.

ex: `String windowname = driver.getWindowHandle();`

Step 2 :- Get all window names.

In order to get all window names, we have one predefined method called 'getWindowHandles()' method available in WebDriver.

syntax: `public Set<String> getWindowHandles();`

whenever we call 'getWindowHandles()', it returns all the window names in the form of 'String' and stores all these window names into "Set".

ex: `Set<String> windownames = driver.getWindowHandles()`

Step③:- Iterate the set to perform the operations on windows.

Example :-

```
public class MultipleWindowsDemo {
    public static void main(String[] args) {
        WebDriver driver=new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().deleteAllCookies();
        driver.manage().timeouts().implicitlyWait(40, TimeUnit.SECONDS);
        driver.get("https://www.bing.com");
        driver.findElement(By.id("sb_form_q")).sendkeys("selenium");
        driver.findElement(By.linkText("outlook.com")).click();
```

```
String firstwindow = driver.getWindowHandle();

System.out.println("First window name is:"+firstWindow);

Set<String> windows=driver.getWindowHandles();

Iterator iterator = windows.iterator();

    while (iterator.hasNext())
    {
    String window = (String) iterator.next();
    if(!window.equalsIgnoreCase(firstWindow))
    {
    driver.switchTo().window(window);
     break;
    }

    }
driver.findElement(By.id("idA_PWD_ForgotPassword")).click();
String secondWindow = driver.getWindowHandle();
System.out.println("second window name is :"+secondWindow);
driver.quit();
    }
}
```

Frames in Selenium

If the web element is present inside the frame, selenium cannot identify that web element i.e, Selenium cannot perform any operation until we identify that frame.

* By using switchTo(), we can identify the frame.

ways to identify the frame:-

we can switch to the frames in three ways.

1) By using frame id or name:

Syntax: `driver.switchTo().frame(String frame id or name)`

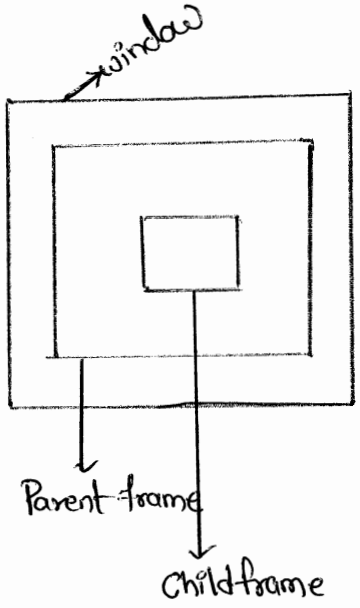
ex: <iframe id="1" name="abc" ---->

```
driver.switchTo().frame("1");
        (or)
driver.switchTo().frame("abc");
```

2) By using frame index:-

syntax: `driver.switchTo().frame(int index)`

ex: `driver.switchTo().frame(0);`



3) By using Web element :-

Syntax: `driver.switchTo().frame(WebElement element)`

ex: <iframe id='1' class=2 name=3 ---->

```
driver.switchTo().frame(driver.findElement(By.xpath("//*[@id='1']")));
```

How to switch back from frame to window:

Whenever we identify a frame, Selenium focus will be on that frame and when we try to perform operations on window, it won't allow you as it is focusing on frame.

* So if we want to perform operations on window, we need to switch back from frame to window.
* In order to switch back from frame to window, we need to call a method called 'defaultContent()'.
* whenever we call defaultContent(), selenium control or focus will move from frame to window.

Example:

```
public class FramesDemo {
  public static void main(String[] args) throws InterruptedException {
    WebDriver driver=new FirefoxDriver();
    driver.manage().window().maximize();
    driver.manage().deleteAllCookies();
    driver.manage().timeouts().implicitlyWait(56, TimeUnit.SECONDS);
    driver.get("https://www.jqueryui.com");
    driver.findElement(By.linkText("Auto complete")).click();
```

```
List < WebElement > frames = driver.findElements (By.tagname ("iframe"));
  System.out.println (frames.size());
  driver.switchTo().frame(0);
//driver.switchTo ().frame (driver.findElement (By.xpath ("//iframe[@ class =
                                                    'demo-frame']")));
  driver.findElement (By.id("tags")).sendkeys ("selenium");
  driver.switchTo().defaultContent();
  driver.findElement (By.linkText (" Date picker")).click();
   Thread.sleep (3000);
   driver.quit();
      }
  }
```

Q:- without seeing html code, identify all the frames in a page and get their Attributes.

Ans:

```
public class FramesAttributeDemo {
    public static void main(String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().deleteAllCookies();
        driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);
        driver.get("https://www.jqueryui.com");
```

```
driver.findElement(By.linkText("Auto complete")).click();
List<WebElement> list = driver.findElements(By.tagName("iframe"));
System.out.println(list.size());
Iterator iterator = list.iterator();
while(iterator.hasNext())
{
WebElement element =(WebElement) iterator.next();
String id= element.getAttribute("id");
System.out.println("Frame id is:"+id);
String class = element.getAttribute("class");
System.out.println("Frame class is:"+class);
String src = element.getAttribute("src");
System.out.println("Frame src is:"+src);
}
//driver.switchTo().frame(0); → through index
driver.switchTo().frame(driver.findElement(By.xpath("//*[@class='demo-
                                                    frame']")));  → through webelement
driver.findElement(By.id("tags")).sendKeys("selenium");
Thread.sleep(2000);
}
}
```

Note1:- RemoteWebElement is the implementation class of WebElement interface.

Syntax:- public class RemoteWebElement implements WebElement
{
=
}

Note2:- RemoteWebDriver is the implementation class of WebDriver interface.

Syntax: public class RemoteWebDriver implements WebDriver
{
=
}

How to perform 'Draggable' and 'Droppable' operations

Draggable operations:-

In order to perform Draggable operations, selenium provided one prede-fined class called 'Actions' class.

* This class has one non-static method called "dragAndDropBy()".
* This method always asks, on which web element you want to perform the draggable operations.

Syntax: `public Actions dragAndDropBy(WebElement element, int xoffset, int yoffset);`

Example:-

```
public class DraggableDemo {
  public static void main(String[] args) throws InterruptedException {

  WebDriver driver = new FirefoxDriver();
  driver.manage().window().maximize();
  driver.manage().deleteAllCookies();
  driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);
  driver.get("https://www.jqueryui.com");
  driver.findElement(By.linkText("Draggable")).click();

    driver.switchTo().frame(0);
    WebElement draggable = driver.findElement(By.id("draggable"));

      Actions actions= new Actions(driver);
```

```
actions.dragAndDropBy(draggable, 150, 150).build().perform();
    Thread.sleep(3000);
    driver.quit();
    }
}
```

Droppable operations:-

* we can perform droppable operations by using 'Actions' class.
* This 'Actions' class has one non-static method called 'dragAndDrop()'.
* This method always asks two elements as arguments i.e, on which web element you want to perform operations.

Syntax:

public `Actions dragAndDrop(WebElement source, WebElement target);`

Example:

```
public class DroppableDemo{
    public static void main(String[] args) throws InterruptedException{
        WebDriver driver = new FirefoxDriver();
        driver.manage().window().maximize();
        driver.manage().deleteAllCookies();
        driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);
        driver.get("https://www.jqueryui.com");
        driver.findElement(By.linkText("Droppable")).click();
```

```
        driver.switchTo().frame(0);

WebElement source = driver.findElement(By.id("draggable"));

WebElement target = driver.findElement(By.id("droppable"));

        Actions actions = new Actions(driver);

    actions.dragAndDrop(source, target).build().perform();

     Thread.sleep(5000);

     driver.quit();

     }
}
```

How to take ScreenShot of a WebElement :-

Program:-

```
public class WebElementScreenShot {

  WebDriver driver;

  @BeforeTest
   public void setup() throws Exception {

     driver = new FirefoxDriver();

     driver.manage().window().maximize();

     driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);

     driver.get("http://only-testing-blog.blogspot.in/2014/09/selectable.html");

       }
```

```
@Test
public void captureScreenShot() throws Exception {
//locate webelement to capture screenshot
WebElement image = driver.findElement(By.xpath("//img[@border='0']"));
//call captureElementScreenShot() to capture screenshot of element
captureElementScreenShot(image);
}
public void captureElementScreenShot(WebElement element) throws IOExce-
                                                                -ption
{
//capture entire page screenshot as buffer
//used TakeScreenShot, OutputType interface of selenium and File class of
java to capture screenshot of entire page.
File screen = ((TakesScreenShot)driver).getScreenShotAs(OutputType.FILE);
//used selenium getSize() method to get height and width of element
//Retrive width of element
int width_Image = element.getSize().getWidth();
//Retrieve height of element
int height_Image = element.getSize().getHeight();
//used selenium Point class to get x y coordinates of Image element.
//get location (x y coordinates) of the element.
Point point = element.getLocation();
```

RAMESHSOFT

SOFTWARE SOLUTIONS

OUR STUDENTS RECENTLY PLACED ON SELENIUM

AHMED

(SD SOFTECH PVT LTD)

RAFEEQ

(EDREEZ SOFTWARE PVT LTD)

MADHUPRIYA

(CIGNITI TECHNOLOGIES)

SWATHI

(COGNIZANT)

MRUDALA

(SOFTSOL TECHNOLOGIES)

GANESH

(PURPLETALK)

MANDEEP

(SOCTRONICS)

VISHWAJIT

(TCS)

SIBASYS

(COGNIZANT)

SRIKANTH.R

(INFOSYS)

SHRAVAN KUMAR

(IBM)

SANGAMESH

(TECH MAHINDRA)

PRAHALAD

(INFOBRAIN)

HARISH

(PRDC)

KIRAN

(CYBAZE)

THAKIL

(UHG)

```
int xcordinate = point.getX();

int ycoordinate = point.getY();

//Reading full image screenshot

BufferedImage img = ImageIO.read(screen);

//cut the Image height, width and x y coordinates parameters.

BufferedImage dest = img.getSubimage(xcordinate, ycoordinate, width-
                                     Image, height-Image);

ImageIO.write(dest, "png", screen);

//used FileUtils class of apache.commons.io.

//save Image screenShot In D: drive

FileUtils.copyFile(screen, new File("D:\\Screenshot.png"));

}

}
```

How to upload a File:-

We can upload a file in three ways

1. By using sendkeys()
2. By using Robot class
3. By using Autoit

By using sendkeys():-

* we can upload a file by using sendkeys()
* If you want to upload a file using sendkeys(), then the html code of that file should contain "type=file" properly.

ex: <input id="upload-file" class="file" type="file" size="1" name="upload-file" >

Example:

```
public class UploadFileDemo {
    public static void main(String[] args) {
        WebDriver driver = new FileforDriver();
        driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);
        driver.get("https://www.sendspace.com");
        driver.findElement(By.id("upload_file")).sendkeys("c:\\....");
                                                          ↓
                                                  path of file
                                                  which is to be upload
        driver.quit();
    }
}
```

2. By using Robot class :-

Example:

```
public class RobotClassDemo {

@Test
public void uploadFile()
{
WebDriver driver = new FirefoxDriver();

driver.manage().window().maximize();
driver.manage().deleteAllCookies();
driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);

driver.get("https://www.sendspace.com");

driver.findElement(By.id("upload_file")).click();

StringSelection stringSelection = new StringSelection("c:\\Users\\
                                  Desktop\\Files Demos\\flipkart.txt");

Toolkit.getDefaultToolkit().getSystemClipboard().setContents(
                                  stringSelection, null);

Robot robot = new Robot();

//press CONTROL V (ctrl+v)

robot.keyPress(keyEvent.VK_CONTROL);
robot.keyPress(keyEvent.VK_V);

//release CONTROL V (ctrl+v)

robot.keyRelease(keyEvent.VK_CONTROL);
```

```
        robot.keyRelease(KeyEvent.VK_V);

        // press Enter
        robot.keyPress(KeyEvent.VK_ENTER);

        // release Enter
        robot.keyRelease(KeyEvent.VK_ENTER);

    }
}
```

3. By using Autoit :—

Please refer window based pop up topic.

How to perform Scroll operations:

Using JavascriptExecutor we can perform Scroll operations.

Example:-

```
public class ScrollingPage
 {
   WebDriver webDriver;
   JavascriptExecutor js;

  @BeforeSuite
  public void launchingFirefoxBrowser()
   {
   webDriver = new FirefoxDriver();
   webDriver.manage().window().maximize();
   webDriver.manage().deleteAllCookies();
   webDriver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);
   }
  @Test
   public void testScript() throws InterruptedException
   {
   webDriver.get("https://www.google.com");
   webDriver.findElement(By.name("q")).sendKeys("selenium by ramesh");
   Thread.sleep(3000);
    webDriver.findElement(By.name("btnG")).click();
   webDriver.findElement(By.linkText("Selenium: selenium WebDriver By Ramesh
                                                    Anapati")).click();
```

```
//Scrolling vertical down
js = (JavaScriptExecutor) webDriver;
//js.executeScript("window.scrollBy(0,750)");
js.executeScript("scroll(0,750)");

Thread.sleep(3000);
//Scrolling vertical up
js.executeScript("scroll(850,0)");
Thread.sleep(2000);

//Scrolling Horizontal right
js.executeScript("window.scrollBy(2000,0)","");

//Scrolling Horizontal left
js.executeScript("window.scrollBy(-2000,0)","");

}
@AfterSuite
public void closeBrowser()
{
webDriver.quit();

}
```

Assert vs Verify

1) Whenever condition is fail in assert, the rest of the code is not going to be executed from that line onwards.

But in verify statements even though condition fails we can execute rest of the code.

2) we can implement assert validations by using Assert class from org.testng.Assert where as we can implement verify validations using control statements like if, if-else etc.

Assert :-

Assert is a class and which is from org.testng.Assert package.

By using assert class we can perform validations and to do this assert class is having the following methods.

1. assertEquals(String actual, String expected)
2. assertEquals(byte actual, byte expected)
3. assertEquals(int actual, int expected)
4. assertEquals(byte[] actual, byte[] expected)
5. assertEquals(char actual, char expected)
6. assertEquals(long actual, long expected)
7. assertEquals(boolean actual, boolean expected)
8. assertEquals(String actual, String expected, String message);
9. assertEquals(string actual, String expected)

10. assertEquals (byte actual , byte expected, String message)

11. assertEquals (int atual, int expected, String message)

12. assertEquals (Collection actual, Collection expected, String message)

14. Assert.assertTrue(boolean condition)

15. assertTrue (boolean condition , String message)

16. assertFalse(boolean condition)

17. assertFalse (boolean condition String message)

Sample Program :-

```
public class ValidationDemo {
    public static void main (String[] args) {
        WebDriver driver = new FirefoxDriver();
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
            driver.get ("https://www.google.co.in/");
        String text = driver.findElement (By.xpath ("//*[@id='hplogo']")).
                                                    getText();
        Assert.assertEquals ("India" , text);
        System.out.println ("web element text is:" + text);
        // Assert.assertTrue (true, text);
        // Assert.assertFalse (false, text);
        driver.quit(); } }
```

assertEquals(String actual, String expected)

This method takes two String arguments and checks whether both are equal, if not it will fail the test and an AssertionError is thrown.

Syntax: public static void assertEquals(String actual, String expected)

assertEquals(String actual, String expected, String message)

Assert that two strings are equal. If they are not, an AssertionError, with the given message, is thrown.

Syntax:- public static void assertEquals(String actual, String expected, String message)

assertTrue(boolean condition):

Asserts that a condition is true. If it isn't, an AssertionError is thrown.

syntax:- public static void assertTrue(boolean condition)

assertTrue(boolean condition, String message)

Asserts that a condition is true. If it isn't, an AssertionError with the given message, is thrown.

syntax: assertTrue(boolean condition, String message)

assertFalse(boolean condition) :-

Asserts that a condition is false. If it isn't, an AssertionError is thrown.

Syntax: public static void assertFalse(boolean condition)

assertFalse(boolean condition, String message):-

Asserts that a condition is false. If it isn't, an AssertionError, with the given message, is thrown

Syntax: public static void assertFalse(boolean condition, String message)

assertEquals(byte actual, byte expected):-

Asserts that two bytes are equal. If they are not, AssertionError is thrown.

Syntax: public static void assertEquals(byte actual, byte expected)

assertEquals(char actual, char expected, String message):-

Asserts that two chars are equal. If they are not, an AssertionFailedError, with the given message is thrown.

Syntax: public static void assertEquals(char actual, char expected, String message)

Arrays

An array is a collection of homogeneous or similar type of elements.

There are 3 types of arrays:

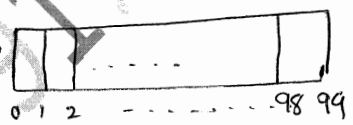
1. One - dimensional array
2. Two - dimensional array
3. Multi - dimensional array

Single or One - dimensional array :-

Declaration :

Syntax: datatype arrayname[ ] = new datatype[size];

ex: int a[ ] = new int[100];



* Array index always starts from 0.
* If we don't perform initialization of array variables, then default values are going to be assigned.

initialization :- we can perform initialization of arrays by using their index.

```
ex: int[] a = new int[20];
    a[0] = 10;
    a[1] = 20;
     :
     :
```

Example :

```
public class OneDimensionArrayDemo {
    public static void main(String[] args) {
```

```
        int[] a = new int[20];
        a[0] = 12;
        a[4] = 56;
        a[8] = 45;
        a[9] = 10;
    for(int i= 0; i<a.length ; i++)
     {
      System.out.println(a[i]);
     }
   }
}
```

Advantages of arrays :-

* we can add homogeneous or similar type of elements only.

Disadvantages of arrays :-

* size is fixed i.e based on our requirements, we cannot increase or decrease the size.
* It allows only homogeneous or similar type of elements only and when you try to add heterogeneous elements, we will get compile time error.
* To overcome these problems, we can go for 'object arrays'.

Object arrays :- In object arrays, we can add both heterogeneous as well as homogeneous elements.

Example: public class ObjectArrayDemo {
    public static void main(String[] args) {

```
Object[] obj = new Object[20];
obj[0] = 12;
obj[1] = 89.58;
obj[2] = "Rameshsoft";
obj[8] = 45.50f;
obj[9] = 10;
for (int i=0; i<obj.length; i++)
{
  System.out.println(obj[i]);
}
}
}
```

Advantages:— we can add both homogeneous as well as heterogeneous elements

disadvantages:—

* size is fixed in object array i.e we cannot increase or decrease the size based on our requirements.
* There are no pre-defined methods available

To overcome the above problems, we can go for "Collection Framework"

Collection Framework

Why Collection Framework :-

To overcome the below mentioned problems we go for Collection Framework

1. Arrays are fixed in size i.e, based on our requirements, we cannot increase or decrease the size.

In Collection framework, size is growable in nature, i.e, based on our requirements, we can increase and decrease the size.

2. In arrays, we can add only homogeneous type of elements.

In collection framework, we can add both homogeneous and heterogeneous elements.

3. In arrays, Programmer is responsible to perform addition, deletion and insertion i.e no predefined methods are available.

In collection framework, predefined methods are available.

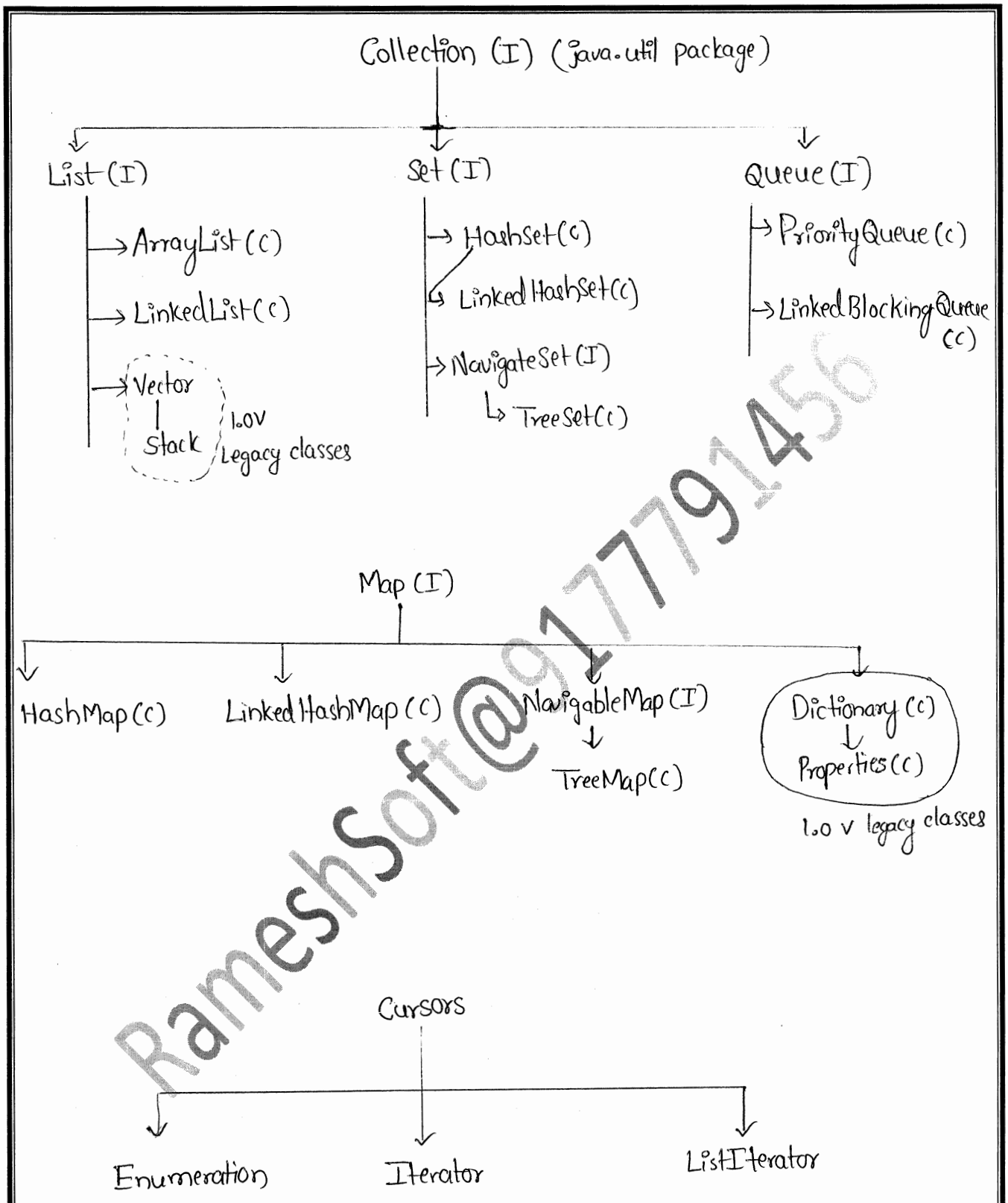
Note: Total Collection Framework is from "java.util" package.

Collection (I) :—

* Collection is an interface
* By using Collection, we can represent a group of objects as a single entity.
* In Collection interface, we can add both homogeneous as well as heterogeneous elements. i.e different types of elements.
* Collections are growable in nature
* Collection is having a bunch of predefined methods.

pseudo code :—

```
interface Collection
{ boolean add(Object o);
  boolean addAll(Collection c)
  boolean remove(Object o)
  boolean removeAll(Collection c)
  boolean contains(Object o)
  boolean containsAll(Collection c)
  boolean isEmpty()
  int size();
  Object[] toArray()
  Iterator iterator();
  boolean retainAll(Collection c)
  public void clear()
}
```



add() method :-

* By using add(), we can add one object to the collection.
* If the object added successfully, it returns true, else it returns false.

Syntax: `public boolean add(Object obj)`

addAll() :-

* By using addAll(), we can add Collection of object or multiple objects at a time to the collections.
* If the objects are add successfully, then it returns true else it returns false.

Syntax: `public boolean addAll(Collection c)`

clear() :-

By using clear() method we can clear the Collection object

Syntax: `public void clear()`

contains() :-

* By using contains() method, we can check whether element of object is present in that object or not.
* If it is there, it returns true, else it returns false.

Syntax: `public boolean contains(Object o)`

containsAll() :-

*By using containsAll(), we can check collection of objects or group of objects available in the Collection or not.

*If objects are there, it returns the true, else it returns false.

syntax: `public boolean containsAll(Collection c)`

isEmpty() :-

*By using isEmpty(), we can check whether the Collection object is empty or not.

*If it is empty, it returns true, else it returns false.

syntax: `public boolean isEmpty()`

size() :-

*If you want to find the size of collection object, then we need to call size() and it returns that size in the form of integer.

syntax: `public int size()`

remove() :-

*If you want to remove an object, then call remove()

syntax: `public boolean remove(Object o)`

iterator() :-

*If you want to iterate the objects one by one from the collection, then we can go for iterator.

Syntax: `public Iterator iterator()`

removeAll() :-

*By using removeAll(), we can remove all the objects from the collection.

Syntax: `public boolean removeAll(Collection c)`

hashcode() :-

*By using hashcode(), we can get the hashcode of the object.

Syntax: `public int hashcode()`

List(I) :-

List is an interface which is a child interface of collection interface.

→ List allows both homogeneous as well as heterogeneous elements.

→ List allows duplicate values.

→ In List, insertion order is preserved, i.e, the way you insert the elements in the same way it is going to display the elements.

→ As List allows duplicate objects or elements, we can differentiate those duplicate elements by using their indexes.

Methods available in List (I) :-

remove() :- If you want to remove an object based on index, then we can go for remove().

Syntax: `public Object remove(int index)`

This method returns the object after removing the object, i.e, it returns the removed object.

listIterator() :-

By using listIterator(), we can retrieve the elements one by one from List Collection objects.

Syntax: `public ListIterator listIterator()`

get() :-

By using get(), we can get one object from the collection

Syntax: `public Object get(int index)`

lastIndexOf() :-

If you want to find the last index of collection object then we can go for lastIndexOf().

Syntax: `public int lastIndexOf(Object o)`

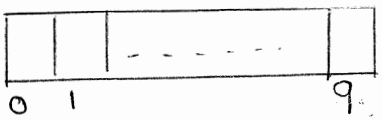
ArrayList (C):—

* ArrayList is the first implementation class of Collection interface and List interface.
* ArrayList allows both homogeneous as well as heterogeneous elements.
* ArrayList allows duplicate elements or object
* In ArrayList, insertion order is preserved i.e, the way you insert the elements, in the same way, you will get the values.
* ArrayList internally implements RandomAccess (I) and Serializable (I).
* As ArrayList implementing RandomAccess (I), we can pick the values from the Collection with the same speed.
* As ArrayList implements Serializable (I), we are maintaining objects.
* Null insertion is possible. * ArrayList implements Cloneable (I)

Constructors:—

① ArrayList arrayList = new ArrayList();

One ArrayList object is going to be created with default size '10'.



arrayList

② ArrayList arrayList = new ArrayList(Collection c);

Creates an equivalent ArrayList object for the given Collection object

This constructor meant for inter-conversion between collection objects.

Program:-

```
import java.util.*;
public class ArrayListDemo{
  public static void main(String[] args) {
     ArrayList arrayList = new ArrayList();
      arrayList.add(10);
      arrayList.add("Rameshsoft");
      arrayList.add(10.50);
      arrayList.add(45.6f);
      arrayList.add('A');
      arrayList.add("Rameshsoft");
      arrayList.add(20);
     System.out.println(arrayList);
     ArrayList arrayList1 = new ArrayList();
     arrayList1.addAll(arrayList);
     arrayList1.add(99);
     System.out.println(arrayList1);
     ArrayList arrayList2 = new ArrayList();
      arrayList2.addAll(arrayList1);
      arrayList2.add("selenium");
      System.out.println(arrayList2);
    }
}
```

Note:-

* Whenever your requirement is to maintain both homogeneous and heterogeneous elements, to allow duplicates and insertion order is preser-ved then prefer List or ArrayList.
* Whenever your requirement is retrival then go for ArrayList.

LinkedList:-

* The underlying data structure is double LinkedList
* Insertion order is preserved
* Duplicate objects are allowed
* Heterogeneous objects are allowed and homogeneous objects also.
* Null insertion is possible
* Implement Serializable and Clonable interfaces but not RandomAccess.
* Best choice if our frequent operation is insertion or deletion in the middle.

Constructors:-

① LinkedList l = new LinkedList()

Creates an empty LinkedList object

② LinkedList l = new LinkedList(Collection c);

Creates an equivalent LinkedList object for the given collection.

* Usually we can use LinkedList to implement stacks and Queues to provide support for this requirement LinkedList class defines the following 6 specific methods.

1) void addFirst(Object o)

2) void addLast(Object o)

3) Object getFirst();

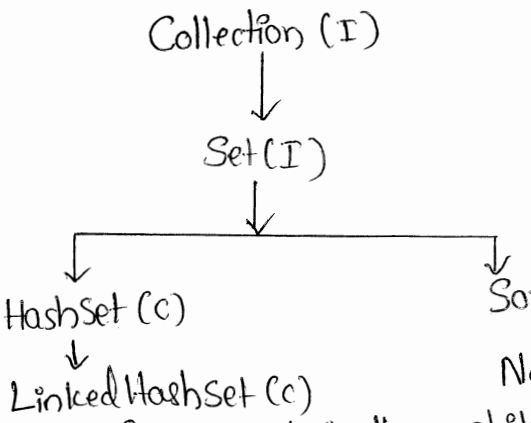
4) Object getLast()

5) Object removeFirst()

6) Object removeLast()

```
Ex:- import java.util.*;
public class LinkedListDemo{
    public static void main(String[] args)
    {
      LinkedList l=new LinkedList();
       l.add("Rameshsoft");
       l.add(30);
       l.add(null);
       l.add("Rameshsoft");
       l.add("Selenium");
       l.set(0,"Java");
        l.removeLast();
       l.addFirst("Real Time Training");
       System.out.println(l); } }
```

Set (I) :—



* Set(I) is an interface, which is the child interface of "Collection (I)".
* Set allows both homogeneous and heterogeneous elements.
* Set doesn't allow, duplicate elements and when you try to add any duplicate elements, you won't get any compile time error, simply it skips that duplicate element.

HashSet(C):—

* Hashset is the first implementation class of set and Collection interface.
* Hashset allows both homogeneous and heterogeneous elements.
* Hashset doesn't allow duplicates
* In Hashset, insertion order is not preserved and it is based on hashcode of the object.
* Hashset implements clonable and serializable but not RandomAccess
* Hashset allows "null" as well
* If our frequent operation is search operation then Hashset is the best choice.

Constructors of Hashset :-

1) HashSet hashset = new Hashset();

2) Hashset hashset = new Hashset (collection c);

Program :- import java.util.*;

```
public class HashsetDemo{
    public static void main(String[] args)
    {
     Hashset haset = new Hashset();
      hashSet.add(10);
      hashset.add("Rameshsoft");
      hashset.add("selenium");
      hashset.add(null);
      hashset.add("selenium");
      hashset.add(10);
      hashset.add('A');
      System.out.println(hashset);
    }
}
```

Output :- RameshSoft , A , 10, null , selenium

LinkedHashSet (c) :-

* It is the child class of Hashset
* Insertion order is preserved.
* It is exactly same as Hashset.

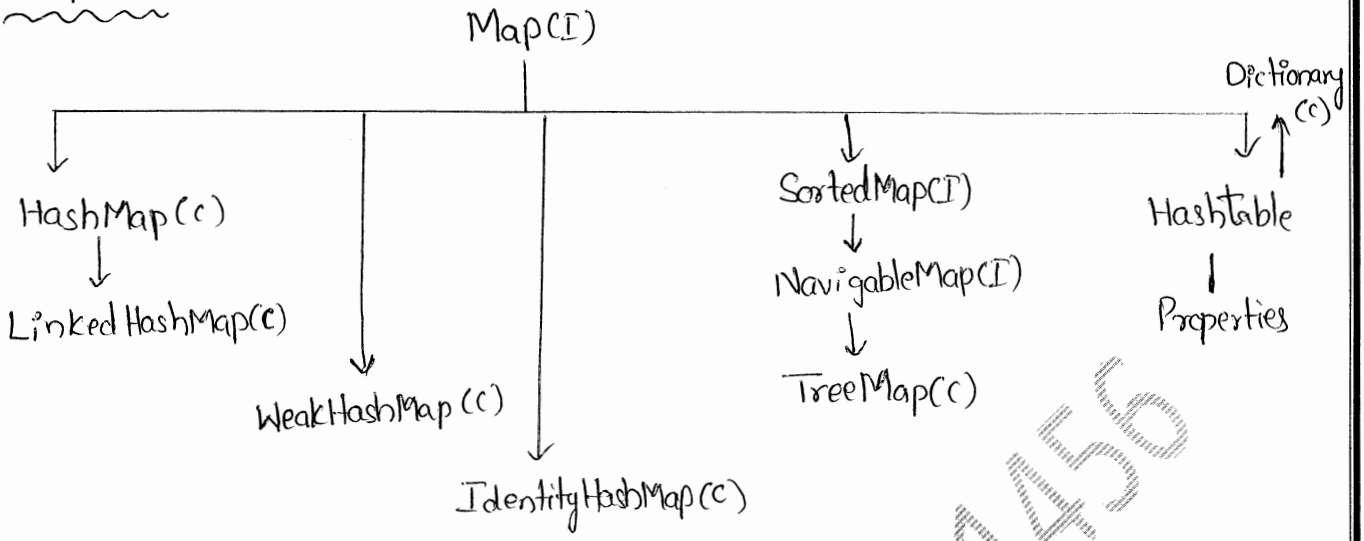
Q Remove Duplicates from ~~the following~~ an array in the following Program

```
import java.util.*;
public class ArrayDemo{
    public static void main(String[] args){
     ArrayList arrayList = new ArrayList();
     arrayList.add(10);
     arrayList.add("Rameshsoft");
     arrayList.add("Rameshsoft");
     arrayList.add(10);
      arrayList.add(10);
    System.out.println(arrayList);
    }
}
```

Solution:—

```
import java.util.*;
public class ArrayDemo{
   public static void main(String[] args){
    ArrayList arrayList = new ArrayList();
      arrayList.add(10);
      arrayList.add("Rameshsoft");
      arrayList.add("Rameshsoft");
      arrayList.add(10);
       arrayList.add(10);
   HashSet hashset = new HashSet(arrayList);
     System.out.println(hashset); } }
```

Map (I):-



* Map is an interface and it is not the child interface of Collection interface.
* Map allows both homogeneous and heterogeneous elements.
* In Map, we can store the values in the form of "key-value" pairs.
* Keys cannot be duplicates but values can be duplicate.

Methods available in Map interface:-

1. Object put (Object key, Object value);

To add one key-value pair, if the key is already available then old value will be replaced with new value and returns old value.

2. void putAll (Map m)
3. Object get (Object key)
4. Object remove (Object key)
5. boolean containsKey (Object key)
6. boolean containsValue (Object)
7. void clear().

8. int size()
9. boolean isEmpty()
10. Set keySet();
11. Collection values()
12. Set entrySet()

HashMap:—

* HashMap is the first implementation class of Map interface.
* HashMap allows both homogeneous and heterogeneous elements.
* In HashMap, values can be duplicate, but keys cannot be duplicate.
* In HashMap, insertion order is not preserved & it is based on Hash code of the object.
* Null is allowed for key (only once) and allowed for values (any no of times).

Program:—

```
import java.util.*;
public class HashMapDemo{
   public static void main(String args){
      HashMap hashmap = new HashMap();
      hashmap.put("1", "Rameshsoft");
      hashmap.put("10", "Selenium");
      hashmap.put("Hi", "120");
      hashmap.put("Hello", 130);
      hashmap.put(null, null);
      System.out.println(hashmap);
   }
}
```

Hashtable :—

* The underlying data structure for Hashtable is Hashtable.
* Duplicate keys are not allowed, but values can be duplicated.
* Insertion order is not preserved & it is based on Hashcode of the keys.
* Heterogeneous objects are allowed for both keys and values.
* Null insertion is not possible for both keys and values, otherwise we will get runtime exception saying NullPointerException.
* Every method present in the Hashtable is synchronized & hence Hashtable object is Threadsafe.

Ex:

```
import java.util.*;
class HashtableDemo
{
  public static void main(String[] args){
    Hashtable h=new Hashtable();
    h.put(new Temp(5),"A");
    h.put(new Temp(2),"B");
    h.put(new Temp(6),"6");
    h.put(new Temp(15),"D");
    h.put(new Temp(16),"E");
    h.put(new Temp(23),"F");
    //h.put("RameshSoft",null); // → RE : NPE
    System.out.println(h);
  }
}
```

```
class Temp
{
  int i;
 Temp(int i)
  {
   this.i = i;
  }
 public int hashCode()
 {
   return i;
 }
 public String toString()
 {
  return i+"";
 }
}
```

Difference b/w HashMap and Hashtable :-

HashMap	Hashtable
1. No method present inside HashMap is synchronized.	1. Every method present inside Hashtable is synchronized.
2. At a time multiple threads are allowed to operator on HashMap object simultaneously and hence it is not Thread safe	2. At a time only one thread is allowed to operate on Hashtable object and hence it is Thread safe.
3. Relatively performance is high.	3. Relatively performance is low.
4. Null is allowed for both keys and values.	4. Null is not allowed for both keys and values, otherwise we will get NullPointerException.

LinkedHashMap:-

It is exactly same as HashMap except the fllowing differences.

HashMap	LinkedHashMap
1. The underlying data structure is Hashtable	1. The underlying data structure is combin-ation of Hashtable and LinkedList.
2. Insertion order is not preserved	2. Insertion order is preserved.

Cursors:-

By using cursors, we can iterate the Values one by one from the collection Object.

These are 3 types of cursors available in Java.

1. Enumeration (I)
2. ListIterator (I)
3. Iterator (I)

Enumeration:-

→ Enumeration is an interface

→ Enumeration is a cursor and by using enumeration, we can retrieve the elements one by one from the collection object, but this enumeration is applicable only for legacy classes only.

→ Legacy classes means the classes which came in 1.0 version (Vector, Stack, Dictio-nary and Properties).

→ In real time, enumeration is not preferable as it is applicable for legacy classes.

2. Iterator

1. We can use Iterator to get objects one by one from collection.
2. We can apply Iterator concept for any collection object. Hence it is universal cursor.
3. By using Iterator object, we can able to perform both read and remove operations.

* We can create Iterator object by using iterator() method of Collection interface.

Syntax: `public Iterator iterator()`

ex: Iterator iterator = any Collection Object.iterator();

iterator() is available in Collection interface, but it is implemented in 'ArrayList' class.

methods present in Iterator interface:—

These are two methods available in Iterator interface in order to retrieve the elements.

1) hasNext()
2. next()

hasNext():— whenever we call hasNext(), it is going to check whether there is any element in that Collection object or not.

→ If the element is there, it returns true, else it returns false.

syntax: `public boolean hasNext()`

next():— By using next(), we can get object or element from the collection.

→ This method always going to return the value in the form of object.

syntax: `public Object next()`

Example:-

```
import java.util.*;
public class IteratorDemo{
   public static void main(String[] args)
    {
     ArrayList arraylist = new ArrayList();
     arraylist.add("10");
     arraylist.add "RameshSoft");
     arraylist.add("30");
     arraylist.add("selenium");
     arraylist.add("90");
     arraylist.add("A");
     array list.add("50.5f");
   System.out.println(arraylist);
  Iterator iterator = arraylist.iterator();
  while(iterator.hasNext())
  {
   String string = (string) iterator.next();
   if(String.equalsIgnoreCase("RameshSoft"))
      {
       System.out.println("Welcome to RameshSoft Software Solutions");
      }
   if(String.equalsIgnoreCase("selenium"))
      {
       System.out.println("Java with selenium in Realtime");
       }
} }
```

3> ListIterator :-

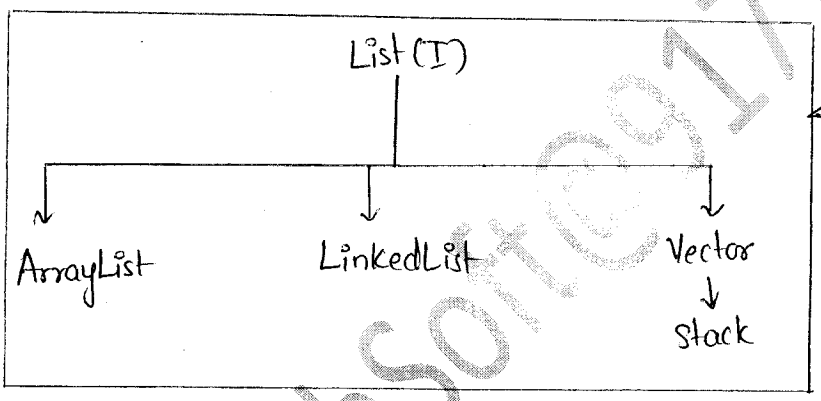
* ListIterator is a cursor and by using this cursor, we can retrieve the elements one by one from the list type of collection objects only.

* By using ListIterator we can move either to the forward direction or backward direction i.e, it is a bi-directional cursor.

* ListIterator is the child interface of Iterator.

* ListIterator is applicable only for List, ArrayList, LinkedList, Vector and stack.

* In real time, ListIterator is not preferable as it is applicable only for 'List' type of objects.



* To overcome the problems of Enumeration and ListIterator we can go for Iterator.

Generics :-

* By using generics, we can restrict the type to homogeneous.
* Once we restrict the type to homogeneous, when we try to add heterogeneous elements, we will get compile time error.

Program:-

```
import java.util.*;
public class GenericDemo{
    public static void main(String[] args)
    {
    List<String> arrayList = new ArrayList<String>();
    // arrayList.add(10); → compile time error
      arrayList.add("Rameshsoft");
      arrayList.add("Masters in Selenium");
      arrayList.add("Real time Training");
      arrayList.add("Hyderabad");
    }
}
```

Note: If we don't comment the 'arrayList.add(10);' line, we get compile time error, because we can not add integer value 10 to the List or ArrayList as they are restricted to type <String>

TreeSet(C)

→ The underlying data structure is Balanced Tree.

→ Insertion order is not preserved and it is based on some sorting order.

→ Heterogeneous are not allowed. If we are trying to insert then we will get runtime exception saying ClassCastException.

→ Duplicate objects are not allowed.

→ Null insertion is possible (only once)

→ Implements Serializable & cloneable interfaces but not RandomAccess.

Constructors:—

① TreeSet t = new TreeSet();

creates an empty TreeSet object, where the elements will be inserted according to default natural sorting order.

② TreeSet t = new TreeSet(Comparator c);

creates an empty TreeSet object, where elements will be inserted according to customized sorting order which is described by Comparator object

③ TreeSet t = new TreeSet(Collection c);

④ TreeSet t = new TreeSet(SortedSet s);

Ex:—

```
import java.util.*;
class TreeSetDemo
 {
    public static void main(String[] args)
        {
```

```
        TreeSet t = new TreeSet();
        t.add("A");
        t.add("a");
        t.add("B");
        t.add("z");
        t.add("L");
    //t.add(new Integer(10)); → RE: ClassCastException
    // t.add(null); → RE : NullPointerException
      System.out.println(t); o/p: [A, B, L, z, a]
    }
}
```

Example 2:

```
import java.util.*;
class TreeSetDemo1
{
 public static void main(String[] args)
  {
    TreeSet t = new TreeSet();
    t.add(new StringBuffer("A"));
    t.add(new StringBuffer("z"));
    t.add(new StringBuffer("L"));
    t.add(new StringBuffer("B"));
    System.out.println(t); → RE: ClassCastException
  } }
```

Null Acceptance :-

→ For empty Treeset as the first element null insertion is possible. But after inserting that null if we are trying to insert any other element we get NullPointerException.

→ For non-empty TreeSet if we are trying to insert null then we get NullPointerException.

If we are depending on default natural sorting order compulsory object should be homogeneous & comparable. otherwise we get Runtime exception saying ClassCastException

→ An object is said to be Comparable if and only if corresponding class implements Comparable interface.

→ All wrapper classes and String class implement Comparable interface, but StringBuffer class doesn't implement Comparable interface.

Hence we are getting ClassCastException in the above example.

Comparable interface :-

* Comparable interface present in java.lang package and it contains only one method i.e, `public int compareTo(Object obj)`

obj1.compareTo(obj2);

- → returns -ve, if and only if obj1 has to come before obj2.
- → returns +ve, if and only if obj1 has to come after obj2
- → returns 0, if and only if obj1 & obj2 are equal.

Ex:- Syso("A", compareTo("Z")); ⇒ o/p: -ve value

Syso("K", compareTo("A")); ⇒ o/p: +ve value

Syso("A", compareTo("A")); ⇒ o/p: 0 value

Syso("A", compareTo(null)); → RE: NullPointerException

* If we are depending on default natural sorting order internally Jvm will compare compareTo() method to place objects in proper sorting order.

Ex: TreeSet t = new TreeSet();

t.add("A");

t.add("Z"); +ve → "Z".compareTo("A");

t.add("B"); +ve → "B".compareTo("A");

-ve → "B".compareTo("Z");

t.add("A"); 0 → "A".compareTo("A");

System.out.println(t); ⇒ o/p: [A, B, Z]

Note:- If we are not satisfied with default natural sorting order or if default natural sorting order is not already available then we can define our sorting order using Comparator.

→ Comparator is meant for customized sorting order.

TreeMap (C):-

1. The underlying data structure is Red-Black Tree.
2. Duplicate keys are not allowed. But values can be duplicated.
3. Insertion order is not preserved and it is based on some sorting order of keys.

* If we are depending on Default Natural Sorting Order then keys should be homogeneous & Comparable, otherwise we will get runtime exception ClassCastException
* If we are defining our own sorting by Comparator then keys can heterogeneous and non-Comparable.
* But there are no restrictions on values, they can be heterogeneous & non Comparable.

Null Acceptance:-

1. For empty TreeMap as the first entry with null key is allowed, but after inserting that Entry if we are trying to insert any other entry we will get runtime exception saying NullPointerException.
2. For non-empty TreeMap if we are trying to insert entry with null key then we will get runtime exception saying NullPointerException.
3. There are no restrictions on null values.

Constructors:-

① TreeMap t=new TreeMap();
for default natural sorting order.

② TreeMap t = new TreeMap(Comparator c);

for customized sorting order

③ TreeMap t = new TreeMap(SortedSet m);

④ TreeMap t = new TreeMap(Map m);

Ex:-

```
import java.util.*;
class TreeMapDemo{
  public static void main(String[] args)
  {
    TreeMap t = new TreeMap();
      t.put(100, "zzz");
      t.put(103, "yyy");
      t.put(101, "xxx");
      t.put(104, 106);
      t.put(107, null);
    // t.put("FFFF", "xxxx"); => RE: ClassCastException
    // t.put(null, "xxx"); => RE: NullPointerException
    System.out.println("TreeMap object is" + t);
  }
}
```

o/p: {100 = zzz, 101 = xxx, 103 = yyy, 104 = 106, 107 = null}

Example2:—

```
import java.util.*;
class TreeMapDemo
{
  public static void main(String[] args)
   {
    TreeMap t = new TreeMap (new MyComparator());
     t.put("xxx", 10);
     t.put("AAA", 20);
     t.put("zzz", 30);
     t.put("LLL", 40);
    System.out.println (t);
     }
  }
class MyCompartor implements Comparator
{
  public int Compare(Object obj1, Object obj2)
   {
    String s1=obj1.toString();
    String s2=obj2.toString();
     return s2.compareTo(s1);
   }
}
```

O/p:- {zzz=30, xxx=10, LLL=40, AAA=20}

for-each loop:-

By using for-each loop, we can iterate the values one by one from the collection object.

Syntax: for (Object variablename : source)
{
=
}

Q write a program to iterate values from collection object.

```
import java.util.*;
public class ForEachDemo
{
    public static void main(String[] args)
    {
        ArrayList arraylist = new ArrayList();
        arraylist.add("10");
        arraylist.add("20");
        arraylist.add("30");
        arraylist.add("40");
        arraylist.add("50");
        arraylist.add("90");

        System.out.println(arraylist);

        //iterate using for-each loop
        for(Object object : arraylist)
            {
```

```
        String string = (String) object;
        System.out.println(string);
        if(string == "90").
          {
          System.out.println ("RameshSoft Software Solutions");
           }
        }
    }
}
```

Q Automate the following testCase.

1. Open the browser
2. Enter the url as www.google.com
3. Enter 'Selenium by Ramesh' in google search.
4. click on 'search' button.
5. Click on the first link (Selenium WebDriver by Ramesh Anapati)
6. check whether that page is opened or not.
7. Find the total number of links in that page and display the count on the Console
8. In that page, find the enabled links and display the count on the console.
9. In that Page, find the disabled links and display the count on the console.
10. Add Enabled and disabled links and check whether it is matching with total no: of links.

Conditions: 1) Follow the coding standards

2) Provide the validations.

```
import java.util.*;
public class LinksDemo {
    public static void main(String[] args) throws InterruptedException
    {
      int enabledLinks=0;
      int disabledLinks=0;
     WebDriver driver = new FirefoxDriver();
     driver.manage().window().maximize();
     driver.manage().deleteAllCookies();
     driver.manage().timeouts().implicitlyWait(56, TimeUnit.SECONDS);
     driver.get("https://www.google.com");
     driver.findElement(By.name("q")).sendKeys("selenium by ramesh");
    driver.findElement(By.name("btnG")).click();
    driver.findElement(By.linkText("selenium: Selenium webDriver by Ramesh")).click();
List<WebElement> links = driver.findElements(By.tagName("a"));
    int totalLinks = links.size();
  System.out.println("Total no of links are: "+totalLinks);
    Iterator iterator = links.iterator();
    while(iterator.hasNext())
      {
```

```
WebElement link=(WebElement) iterator.next();

if(link.isDisplayed() && link.isEnabled())

  {
    enabledLinks++;

  }else

   {
   disabledLinks++;
   }

System.out.println("Enabled links are :"+enabledLinks);

System.out.println("Disabled links are :"+disabledLinks);

int total=enabledLinks+disabledLinks;

if(total==totalLinks)

 {
 System.out.println("Count is matching");

 }
 else
 {
 System.out.println("Count is not matching");

 }

Thread.sleep();

driver.quit();

 }

}
```

TestNG

* TestNG is a framework
* TestNG is an open-source testing framework and 'NG' represe-nts "Next Generation".
* TestNG is more powerful than other tools like junit etc.

Advantages:

1) TestNG generates reports i.e after execution of test cases, 'TestNG' by default generates reports in a folder called "test-output" and these reports are generated in the form of html and xml.
2) By using 'TestNG' we can execute the multiple test cases at a time.
3) Based on our requirement, we can 'enable' or 'disable' the test cases.
4) As 'TestNG' has group of annotations, it makes testers life very easy.
5) we can execute the test cases based on priority by using 'TestNG'
6) we can execute the test cases based on grouping mechanism. in TestNG as well as. (e.g. SmokeSuite, Regression Suite etc).
7) TestNG supports for data parameterization for Test cases.
8) TestNG supports dependent method testing.

How to install TestNG:-

Open Eclipse → go to Help → go to Eclipse market place → type "TestNG" in search field → click on search button → we can see 'TestNG for eclipse' → click on 'install' → finish → restart.

How to check whether TestNG is installed or not?

Select the package → right click → new → other → type TestNG in search field → if "TestNG" is there, then "TestNG" is successfully installed.

Note: There are three ways to install 'TestNG' in Eclipse

1) Directly from the Eclipse Market place
2) Using the 'install new software' feature from plugin
3) Via offline jar files.

Annotations in TestNG

* @BeforeSuite :
  This annotation method will be run only once before all tests in this suite have run.

* @AfterSuite :
  This annotated method will be run only once after all the tests in this suite have run.

* @BeforeClass :

This annotated method will be run only once before the first test methods in the current class is invoked.

* @AfterClass :

This annotated method will be run only once after all the tests methods in the current class have is invoked.

* @BeforeTest :

This annotated method will be run before any test method belonging to the classes inside the <test> is run.

* @AfterTest :

This annotated method will be run after all the test methods belonging to the classes inside the <test> have run.

* @BeforeGroups :

The list of groups that this configuration method will run before. This method is guaranteed to run shortly before the first test method that belongs to any of these groups is invoked.

* @AfterGroups :

The list of groups that this configuration method will run after This method is guaranteed to run shortly after the last test method that belongs to any of these groups is invoked.

* @BeforeMethod :

This annotated method will be run before each test method.

* @AfterMethod :

This annotated method will be run after each test method.

* @Test :

Marks a class or a method as a part of the test.

* @dataProvider :

Marks a method as supplying data for a test method. The annotated method must return an Object[][].

* @Factory :

Marks a method as factory that returns objects that will be used by TestNG as Test classes. The method must return Object[].

* @parameters :

Parameterization means passing parameter or passing test data to the method. By using @parameters, we can pass the test data to the test case and we can perform cross-browser testing.

* @Optional :

If we don't define <parameter> in testng.xml file, then test method will receive the default value which is specified in @optional annotation.

Q: Write a Program for TestNG Annotations and then execution order.

```
public class TestNGAnnotationsDemo
{
  @Test
  public void testCase1()
     {
    System.out.println("This is the test case1");
     }
  @Test
  public void testCase2()
   {
   System.out.println("This is the testcase2");
   }
  @BeforeMethod
  public void beforeMethod()
   {
   System.out.println("This will execute before every method");
   }
  @AfterMethod
  public void afterMethod()
   {
   System.out.println("This will execute after every method");
   }
```

```
@BeforeClass
public void beforeClass()
{
System.out.println("This will execute before the class");
}
@AfterClass
public void afterClass()
{
System.out.println("This will execute after the class");
}
@BeforeTest
public void beforeTest()
{
System.out.println("This will execute before the test");
}
@AfterTest
public void afterTest()
{
System.out.println("This will execute after the tests");
}
@BeforeSuite
public void beforeSuite()
{
System.out.println("This will execute before the test suite");
}
```

```
@AfterSuite
 public void afterSuite()
  {
  System.out.println("This will execute after the test suite");
   }
```

Output:-

This will execute before the test Suite

This will execute before the test

This will execute before the class

This will execute before every method

This is the testcase1

This will execute after every method

This will execute before every method

This is the testcase2

This will execute after every method

This will execute after class

This will execute after test

This will execute after the Test suite.

Sequential Order of Execution of Annotations:-

@BeforeSuite

@BeforeTest

@BeforeClass

@BeforeMethod

@Test

@AfterMethod

@AfterClass

@AfterTest

@AfterSuite

How to run a Test case in TestNG :

If You want to run a TestNG Test case, select the class → run as TestNG.

* After execution of testcases by TestNG. TestNG generates reports by default and we can see these reports in "test-output" folder generated by TestNG.

* In that folder, we can see four kinds of reports.

1. default-report.html
2. index.html
3. emailable-report.html
4. testNG-failed.xml

After execution of test cases, if you want to see the reports in test-output folder, we need to refresh the project once.

How do we run multiple test cases in selenium :

In order to execute or run test cases simultaneously, we need to use "testng.xml" file.

Structure of testng.xml :-

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd" >
  <suite name="---" >
    <test name="---" >
      <classes>
        <class name="---"/>
        <class name="-----"/>
        :
      </classes>
    </test> <!-- Test --->
  </suite> <!-- suite ---->
```

Explanation :-

<suite> :-

* It is the root tag of testng.xml file
* <suite> indicates bulk of test cases
* suite contains attributes like name, parallel .. etc.>
* suite name can be anything but it should be meaningful

<test> :-

* It is the child tag of <suite> tag
* It contains test cases.

<classes> :-

* By using <classes> tag, we can configure test cases or classes inside this ".xml" file.
* In order to define individual classes, this <classes> tag contains sub class (<class>) tags.
* Each <class> tag contains name and we need to give this name in the following way.

packagename.className

ex: <class name = "com.rameshsoft.testng.TestingDemoOne" />

com.rameshsoft.testng → package name.
TestingDemoOne → className

Note: Whenever we execute test cases, it follows the execution order based on alphabetical order of the test cases if priority is not mentioned.

Example : Write two classes using TestNG annotations and run the two classes in "xml" file.

Program1: TestNGDemoOne.java

```
package com.rameshSoft.testng;

public class TestNGDemoOne
{
WebDriver driver;

@BeforeSuite
public void browserOpen()
{
driver = new FirefoxDriver();

driver.manage().window().maximize();

driver.manage().deleteAllCookies();

driver.manage().timeouts().implicitlyWait(50, TimeUnit.SECONDS);

  }

@AfterSuite
public void closeBrowser()
  {
   driver.quit();
  }

@Test
public void main()
 {
driver.get("https://www.flipkart.com");
System.out.println("Flipkart");
} }
```

Program 2: TestNGDemoTwo.java

```
package com.rameshsoft.testng;

public class TestNGDemoTwo
 {
 WebDriver driver;

@BeforeSuite
public void browserOpen()
 {
 driver = new FirefoxDriver();

 driver.manage().window().maximize();

 driver.manage().deleteAllCookies();

 driver.manage().timeouts().implicitlyWait(45, TimeUnit.SECONDS);

   }
 @AfterSuite
  public void closeBrowser()
   {
   driver.quit();
   }

@Test
 public void amain()
  {
  driver.get("https://www.facebook.com");

   System.out.println("facebook");

   }
```

```
@Test
public void gmain()
{
driver.get("https://www.gmail.com");
System.out.println("gmail");
}
@Test
public void bmain()
{
driver.get("https://www.google.com");
System.out.println("google");
}
}
```

testng.xml file:-

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE suite SYSTEM "https://testng.org/testng-1.0.dtd">
<suite name="Suite">
<test name="Test">
<classes>
<class name="com.rameshsoft.testng.TestNGDemoOne"/>
<class name="com.rameshsoft.testng.TestNGDemoTwo"/>
</classes>
</test>
</suite>
```

How to run xml suite :-

select the testng.xml file → right click → run as → TestNG suite → click.

Q: How to create basic testNg.xml?

Select the package which test cases you want to execute → right click → TestNG → Convert to TestNG → finish

Attributes in TestNG

priority attribute :-

If you don't want to follow the alphabatical order execution and we want our own execution order, then we have one attribute in TestNG called "priority". This 'priority' attribute can be specified at @Test annotation level.

* priority always starts from 0, 1, 2, ... etc (i.e High → low
  0 - means High priority, and execates first)

ex:
```
@Test(priority = 2)
public void main()
{
driver.get("https://www.gmail.com");
System.out.println("Gmail");
}
```

enabled attribute :-

* If we don't want to execute a particular test case, then we have one attribute in testng called "enabled".

* This 'enabled' attribute always takes "boolean" as value.

i.e if 'enabled = true' then test case is going to be executed and if 'enabled = false', then test case is not going to be execut-ed because it is disabled.

* we need to specify "enabled" attributed at @Test level.

* 'enabled= true" is the default attribute for all test cases until and unless we specify enabled = false

Example :

```
public class TestNGDemo
{
  WebDriver driver;
  @BeforeSuite
   public void browserOpen()
   {
   driver = new FirefoxDriver();
   driver.manage().window().maximize();
   driver.manage().timeouts().implicitlyWait(40, TimeUnit.SECONDS);
   }
```

```
@AfterSuite
public void closeBrowser()
 {
 driver.quit();
 }
@Test(priority=2, enabled=true)
 public void amain()
 {
 driver.get("https://www.facebook.com");
 System.out.println("Facebook");
 }
@Test(priority=1, enabled=false)
 public void gmain()
 {
 driver.get("https://www.google.com");
 }
@Test(priority=0)
 public void bmain()
 {
 driver.get("https://www.flipkart.com");
 System.out.println("Flipkart.com");
 }
```

Output:-

Flipkart.com

Facebook

Execution Order: BeforeSuite

bmain()

amain()

AfterSuite

* gmain() is not going to executed as it is disabled using enabled = false.

* In the above program bmain() has two attributes, one is 'priority = 0' and other is default attribute of 'enabled = true'.

dependsOn attribute :-

Sometime the test cases may depend on other test cases. In order to achieve this depends on mechanism, TestNG provided one functionality called 'dependsOn' functionality.

we can specify 'dependsOn' attribute at two levels:

1. 'dependsOn' at test case level
2. 'dependsOn' at grouping level.

Example for dependsOnMethods:

```
public class DependsOnMethodsDemo
{
```

```
@BeforeSuite
public void dmain()
 {
  public void System.out.println("dmain");
 }
@AfterSuite
 public void cmain()
 {
 System.out.println("cmain");
 }
 @Test
 public void bmain()
 {
 System.out.println("bmain");
 }
@Test(dependsOnMethods="bmain")
  public void amain()
  {
  System.out.println("amain");
  }
@Test
public void gmain()
 {
 System.out.println("gmain");
 }
}
```

Example for dependsOnGroups :

```
public class DependsOnGroupsDemo{
  @Test(groups = {"group1"})
   public void testPrintWelcomeMessage()
    {
     System.out.println("welcome to rameshsoft");
    }
  @Test(dependsOnGroups = {"group1.*"})
   public void testMessage()
    {
     System.out.println("Masters in java with selenium");
    }
  @Test(groups = {"group1"})
   public void testLocation()
    {
     System.out.println("At Hyderabad");
    }
}
```

output : At Hyderabad

Welcome to RameshSoft

Masters in java with selenium.