Java-Selenium-Maven-Frameworks

About Java ?

Java is a simple programming language
Writing, compilation and debugging a program is very easy in java
It helps to create reusable code

Features of Java ?

Java has more features,
1. Platform independent -
Using byte code we can run the application to any platform such as windows, mac, linux, etc
2. Open source - program in which source code is available to the general public for use and/or modification
from its original design at free of cost is called open source
3. Multithreading - Java supports multithreading
It enables a program to perform several task simultaneously
4. More secure - It provides the virtual firewall between the application and the computer
So it's doesn't grant unauthorized access
5. Portable - "Write once run anywhere"
Java code written in one machine can run on another machine

About JDK / JRE / JVM?

JDK: JRE + Dev tools.
Java Development Kit
If run any applications we need JDK have to installed
JDK versions: 1.0 to 1.9
Mostly V1.8 is used now
JRE: JVM + lib files.
Java Runtime Environment
It is a pre-defined. class files (i.e.) library files
JVM:
Java Virtual Machine
It is mainly used to allocate the memory and compiling
JIT- Just in time convert into machine code.

About Heap Memory ?

Heap Memory:
Object are stored in heap memory
RAM---- JVM ------Heap memory

About Garbage Collection?

Garbage collection:
It makes java **memory efficient** because garbage collector removes the unreferenced objects from heap memory

About OOPS Concept?

OOPS CONCEPT:
Object Oriented Programming Structure
OOPS is a method of implementation in which programs are organized as collection of objects, class
and methods

Oops principles are
1. Class
2. Method
3. Object
4. Abstraction
5. Encapsulation
6. Inheritance
7. Polymorphism

About Class / method / object?

CLASS:
- Class is nothing but collection of methods or collection of objects.
- A **class** can be defined as a template/blueprint that describes the behavior/state that the **object** of its type support.

METHOD:
- A set of action to be performed

OBJECT:
- Run time memory allocation
- Using object we can call the any methods

- Example Program:

```
public class FirstProgram {

    public void javaTest() {
        System.out.println("JAVA  FIRST PROGRAM");
    }

    public static void main(String[] args) {
        FirstProgram myobj = new FirstProgram();
        myobj.javaTest();
    }
}
```

About Encapsulation ?

ENCAPSULATION
Structure of creating folder.. If you are creating class you are doing encapsulation

About Inheritance and its type ?

INHERITANCE:

To reduce object memory we go for inheritance

We can access one class property into another class using 'extend' keyword and reusable

Purpose. And to reduce object memory

Ddefining a new class based on existing class by extending its props.

Child class Sub class derived class – class that extends another class

Parent class Super class base class – whose class are used or inherited by another class

Types:

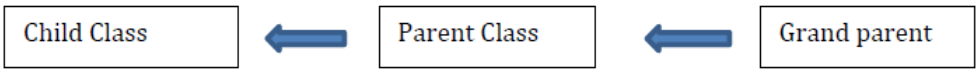
1. Single Inheritance
2. Multilevel Inheritance
3. Multiple Inheritances
4. Hybrid Inheritance
5. Hierarchical Inheritance

1. Single Inheritance :

One parent class is directly support into one child class using extend keyword

2. Multilevel Inheritance:

One child class and more than one parent class



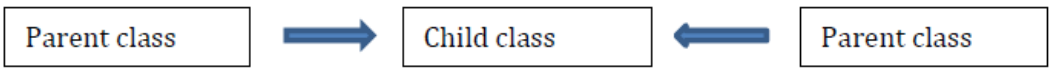
3. Multiple Inheritance:

More than one parent class parallely support into one child class but it won't suport in java

because

1. Priority problem
2. Compilation error/syntax error

(i.e) if both parent class having same method name it will get priority problem so it doesn;t work in java



4.Hybrid Inheritance:

It is a combination of single and multiple inheritance

5. Hierarchical Inheritance:

One parent class and more than one child class

About Access specifier ?

ACCESS SPECIFIER:

1. Public
2. Protected
3. Default
4. Private

1. Public:

It is global level access(same package + different package)

2. Private:

It is a class level access

3. Default:

Package level access

Without access specifier within the package we can access

4. Protected:

Inside package + outside Package (Extends)

About Data types?

Data Types	Size	Wrapper Class	Default Value
Byte	1	Byte	0
Short	2	Short	0
Int	4	Integer	0
Long	8	Long	0
Float	4	Float	0.0
Double	8	Double	0.0
Boolean	-	Boolean	False
Char	-	Character	-
String	-	String	null

For byte,

1 byte =8 bits

To find range: formula

-2^n-1 to +2^n-1 - 1

About wrapper class ?

WRAPPER CLASS:
Classes of data types is called wrapper class
It is used to convert any data type into object
All classes and wrapper classes default value is Null

About user input ? / scanner class

```
Scanner sc=new Scanner(System.in);
System.out.println("PLs enter value");
int i=sc.nextInt();
 byte nextByte();
 short nextShort();
 int nextInt();
 long nextLong();
 float nextFloat();
 double nextDouble();
 char next().charAt(0);
 String next();
 String nextLine();
 boolean nextBoolean();
```

About = and ==?

Difference between "=" and "=="
= is used to assigning the value
== is used for condition checking

About logical and bitwise?

1. logical &&,// logical && check first condition if its fail it doesn't check second
2.Bitwise &,/bitwise & is check both condition
So logical && is better than bitwise

About Null?

Null:
Null is a undefined/unknown/unassigned value
Null is won't create any memory

About Polymorphism ?

POLYMORPHISM:
Poly-many
Morphism-forms
Taking more than one forms is called polymorphism or one task completed by many ways
It has 2 types,
1.Method overloading(static binding/compile time polymorphism)
2.Method overriding(dynamic binding/run time polymorphism)

About Method Overloading ?

1.Method overloading:
Class-same
Method-same
Argument-differ
In a same class method name is same and the argument is different is called method overloading
the argument is depends on
data types
data types count
data type order

```
public class StudentInfo {
	private void studentId(int num) {
	}
	private void studentId(String name) {    \\ depends on order
	}
	private void studentId(String email, int ph) {  \\depends on data type
	}
	private void studentId(int dob, String add) {  \\depends on datatype count
	}
	public static void main(String[] arg) {
		StudentInfo info = new StudentInfo();
	}
}
```

About Method overridding ?

2.Method overriding:
Class name-differ(using extends)
Method-same
Argument- same
In a different class , the method name should be same and argument name should be same is
called overriding

We can assign our sub class to our super class but can't reverse
Marriage b=new Boy() is possible
Boy b=new Marriage() impossible

```
public class Boy extends Marriage {
	public void girlName() {
		System.out.println("ramya");
	}
	public static void main(String[] args) {
	Boy.b=new Boy();
	b.girlName();
	}
```

- 2nd class(super class)

```
public class  Marriage {
	public void girlName() {
		System.out.println("priya");
	}
```

About abstraction ?

ABSTRACTION:

Hiding the implementation part is called abstraction
it has 2 types,
1.Partially abstraction(abstract class)
2.Fully abstraction(interface)

About Abstract class?

1.Partially Abstraction(Abstract class):

It will support abstract method and non-abstract method.
We can't create object for abstract class because in the method signature we didn't mention any
business logic
It have 2 class, abstract class(sub class) and super class. we create object and business logic only
in super class, won't create in abstract class

```
// abstract class

public abstract class Abstraction {

    abstract void test1();

    public void test4() {
        System.out.println("goodiee");
    }
}
// super class
class abstract1 extends Abstraction {

    @Override
    void test1() {
        System.out.println("test1");
    }
    public static void main(String[] args) {
        abstract1 obj = new abstract1();
        obj.test1();
        obj.test4();
    }

}
```

About Interface?

2. Interface/Fully Abstraction;
It will support only abstract method, won't support non abstract method
In interface "public abstract" is default. we no need to mention
It using implements keywords
multiple inheritance its won't support in java but using interface its support here we have to create 2 interface(super class) and one sub class(normal).
In the sub class we implement both interface

```
public interface Sample {
    void test5();
    void test6();
}

class inter implements Sample {
    @Override
    public void test5() {
        System.out.println("testing2");
    }

    @Override
    public void test6() {
        System.out.println("testing3");
    }

    public static void main(String[] args) {
        inter obj = new inter();
        obj.test5();
        obj.test6();
    }

}
```

About arrays?

ARRAYS:

Collection of similar data
The value are stored based on index
The index will start 0 to n1

Syntax:
int num[]=new num[5]
Here,
int data type
num variable
[] Array
5 Array len
If we didn't assign any value, it will takes the default value of data types.

About adv and disadv of arrays?

Advantage of array:
In a single variable we can store multiple values

Disadvantage of arrays:
It support only similar data types
It is a fixed size
Memory wastage is high
To overcome these we go for collections

About upcasting / downcasting ?

Upcasting	Downcasting
casting to a supertype	casting to a subtype
happens automatically	is not directly possible in Java, explicitly we have to do.

About String?

STRING:
Collections of character or word enclosed with double quotes

About types of string method ?

String Functions	Descriptions	Return type
charAt()	to print the particular character	Char
Equals()	used to check our string index is true or false	Boolean
Contains()	used to check the particular character or word in the string	Boolean
Equalsignorecase()	It is like a equals() method but it is not case sensitive	Boolean
split()	used to split the string by space or character or word or whatever	String
subString()	to print from, which character we want in the string index	String
toUpperCase()	to convert the string into uppercase	String
toLowerCase()	to convert the string into lowercase	String
Replace()	to replace the index character or word	String
indexOf()	to print the position of the character in the string. if the character is not available , it will print "-1“ if multiple same character is have, it takes first one position	Int
lastIndexof()	If multiple same character , it takes last one	Int
Trim()	To remove white space in string	String
isEmpty()	to check the index length is zero or not, If its zero, its true otherwise false	Boolean

String Functions	Descriptions	Return Type
Compareto()	Compare based on ASCII Values	int

Every character have one ASCII value
A-Z 65 to 90
A-z 97 to 122
0-9 48 to 57
remaining special characters

```
char ch='M';
int x=ch;
System.out.println(x);
```

About literal and non-literal string ?

Literal String

It's stored inside the heap memory (string pool or string constant).
It will share the memory if same value (duplicate value)

Non-literal string

Its stored in the heap memory.
Its create a new memory every time even if its duplicate value(same value)

About string manipulation ?

String manipulation:
The action of the fundamental operations on **strings**, including their creation, concatenation, the extraction of **string** segments, **string** matching, their comparison, discovering their length, replacing substrings by other **strings**, storage, and input/output. "**string manipulation**."

About mutable and immutable string ?

Immutable string

- We can store more duplicate value in same memory
- We can't change the value in memory
- In concord nation, it's have to create new memory
- Supports both literal string and non-literal string

Mutable string

- we can't store duplicate value in same memory
- we can change the value in memory
- In concord nation, its takes same memory
- Supports only non literal string

About String buffer and string builder ?

String Buffer	Synchronized
	Thread safe
String Builder	Asynchronized
	Not Thread Safe

String is a Immutable object

identityHashcode() is used to print the reference value(storage reference)

About Collections and its type ?

COLLECTIONS

It will support dissimilar data types.

It is dynamic memory allocation

No memory wastage like array

Overcome disadvantage of arrays

- List
  - -Interface
  - - Displays on Insertion order
  - -Index based
  - - Allows Duplicate
- Set
  - -Interface
  - - Not maintain any order to display
  - -Value based
  - - Ignore Allows Duplicate
- Map
  - -Interface
  - - Key and value pair is one entry
  - -Key ignore the duplicate value and value allow the duplicate

List<Integer> ex=new Array List<Integer>();
List – Interface
< > - Generics (mention wrapper class)
Array List - Class

About array list / linked list / vector list ?

Array List

- Best Case- retrieve/searching .i.e have 100 index is there, if we going to print 60th value, we can easily search
- Worst Case- Deletion and insertion is a worst one because if we delete/insert one index value after all the index move to forward/backward. It makes performance issue

Linked List

- Best Case- Insertion / deletion because all values based on the separate nodes. (i.e) if we delete one value, the next node will join to the previous one
- Searching/retrieving (ie), if we have 100 nodes, we have to print 90th node value, it will pass to all the previous nodes and comes to first and then it will print. It's makes performance issue

Array List	Vector List
☐ Asynchronized	☐ Synchronize
☐ Not Thread safe	☐ Thread safe

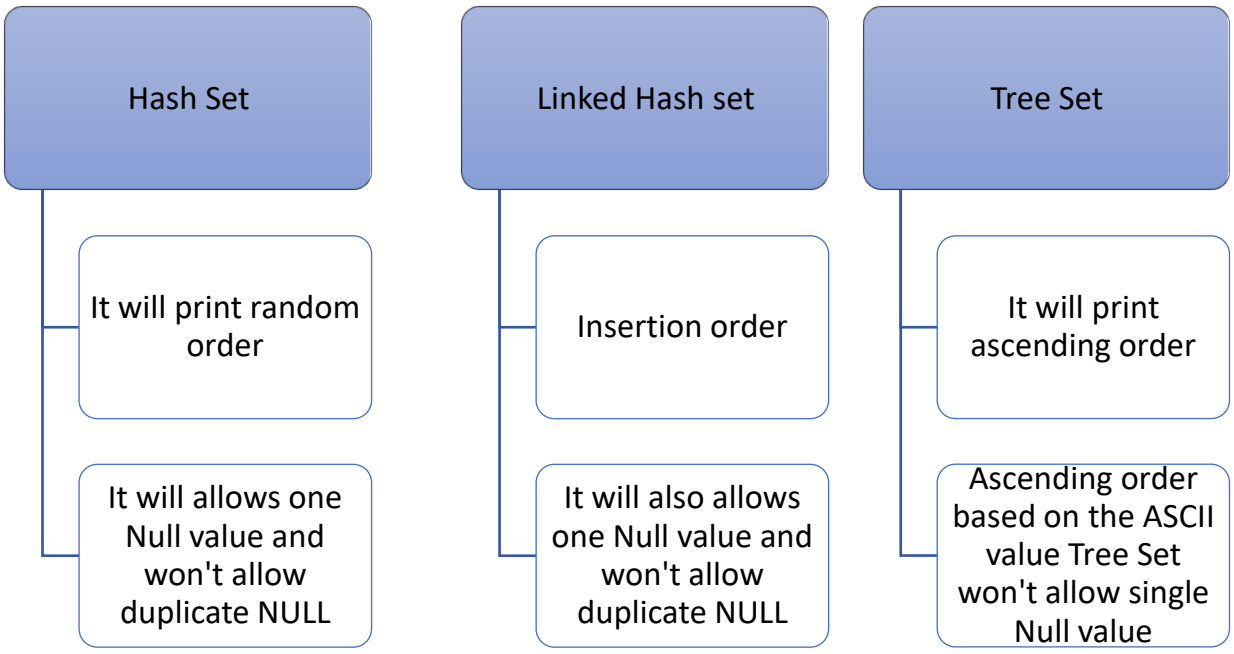
About Generics ?

Generics
It will support particular datatypes or object only
It is a features of jdk 1.5
In the generics, we can mention only wrapper class
< >- This is generic symbol, is used to define the particular datatype
If we need integer datatype,

About types of methods in list ?

Methods	Descriptions
add()	to insert a value
size()	to find the size
get()	used to print the particular value
Remove()	to remove the particular index value
add(1, 10)	Index based add used to add the value based on the index
set()	is used to replace the value
contains()	to check the particular value or object
clear()	used to clear the all index value
Lastindexof()	print the position from the last
indexof()	used to print the position of the list
addAll()	used to copy from one list to another list
removeAll()	used to compare the both list and remove all the list1 values in list
retainAll()	is used to compare both list and print the common values

About diff types of Set?



Normal for loop is not work here because it is not index based, it is value based

About diff types of map ?

Hash Map	Linked Hash Map	Tree Map
• Random Order • If duplicate key is there, it takes the last one • Key will allows the only one Null • Value allow the duplicate null	• Insertion Order • Key will allows the only one Null • Value allow the duplicate null	• Ascending Order • Key won't allow Null(even single null) • Value allow the duplicate null

Hashtable	**Concurrent hashmap**	**Hashmap**
• Random order • Both key and values are ignore the Null • Synchronize(thread safe)	• Random order • Both key and values are ignore the Null	• Random order • Asynchronies(not thread safe)

About diff types of method in map?

Method	Description
get()	used to print the value based on key
keyset()	it is used to seperate the key
Value()	it is used to seperate the value
entryset()	It is used to iterate the map

Example of Map ?

```
class Map {
public static void main(String[] args) {
LinkedHashMap<Integer,String> ex =new LinkedHashMap<Integer, String>();
ex.put(10, "test1");
ex.put(11, "test2");
ex.put(12, "test3");
ex.put(null, null);
ex.put(23, null);
System.out.println(ex);
System.out.println(ex.get(11));
System.out.println(ex.keySet());
System.out.println(ex.values());
Set<Entry<Integer, String>> ey = ex.entrySet();
for (Entry<Integer, String> entry : ey) {
System.out.println(entry.getValue() + "=" + entry.getKey());
}}}
```

About private class ?

private classes are allowed but only as inner or nested classes. If you have a private inner or nested class, then access is restricted to the scope of that outer class.
If you have a private class on its own as a top-level class, then you can't get access to it from anywhere. So it does not make sense to have top level private class.

About iterator and list iterator and its methos?

Iterator and List Iterator:
Both of these interfaces are used for traversing.

Iterator:

- Iterator is used for traversing List and Set both.
- We can traverse in only forward direction using Iterator.
- We cannot add element or replace to collections

Methods of Iterator:

- hasNext()
- next()
- remove()

ListIterator:

- We can use ListIterator to traverse List only, we cannot traverse Set using ListIterator.
- we can traverse a List in both the directions (forward and Backward).
- We can add element at any point of time & By using set(E e) method of ListIterator we can replace the last element returned by next() or previous() methods.

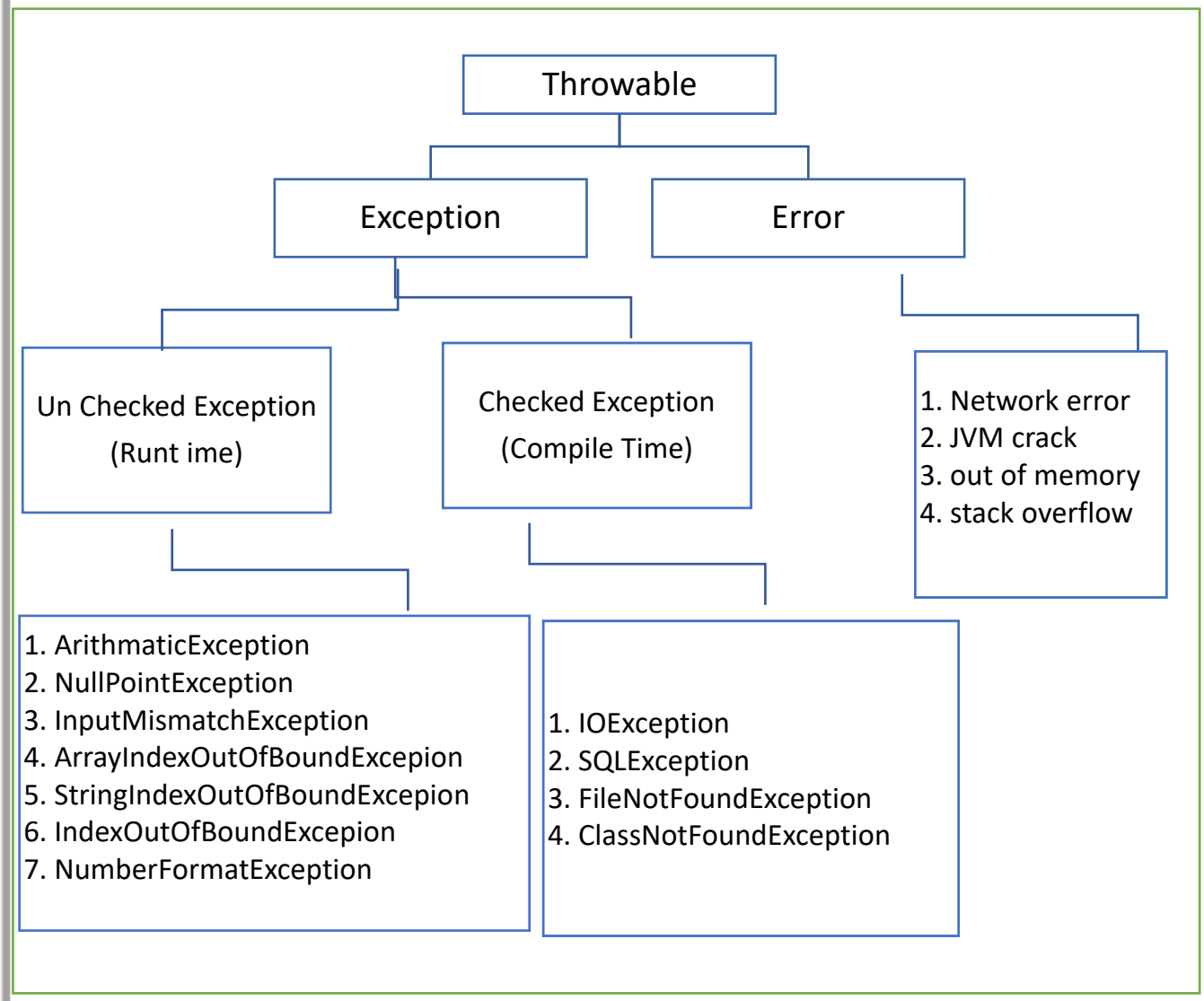
Methods of ListIterator:

- add(E e)
- hasNext()
- hasPrevious()
- next()
- nextIndex()
- previous()
- previousIndex()
- remove()
- set(E e)

About sys.out,println?

- Inside the System class is PrintStream object called out which is static (that's why you can call it using the class name System).
- Now this PrintStream has a methods called print and println which are void. They return nothing.
- System.out.**println** is not used to control how your program executes. It is a merely way to inform the user of what is going on inside your program.

About exceptions and its flow ?



About exceptions ?

Exception is like a error, the program will terminated that line itself
Super Class of Throwable – Object
Super class of all Exceptions – Throwable
Super class of Checked/Unchecked or Runtime/Compile time Exception – Exceptions

About exception handling methods ?

Exception Handling:
1. Try
2. Catch
3. Finally
4. Throw
5. Throws

About Try/catch/finally ?

Try and catch:
If we get exception, try will throw the exception and catch will catch the exception
Finally:

- finally will execute always whether the exception through or not. We can give the combination like try---- catch--- finally, we can't reverse/interchange.
- If we give try --- finally, again it will show the exception
- In one try block, we can write only one finally block

About throw and throws?

Throw and Throws:
Throw:

- Throw is a keyword, we can through any exception inside the method
- At a time we can throw only one exception
- It throws the exception

Throws:

- Throws is a keyword, it is used to declare the exception(in method level)
- At a time we can declare more than one exception
- It defines the exceptions

Key points in exception handling :

- In a try block, one catch we can use same excepion and another catch we use throwable exception At this time, it will through the first one if it is match, will print. if it is not correct will throw the second.
- In more than one catch block, we can use like sub class and super class combination. But we can’t use reverse
- In between try ,catch and finally, we won't write any logics
- In one try block we can use n-number of catch blocks but we can't repeat the same exception
- In one try block we can handle only one exception

Inner try:

- If we use inner try, it will print inner catch, inner finally and outer finally. But one try block handle one exception only, even if we use inner try also.
- If main try have no exception, it will through inner try. in that inner try if catch exception is wrong, it will goes and print outer finally.

About custom exceptions?

custom exception:
If you are creating your own **Exception** that is known as **custom exception** or user-defined **exception**. Java **custom exceptions** are used to customize the **exception** according to user need. By the help of **custom exception**, you can have your own **exception** and message

```
package in.oops.we;

public class EmployeeNotFound extends Exception {
    public EmployeeNotFound() {
        System.out.println("Exeception");
        getMessage();
    }
}
```

```
package in.oops.we;

public class UserDefinedException {
     public static void main(String args[]){
         try{
               throw new EmployeeNotFound();
          }
         catch(EmployeeNotFound e){
            System.out.println(e) ;
         }
        }
```

About null point exception?

Null Point Exception:
Local variables, the properties of a class, method parameters, and method return values can all contain a **null reference**. Calling methods or properties of these variables when they are **null** causes a **NullReferenceException**.

How to avoid the NullPointerException
In order to avoid the NullPointerException, ensure that all your objects are initialized properly, before you use them. Notice that, when you declare a reference variable, you are really creating a pointer to an object. You must verify that the pointer is not null, before you request the method or a field from the object.
we shouldn't catch that exception. we should just fix it

About constructors? And its type

Constructor:

- Class name and constructor name must be same.
- It doesn't have any return type.
- We don't want to call constructor which is creating object itself.
- It will automatically invoke the default constructor.
- It will support in method overloading but won't support in method overriding

public Const1() →Non- Argument based constructor
public Const1(int a) → Argument based constructor
In argument base constructor we have to pass the argument in object
Const1 c1=new Const1(10);
In non-argument base constructor we don't want to pass any argument
Const1 c=new Const1();

About this and super ?

This():

- It will refer the class level variable value
- It is a keyword.
- If we use 'this' argument , we can pass without object
- This argument we must use only in first statement
- If same variable name in local level and class level, it will give first preference is local level

Super :

- It is a keyword.
- It will refer the parent class level variable value

why java main method is static?
because object is not required to call static method if it were non-static method, jvm create object first then call main() method that will lead the problem of extra memory allocation.
Can we execute a program without main() method?
Yes, one of the way is static block Except jdk 1.7

About final and static ?

Final	Static
• It's a keyword. It is like constant. • If we use final before the variable, we can't overwrite. If we trying to overwrite it show compile time error. • As well as if we use final before the method it cant be override • If we use in class, we can't extend /inherited.	• The static variable can be used to refer the common property of all objects • If we use static in method, we don't want to create object. • static variable will get the memory only once, if any object changes the value of the static variable, it will retain its value • Java static property is shared to all objects.

About types of variables ?

Local variables	Instance variables-	Class/Static variables
declared in methods, constructors, or blocks.	declared in a class, but outside a method, constructor or any block.	declared with the static keyword in a class, but outside a method, constructor or a block.
Access modifiers cannot be used for local variables	Access modifiers can be given for instance variables. Instance variables have default values	Class variables also known as static variables.
There is no default value for local variables, so local variables should be declared and an initial value should be assigned before the first use.	Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.	Static variables can be accessed by calling with the class name ClassName.Variable Name.

About final and finally ?

Finally:

- finally will execute always whether the exception through or not. We can give the combination like try---- catch--- finally, we can't reverse/interchange.

Final :
It's a keyword. It is like constant.
If we use final before the variable, we can't overwrite. If we trying to overwrite it show compile time error.

About JDBC Connections and its steps

JAVA JDBC-

Java JDBC is a java API to connect and execute query with the database. JDBC API uses jdbc drivers to connect with the database.

Why use JDBC
Before JDBC, ODBC API was the database API to connect and execute query with the database. But, ODBC API uses ODBC driver which is written in C language (i.e. platform dependent and unsecured). That is why Java has defined its own API (JDBC API) that uses JDBC drivers (written in Java language).

API (Application programming interface) is a document that contains description of all the features of a product or software. It represents classes and interfaces that software programs can follow to communicate with each other. An API can be created for applications, libraries, operating systems, etc

JDBC Driver is a software component that enables java application to interact with the database.There are 4 types of JDBC drivers:JDBC-ODBC bridge driver
Native-API driver (partially java driver)
Network Protocol driver (fully java driver)
Thin driver (fully java driver)

7 Steps for JDBC Connections:
1.Load the driver
2.Connect to the Database.
3.Write the Sql query
4.Prepare statement
5.Execute the query
6.Get the data result.
7.Close the DB Connections

```java
public class Jdbc {
    public static void main(String[] args) throws Throwable {
        Connection con = null;
        try {
            //load the driver
            Class.forName("com.mysql.jdbc.Driver");
            //connect to database
            con = DriverManager.getConnection("jdbc:mysql://localhost:3306/jdbc_connection","root", "");
            //write sql qurey
            String sql = "SELECT * FROM jdbc_connection.prem";
            // prapare statement
            Statement ps = con.prepareStatement(sql);
            //execute query
            java.sql.ResultSet re = ps.executeQuery("SELECT * FROM jdbc_connection.prem");
            //get the data result
            while (re.next()) {
                System.out.println(re.getInt(1) + re.getString(2) + re.getString(3));
            }

        } catch (Exception e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        //clode db con
        con.close();

    }

}
```

Constraints:

Enforced rule on atable.
Restriction.

Constraints	Description
Primary Key	Entity Integrity- Ignore null values and duplicate value
Unique	Entity Integrity. Ignore Duplicate and accept one null value.
Not Null	Domain integrity. Ignore null value
Check	Domain Integerity. To satisfy condition
Foreign Key	Referencial integrity Refer the another table, primarykey or un constraints

About nested and anonymous class ?

In **Java**, just like methods, variables of a **class** too can have another **class** as its member. Writing a **class** within another is allowed in **Java**. The **class** written within is called the **nested class**, and the **class** that holds the **inner class** is called the outer **class**.

It is an inner **class** without a name and for which only a single object is created. An **anonymous** inner **class** can be useful when making an instance of an object with certain "extras" such as overloading methods of a **class** or interface, without having to actually subclass a **class**

A concrete class is a class that has an implementation for all of its methods that were inherited from abstract or implemented via interfaces

```
. Test t = new Test()
{
  // data members and methods
  public void test_method()
  {
    ........
    ........
   }
};
```

About multi-dimensional array ?

to create an array of arrays known as multidimensional array

```
int[][] a = new int[3][4];
```

Here, a is a two-dimensional (2d) array. The array can hold maximum of 12 elements of type int.

About string join and concat ?

new Method join() in the **Java String** class. Java String join() method **concatenates the given Strings** and returns the concatenated String.

```
public class Example{
  public static void main(String args[]){
        //The first argument to this method is the delimiter
        String str=String.join("^","You","are","Awesome");
        System.out.println(str);
  }
}
```

To concat:

```
String s1="hello";
s1= s1.concat("world").concat(".").concat("com");
```

About data structures?

a **data structure** is a particular way of storing and manipulating the internal **data** of a computer program.
data structure types include the array, the file, the record, the table, the tree, and so on

About collection.sort?

Collecrion.sort - It is used to sort the elements present in the specified list of Collection in ascending order.

Collections.sort(al);

Reverse order:
Collections.sort(al, Collections.reverseOrder());

For userdefined sort-
We can use Comparator Interface for this purpose.

About Stub?

A **stub** is a small program routine that substitutes for a longer program, possibly to be loaded later or that is located remotely.

About Java Regex / Matchers / Patters / dot rep?

The **Java Regex** or Regular Expression is an API to *define pattern for searching or manipulating strings*.
It is widely used to define constraint on strings such as password and email validation.
Java Regex API provides 1 interface and 3 classes in **java.util.regex** package.

Matcher Class- It implements **MatchResult** interface. It is a *regex engine* i.e. used to perform match operations on a character sequence.
Pattern
It is the *compiled version of a regular expression*. It is used to define a pattern for the regex engine.

The . (dot) represents a single character.

```
import java.util.regex.*;
class RegexExample2{
public static void main(String args[]){
System.out.println(Pattern.matches(".s", "as"));//true (2nd char is s)
System.out.println(Pattern.matches(".s", "mk"));//false (2nd char is not s)
System.out.println(Pattern.matches(".s", "mst"));//false (has more than 2 char)
System.out.println(Pattern.matches(".s", "amms"));//false (has more than 2 char)
System.out.println(Pattern.matches("..s", "mas"));//true (3rd char is s)
}}
```

About Selenium and its types ?

Selenium:
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 systems. Selenium is not just a single tool but a set of tools that helps testers to automate web-based applications more efficiently.

Tool	Description
Selenium IDE	Selenium Integrated Development Environment (IDE) is a Firefox plugin that lets testers to record their actions as they follow the workflow that they need to test.
Selenium RC	Selenium Remote Control (RC) was the flagship testing framework that allowed more than simple browser actions and linear execution. It makes use of the full power of programming languages such as Java, C#, PHP, Python, Ruby, and PERL to create more complex tests.
Selenium WebDriver	Selenium WebDriver is the successor to Selenium RC which sends commands directly to the browser and retrieves results.
Selenium Grid	Selenium Grid is a tool used to run parallel tests across different machines and different browsers simultaneously which results in minimized execution time.

About advantages and disadvantages of selenium?

Advantages of selenium:
Selenium is an open-source tool.
Has capabilities to execute scripts across different browsers.
Can execute scripts on various operating systems.

Disadvantage:
Supports only web-based applications.
No default test report generation.

About Selenium webdriver ?

Selenium Webdriver:
Selenium WebDriver is a tool for automating testing web applications.
WebDriver interacts directly with the browser without
any intermediary, unlike Selenium RC that depends on a server

About Locators ?

Locators
Locating elements in Selenium WebDriver is performed with the help of findElement() and findElements() methods provided by WebDriver and WebElement class.

findElement() returns a WebElement object based on a specified search criteria.
Used to find locators.

findElements() returns a list of WebElements matching the search criteria. If no elements are found, it returns an empty list.

Element Locators help Selenium to identify the HTML element the command refers to.
Here an object is accessed with the help of IDs. In this case,
By locators-
Id
Name
Classname
Xpath
tagname
Link
Partial link
Css selectors

About types of browser name and driver / class name ?

Browser	Driver Name	Class name
Chrome browser	Chromedriver.exe	ChromeDriver();
IIE browser	Iedriverserver.exe	FirefoxDriver();
Firefox browser	Geckodriver.exe	InternetExplorerDriver()

About selenium webdriver advantages

Some advantages of Selenium are:

1. It is free and open source
2. It has a large user base and helping communities
3. It has cross Browser compatibility (Firefox, chrome, Internet Explorer, Safari etc.)
4. It has great platform compatibility (Windows, Mac OS, Linux etc.)
5. It supports multiple programming languages (Java, C#, Ruby, Python, Pearl etc.)
6. It has fresh and regular repository developments
7. It supports distributed testing-

About testing supported in selenium webdriver ?

Selenium supports two types of testing:
Regression Testing: It is the act of retesting a product around an area where a bug was fixed.
Functional Testing: It refers to the testing of software features (functional points) individually

About continuous and automation testing ?

•**Continuous Testing** is the process of executing automated tests as part of the software delivery pipeline to obtain immediate feedback on the business risks associated with in the latest build.

• In this way, each build is tested continuously, allowing Development teams to get fast feedback so that they can prevent those problems from progressing to the next stage of Software delivery life-cycle.

•This dramatically speeds up a developer's workflow as there's no need to manually rebuild the project and re-run all tests after making changes.

•**Automation testing** or Test Automation is a process of automating the manual process to test the application/system under test. Automation testing involves use of separate testing tools which lets you create test scripts which can be executed repeatedly and doesn't require any manual intervention

About different methods to enter and click in web app ?

Different ways to enter text:
Passing values to webelement.

1.Using webelement SendKeys()
Element.sendkeys();
2.Using Actions class
ac.sendKeys(txtUser, "premkumar");
3.Java Script
js.executeScript("arguments[0].setAttribute('value','Hello')",webelement

Different ways to Click:
To click webelement.

1.Using webelement click ();
Element.click();
2.Using Actions class
ac.sendKeys(Keys.*ENTER).*
3.Java Script:
js.executeScript("arguments[0].click()", findElement);

About Launching browser ?

Launch browser:

//means exactly what it says. Set the system property propertyName to have the value value.
System.setProperty("webdriver.chrome.driver", “path");
WebDriver driver = new ChromeDriver();
//if property not set . Error: The path to the driver executable must be set by the webdriver.chrome.driver system property

WebDriver driver = new ChromeDriver();
Webdriver – Interface
Driver – reference of webdriver
New – creating new object for webdriver
Chromedriver() - Class

About different methods in Selenium webdriver?

Thread.sleep(3000) it is used wait for some time(here,3000ms) to execute

getCurrenturl():
It is a method, used to check whether particular page is open or not

getTitle()
It is a method, used to print the title of the page

getText():
It is a method, used to print the value whatever you gave in the text box

getAttribute(value)
To get the attribute value using element.getAttribute(attributeName)'.
If we try to get the attribute value that doesn't exists for the tag, it will return null value.

Method	Description	Return type
isDisplayed()	used to check the particular id/value is available or not	Boolean
isEnabled()	used to check particular text box is enable to print or not	boolean
isSelected()	used to check the particular radio button is selected or not	boolean

NoSuchElementException:
It is throws when particular id/xpath/class or whatever is not available in DOM structure

Close()	It will close the current browser.
Quit()	Calls dispose method. Destroy the object.
Dispose()	Close all browser and safely ends the session.

About navigation commands ?

Navigate commands:

1.navigate().to():

It is a method, used to navigate one website to another website.

Driver.navigate().to(url)

2.refresh():

It is a method, used to refresh the navigation website

Driver.navigate().refresh()

3.back():

It is a method, used to move back to the first page

Driver.navigate().back()

4.forward();

It is a method, used to move forward to next page

Driver.navigate().forward();

About debugging steps ?

Eclipse Debugging steps:

When our test case not working properly, we can check step by step execution by using debugging method

It is used run a program step by step in eclipse.

Mark some lines from left corner of the program

Then right click debug as

Then click to check step over/step down/step info in the eclipse top

About xpath and different ways ?

Xpath:

Structure or combination of absolute path and relative path

Finding Xpath using different ways:

1. //label[contains(text(), “female”)]

2. //label[contains(@class , “”]

3. //label[text(), “hcdjhb”]

2. (**//**label[@class=‟-58mg‟])[1]

If we get 2 matching points , we give 1 or 2

3. //input[@id=‟email‟]

4. //*[@id=‟email‟]

If we use *, consider any tag

About Check box ?

CHECK BOX:
In check box, we can able to select more than one value at a time.

To select all particular check box-

```
List<WebElement> ck1 = driver.findElements(By.xpath("//input[@type='checkbox']"));
for (int i = 0; i < 3; i++) {
if (ck1.get(i).getAttribute("value").equals("dance")) {
ck1.get(i).click();
}
```

To select all check box- using list and enhanced for loop and click.

To select more than one value- using list and enhanced for loop and using if condition click check box

About web tables ?

WEB TABLE:
tr Table row
th Table heading
td Table data

To get all cell values:

```
List<WebElement> tRow = driver.findElements(By.tagName("tr"));
for (WebElement rows : tRow) {
List<WebElement> td = driver.findElements(By.tagName("td"));
for (WebElement data : td) {
String text = data.getText();
System.out.println(text);
```

To get text for particular cell:

```
driver.findElement(By.xpath("//*[@id='content']/table/tbody/tr[2]/td[2]")).getText();
```

About absolute and relative path ?

Absolute path	Relative path
An absolute xpath in HTML DOM starts with /html	A relative xpath finds the closed id to the dom element and generates xpath starting from that element
/html/body/div[5]/div[2]/div/div[2]/div[2]/h2[1]	.//*[@id='answers']/h2[1]/a[1]

About drop down and its methods ?

DROP DOWN:
To select values from drop down box
1.Single value
2.Multiple value
To select/deselect values create one class called “Select”

Ex: for select using visible text method
Select sc = **new Select(element);**
sc.selectByVisibleText(val);

Methods in drop down:

1. SelectByIndex
2. SelectByValue
3. SelectByVisibletext
4. Getoptions
5. Getallselectedoptions
6.Getfirstselectedoption
7. Ismultiple
// to deselect
8. deSelectByIndex
9. deSelectByValue
10. deSelectByVisibletext
11.Deselectall

isMultiple():
It is a method, used to check we can able to select multiple values or not

getAllSelectedOptions()
It is a method, used to print all selected options

About mouse over actions ?

MOUSE OVER ACTION:
Using Actions class.

Actions a=new Actions(driver);
a.moveToElement(element).perform();
Actions class
Action interface

About screenshot ?

SCREENSHOT:
using
TakeScreenShot – Interface
Assigning driver interface to takescreenshot interface.
File ll stored in temp folder.

```
TakesScreenshot tk=(TakesScreenshot) driver;
//temp
File f = tk.getScreenshotAs(OutputType.FILE);
File f1=new File("F:/facebook.png");
FileUtils.copyFile(f,f1 );
```

About right click / context click ?

Right Click:

Using Actions class with contextclick method to right click
And
Using KEYBOARD ACTIONS
Create robot class with keypress and keyrelease method

```
public static void main(String[] args) throws AWTException {
BaseClass b =new BaseClass();
WebDriver driver = b.getDr("https://www.google.co.in/");
WebElement target = driver.findElement(By.xpath("(//a[@class='gb_P'])[2]"));
Actions ac =new Actions(driver);
ac.contextClick(target).build().perform();
Robot r =new Robot();
r.keyPress(KeyEvent.VK_DOWN);
r.keyRelease(KeyEvent.VK_DOWN);
r.keyPress(KeyEvent.VK_ENTER);
r.keyRelease(KeyEvent.VK_ENTER);
}
```

About singleton and design factory pattern ?

Singleton Pattern:

Singleton Pattern says that just**"define a class that has only one instance and provides a global point of access to it".**
In other words, a class must ensure that only single instance should be created and single object can be used by all other classes.
Advantage of Singleton design pattern
Saves memory because object is not created at each request. Only single instance is reused again and again.

```
public class SingleFactory {

public static void main(String[] args) {
Singleton sin = Singleton.getInstance();
Factory fac = Factory.getinstance();
sin.getName();
fac.getName();
}}

public class Singleton {
private static Singleton s= null;

private Singleton() {
System.out.println("13447654");
}
public static Singleton getInstance() {
if (s==null) {
s= new Singleton();
}
System.out.println(s);
return s;
}
public void getName() {
System.out.println("ndskjandksnd");
}}
```

Design Factory Pattern:
In Factory pattern, we create object without exposing the creation logic to the client and refer to newly created object using a common interface.

About scrolling and its type?

Scroll up & Scroll down:
Scroll up/Scroll down method using JavaScript,
In this method, we first assign up to which path locator/web element we want to down/up and perform
Example Program:

To scroll up:
```
JavascriptExecutor js = (JavascriptExecutor) driver;
js.executeScript("arguments[0].scrollIntoView(true);", element);
```
To scroll down:
```
JavascriptExecutor js1 = (JavascriptExecutor) driver;
js1.executeScript("arguments[0].scrollIntoView(false);", element);
```

To scroll using pixel:
```
JavascriptExecutor jse = (JavascriptExecutor) driver;
jse.executeScript("scroll(0, 250);");
```

To scroll to bottom page:
```
js.executeScript("window.scrollTo(0, document.body.scrollHeight)");
```

To scroll horizontal:
```
js.executeScript("arguments[0].scrollIntoView();",Element );
```

Using Robot Class: (JAVA.AWT)
robot.keyRelease(KeyEvent.***VK_PAGE_DOWN);***

Using Actions class:
ac.sendKeys(Keys.***PAGE_DOWN).build().perform();***

About drag and drop?

Drag and Drop:
Drag and drop which is used to move web element from one place to another place
Using actions class and with draganddrop method;

Actions acc =**new Actions(driver);**
acc.dragAndDrop(source, target).build().perform();

About screenshot ?

What is a headless browser?
used to define browser simulation programs which do not have a GUI. These programs behave just like a browser but don't show any GUI. Famous ones are HtmlUnit and the NodeJs headless browsers.

What is the use of Headless browsers?
Headless browsers are typically used in following situations
1. You have a central build tool which does not have any browser installed on it. So to do the basic level of sanity tests after every build you may use the headless browser to run your tests.
2. You want to write a crawler program that goes through different pages and collects data, headless browser will be your choice. Because you really don't care about opening a browser. All you need is to access the webpages.
3. You would like to simulate multiple browser versions on the same machine. In that case you would want to use a headless browser, because most of them support simulation of different versions of browsers. We will come to this point soon.

About stale element exception?

A stale element reference exception is thrown in one of two cases, the first being more common than the second:
The element has been deleted entirely.
The element is no longer attached to the DOM.

To allow for those cases, you can try to access the element several times in a loop before finally throwing an exception.
Using try catch

About uploading a file ?

Upload a file:

1.Using Sendkeys: uploadElement.sendKeys("C:\\Users\\ELCOT\\Downloads\\hearts-clipart-10.gif");

2. **Using Robot Class**
3. **Using AutoIT**

About keyboard actions?

Keyboard Actions and Mouse Actions: (Multiple actions)
Handling special keyboard and mouse events are done using the Advanced User Interactions API

1.keyDown(Keys modifierKey)-

It is used to simulate the action of pressing a modifier key, without releasing. The expected values for the keyDown() method are - Keys.SHIFT, Keys.ALT and Keys.CONTROL only, passing key other than these results in IllegalArgumentException.

2.keyUp(Keys modifierKey)-
The keyUp() method is used to simulate the modifier key-up or key-release action. This method follows a preceeding key press action.

3.sendKeys(CharSequence KeysToSend)-
The sendKeys(CharSequence KeysToSend) method is used to send a sequence of keys to a currently focussed web element. Here, we need to note that it is different from the webElement.sendKeys() method.

Note: If you want to particular webelement pass the webelement first and key

Ex; keyUp(WebElement element, Keys modifierKey)-
Given below are the methods to perform keyboard actions:

```
//find element
WebElement txtUser =driver.findElement(By.id("email"));
Actions ac =new Actions(driver);
//mouse hover
ac.moveToElement(txtUser);
//click
ac.click();
//key press till expected values
ac.keyDown(txtUser, Keys.SHIFT);
//send values
ac.sendKeys(txtUser, "premkumar");
//key release
ac.keyUp(txtUser, Keys.SHIFT);
//double click or highlight the text
ac.doubleClick(txtUser);
//right click
ac.contextClick();
ac.build().perform();
```

About mouse actions?

Mouse actions Method:

Click - Performs a Click. We can also perform a click based on coordinates.
contextClick - Performs a context click/right-click on an element or based on the coordinates.
doubleClick - Performs a double-click on the webelement or based on the coordinates. If left empty, it performs double-click on the current location.
mouseDown - Performs a mouse-down action on an element or based on coordinates.
mouseMove - Performs a mouse-move action on an element or based on coordinates.
mouseUp - Releases the mouse usually followed by mouse-down and acts based on coordinates.

About javascript method in selenium?

JAVASCRIPT:
Using javascript to enter text and click.

Syntax:

```
JavascriptExecutor js=(JavascriptExecutor) driver;
js.executeScript("document.getElementById('email').setAttribute('value','Hello'
)");
```

Another method

```
JavascriptExecutor js=(JavascriptExecutor) driver;
js.executeScript("arguments[0].setAttribute('value','Hello')",webelement
```

About find all link in page ?

Find All Links
We can easily
do so by finding all elements with the Tag Name "a", as we know that for any link reference in HTML, we need to use "a" (anchor) tag.

```
java.util.List<WebElement> links =
driver.findElements(By.tagName("a"));
System.out.println("Number of Links in the Page is " +
links.size());
for (int i = 1; i<=links.size(); i=i+1)
{
System.out.println("Name of Link# " + i - +
links.get(i).getText());
}}
```

About windows handling?

WINDOW HANDLING:
It is used to move one window to another window.

To get parent window id:
String pId = driver.getWindowHandle();
To get all windows id:
Set<String> childId = driver.getWindowHandles();

To move to particular window:

```
Set<String> childId = driver.getWindowHandles();
List<String> list = (List<String>) childId;
int count = 0;
for (String string : list) {
count ++;
if (count==4) {
driver.switchTo().window(string);
}}
```

Using foreach to move to child window

```
for (String x : childId) {
if (!pId.equals(x)) {
driver.switchTo().window(x);
}
}
```

Without using if condition:

```
Set<String> childId = driver.getWindowHandles();
List<String> list = (List<String>) childId;
Driver.switchto().window(list.get(index))
```

About browser popup ?

Browser level Pop-up:

//Create a instance of ChromeOptions class and add arguments :

```
ChromeOptions options = new ChromeOptions();
 options.addArguments("--disable-notifications");
 System.setProperty("webdriver.chrome.driver","path/to/driver/exe");
 WebDriver driver =new ChromeDriver(options);
```

About alerts ?

Alerts :
It has 3 types,
1. Simple Alert
2. Confirm pop-up
3. Prompt pop-up

1. Simple:
In this method, we just click single ok button in the alert.

```
Alert al = driver.switchTo().alert();
 al.accept();
```

2. Confirm:
In this method, we have to click OK or Cancel, if we click OK/Cancel, then it will confirm.

```
Alert al = driver.switchTo().alert();
al.dismiss();
Or
 al.accept();
```

3. Prompt:
In this method, first we have insert Yes/No and then we will click OK/Cancel

```
Alert al = driver.switchTo().alert();
alert2.sendKeys("prem");
alert2.accept();
```

About frames ?

Frames:
To switch to frames

What is Iframe?

IFrame is a web page which is embedded in another web page or an HTML document embedded inside another HTML document.
The IFrame is often used to insert content from another source, such as an advertisement, into a Web page. The <**iframe**> tag specifies an inline frame.

By Index
By Name or Id
By Web Element

Suppose if there are 100 frames in page, we can switch to the iframe by using index.
Ex:

```
driver.switchTo().frame(0);
driver.switchTo().frame(1);
```

About synchronizations , Waits and its types ?

WAITS:

Synchronization
To synchronize between script execution and application, we need to wait after performing appropriate actions.

If we trying to find the elements due to application late response , it throws exception.
Using waits we can resolve this exception

Implicit wait:

It applicable for all elements The implicit wait will tell to the web driver to wait for certain amount of time before it throws a "No Such Element Exception".
Once we set the time, web driver will wait for that time before throwing an exception.
Syntax:
driver.manage().timeouts().implicitlyWait(time, TimeUnit.SECONDS);

Explicit wait:

The explicit wait is used to tell the Web Driver to wait for particular element to certain conditions (Expected Conditions) or the maximum time exceeded before throwing an exception.
Once we declare explicit wait we have to use "ExpectedCondtions" or we can configure how frequently we want to check the condition using Fluent Wait.
Syntax:
WebDriverWait - class
WebDriverWait wait =**new WebDriverWait(driver, time);**
wait.until(ExpectedConditions.*visibilityOf(element));*

Fluent Wait: Let's say you have an element which sometime appears in just 1 second and some time it takes minutes to appear. In that case it is better to use fluent wait, as this will try to find element again and again until it find it or until the final timer runs out.

public static void fluent(WebElement element) {
FluentWait<WebDriver> f = **new FluentWait<WebDriver>(*driver);***
f.withTimeout(30, TimeUnit.***SECONDS);***
f.pollingEvery(1, TimeUnit.***SECONDS);***
f.ignoring(NoSuchElementException.**class);**
f.until(ExpectedConditions.*visibilityOf(element)).click();*

}

About dynamically changing xpath / Stale exception / webdriver architecture / POI / Severity and priority / volatile methods?

how-to-handle-dynamic-changing-ids-in-xpath

1.By using full absolute path
/html/body/div[3]/div[2]/div[2]/div[1]/form[1]/input[1]
2. We can use starts-with function. In this xpath's ID attribute, "post-body-" part remain same every time.
//div[starts-with(@id,'post-body-')]/div[1]/form[1]/input[1]
3. We can use contains function. Same way you can use contains function as bellow.
div[contains(@id,'post-body-')]/div[1]/form[1]/input[1]

Stale exception:
This can happen if a DOM operation happening on the page is temporarily causing the element to be inaccessible.
To allow for those cases, you can try to access the element several times in a loop before finally throwing an exception.
And also give wait until element visible.

WebDriver's architecture is simpler than Selenium RC's.
It controls the browser from the OS level
All you need are your programming language's IDE (which contains your Selenium commands) and a browser.

Webserver <<----->> Browser<< ---- >> Selenium Commands

Apache **POI** for Data-driven Tests.
Apache **POI** is the most commonly used API for**Selenium** data driven tests.
POI is a set of library files that gives an API to manipulate Microsoft documents like .xls and .xlsx. It lets you create, modify, read and write data into Excel.

Severity of a defect is related to how severe a bug is.
Usually the **severity** is defined in terms of financial loss, damage to environment, company's reputation and loss of life.

Priority of a defect is related to how quickly a bug should be fixed and deployed to live servers.

Volatile is used to indicate that a variable's value will be modified by different threads.
Using volatile is yet another way (like synchronized, atomic wrapper) of making class thread safe
the volatile keyword in Java is poorly documented, poorly understood, and rarely used

About optimizations and SRS / Boundary value analysis and Marker interface and Run time Execution ?

Optimization:

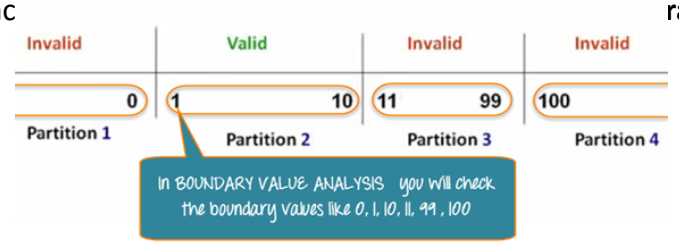
Largest challenges with creating any test automation suite is
simultaneously ensuring scalability, maintainability and reliability.

Using 3 Approaches:
Page Object Model
Behavior Driven Development
Keyword Driven Testing

Software Requirement Specification (**SRS**) A **software** requirements specification (**SRS**) is a detailed description of a **software** system
to be developed with its functional and non-functional requirements.
Using the **Software** requirements specification (**SRS**) document
on QA lead, managers creates **test** plan

The Requirements Traceability Matrix (**RTM**) is a document that links requirements throughout the validation process. The purpose of the Requirements Traceability Matrix is to ensure that all requirements **defined** for a system are **tested** in the **test**protocols.

Boundary Value Analysis- in Boundary Value Analysis, you test boundaries between equivalenc **range checking**.



Marker interface in Java
It is an empty interface (no field or methods). Examples of marker interface are Serializable, Clonnable and Remote interface. All these interfaces are empty interfaces.

To Open a file using Selenium (to open note pad etc): How to open note pad using selenium:
1. Using If

```
if (Desktop.isDesktopSupported()) {
Desktop.getDesktop().edit(file); }
else {} // dunno, up to you to handle this
```

2. Using Proces Builder

```
ProcessBuilder pb = new ProcessBuilder("Notepad.exe", "myfile.txt");
pb.start();
```

3. Using Runtime

```
Runtime runtime = Runtime.getRuntime();
runtime.exec("C:\\path\\to\\notepad.exe C:\\path\\to\\file.txt");
```

FRAMEWORKS AND ITS TYPES:

FRAMEWORK:
A framework is a set of concept and practices, that provide support for automation testing.
Selenium Framework is a code structure that helps to make code maintenance easy. Without frameworks, we will place the “code” as well as “data” in the same place which is neither re-usable nor readable.

Different frameworks are available like,
POM(Page Object Model)
Junit
TetNG
Cucumber
Data driven

Types of Framework:

1.Keyword Driven Testing Framework: The Keyword Driven testing framework is an extension to Data-driven Testing Framework in a sense that it not only segregates the test data from the scripts, it also keeps the certain set of code belonging to the test script into an external data file.
2.Hybrid Testing Framework: Hybrid Testing Framework is a combination of more than one above mentioned frameworks. The best thing about such a setup is that it leverages the benefits of all kinds of associated frameworks.
3.Behavior Driven Development Framework: Behavior Driven Development framework allows automation of functional validations in easily readable and understandable format to Business Analysts, Developers, Testers, etc.

Maven

Maven configuration

1. Downlaod Maven Source file from maven.apache.org
2. Config the JAVA_HOME, MAVEN_HOME in Environmental varaibles
3. Verify Maven installed correctly using mvn –version in CMD Prompt
4. Install maven plugin in eclipse
5. Create Maven project
6. Add maven depencies in pom.xml

Why MAVEN ?

- Insatlled jars easily by adding dependencies
- Created default folder structures.

Maven	Gradle	Ant
Pom.xml	Build.gradle	Build.xml
Maven also uses XML as the format	Instead of XML, Gradle has its own DSL based on Groovy (one of JVM languages)	Ant uses XML as the format.
Maven add dependencies on its own Maven introduced the ability to download dependencies over the network	Gradle used Apache Ivy for its dependency management.	Ant is simply a Build tool, It doesn't have dependency management system.
mvn clean mvn compile mvn verify mvn install mvn clean install mvn test	gradle clean gradle assemble gradle build	ant clean ant ant compile

Build tool goals:

ANT Commands	GRADLE commands
ant clean	gradle clean
ant	gradle assemble
ant compile	gradle build

MAVEN commands:

mvn clean
mvn compile
mvn verify
mvn install
mvn clean install
mvn test

Maven is based around the central concept of a build lifecycle.
There are three built-in build lifecycles:
default
clean
site

Phases	Descriiption
validate	validate the project is correct and all necessary information is available
compile	compile the source code of the project
test	test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
package	take the compiled code and package it in its distributable format, such as a JAR.
integration-test	process and deploy the package if necessary into an environment where integration tests can be run
verify	run any checks to verify the package is valid and meets quality criteria
install	install the package into the local repository, for use as a dependency in other projects locally
deploy	done in an integration or release environment, copies the final package to the remote repository for sharing with other developers and projects
Clean	Cleans the target folder,(compiled code) , installed packages etc. / projects

POM

POM:
Page Object Model
POM is an object repository design pattern in selenium webdriver.
POM creates our testing code maintainable and reusable
Page factory is an optimized way to create object repository

Advantages
POM is an implementation where test objects and functions are separated from each other, thereby keeping the code clean.
The objects are kept independent of test scripts. An object can be accessed by one or more test scripts, hence POM helps us to create objects once and use them multiple times.
Since objects are created once, it is easy to access as well as update a particular property of an object.

It helps make the code **more readable, maintainable**, and **reusable.**

What is Page Factory?
Page Factory is an inbuilt Page Object Model concept for Selenium WebDriver but it is very optimized.
Here as well, we follow the concept of separation of Page Object Repository and Test Methods. Additionally, with the help of PageFactory class, we use annotations **@FindBy** to find WebElement. We use initElements method to initialize web elements

POM Steps & Rules:
Create a maven project/java project
We have to create 3 source folders(packages)
1. Src/main/java
It contains POM information.
In the main package /POM package, we have to create different class for different web pages,
Then all the annotations should be in private and use getters & setters
Create a default constructor and contains one page factory
This is a gateway to initialize the value
2. Src/test/java (Junit,TestNG)
In the test package(Junit), all the methods/annotations should be in public(access specifier).
This package get the variables from main pom package and initialize(input) the value

It contains login, asserts & registration
3. src/resources/java
It contains reusable methods/codes.
Ex, browser coding, radio button, scroll down and etc.

@FindBy can accept **tagName, partialLinkText, name, linkText, id, css, className, xpath** as attributes.

@FindBys : When the required WebElement objects need to match all of the given criteria use @FindBys annotation
@FindAll : When required WebElement objects need to match at least one of the given criteria use @FindAll annotation

```
@FindBys( {
  @FindBy(className = "class1")
  @FindBy(className = "class2")
} )
private List<WebElement> elementsWithBoth_class1ANDclass2;

@FindAll({
  @FindBy(className = "class1")
  @FindBy(className = "class2")
})
private List<WebElement> elementsWithEither_class1ORclass2
```

JUNIT

JUNIT:

What is Unit testing?
Unit testing is a *testing* methodology where *individual units are tested* in isolation from other units. This is usually done by ***developers***.
A unit can be considered as a class or methods inside a class which needs to be tested individually. It is also know as ***White Box*** Testing, as developer is able to see the code for the functionality.
What is Junit?
To do *Unit testing* in *Java* we have an excellent framework called ***Junit***. Junit is a *unit testing framework*. This frameworks provides us with following facilities
– Base classes and Annotations to write unit tests
– Base class support to run tests, TestRunner class.
– Base class and Annotation support to write test suites, @RunWith(Suite.Class)
– And of course reporting of test results.

Test suite is used to bundle a few unit test cases and run them together. In JUnit, both **@RunWith** and @Suite annotations are used to run the suite tests.
@RunWith annotation is used to specify its runner class name.

JUnit is a unit testing framework for the Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks which is collectively known as xUnit that originated with SUnit.

We can perform all test action here
First launch the browser which one we want
Then Verify the login details
Assert class
assertTrue() is a method used to check the particular value is available or not.
If its not available, it will fail the test case
As well as assertEquals() is also one of the method to check
Then close the browser
Ex:
Assert.*assertEquals(expected, actual);*
Assert.*assertTrue(**true);***

To run the suite test, you need to annotate a class using below-mentioned annotations:
@Runwith(Suite.class)
@Suite.SuiteClasses ({test1.class, test2.class})
With above annotations, all the test classes in the suite will start executing one by one.

Suite – collection of test cases

Junit Methods:

```
//To get result
Result result = JUnitCore.runClasses(Junit.class, Junit1.class);
//to get failurecount
System.out.println(result.getFailureCount());
//to get ignore count
System.out.println(result.getIgnoreCount());
// to get run time for test case
System.out.println(result.getRunTime());
// to get run count
System.out.println(result.getRunCount());
// to get failures cases
List<Failure> failures = result.getFailures();
for (Failure failure : failures) {
System.out.println(failure);
}
```

Annotations used in JUNIT :

@Test

it tells JUnit framework that the following is a Test Method.

@Before

The @Before annotation is used to identify the method which is executed before executing the Test Method. This method can be used to set up the test environment.

@After

The @After annotation is method which is executed after executing the Test Method. This method can be used to do teardown i.e. deleting temporary data or setting up default values or cleaning up test environment etc.

@BeforeClass

The @BeforeClass method before start all tests. Basically this is used to perform time intensive activities, like connect to a database.

@AfterClass

The @AfterClass method is used only once after finish executing all tests. Basically this is used to perform clean-up activities, like disconnect from a database.

@Ignore: Method annotated as @Ignore lets the system know that this method shall not be executed.

Assert: if the assert condition is true then the program control will execute the next test step but if the condition is false, the execution will stop and further test step will not be executed.

To overcome this we use Soft Assert in TestNG

When a "**verify**" command fails, the test will continue executing and logging the failure.

Order of EXECUTION Junit Framework:

@Before Class
@Before
@Test
@After
@Before
@Test
@After
@After Class

TestNg:

•Generate default report
•cross browser and parallel execution possible
•Grouping of test case is easy
•Re run failed test cases
•Generate the log automatically
•Order and priority can be set
•data parameter is using data provider

TestNg

Points to Remember:

•suite - collections of test cases.
•test cases will execute in ascending order by default
•if you want to run 1000 times the particular test case use @test (**invocationcount** = 1000)
•if you want to set order use @**test(priority** = 1) priority starts from -ve to +ve
•if both non priority and priority methods, first it ll give preference to non prirority method (ascending order) and then go to priority method
•if we give same test name in **testng.xml** file , it will throw error
•if you want to give instruction to compiler which method execute first use @**test(dependsonMethod**= {"test", "test2"})
•if we have super class test cases, first it ll execute super class then sub class
•if you want to give **dependsonmethod** for super class, first extends super class and give that method name in dependsonmethod
•if you want to ignore particular test case, use @test(enabled=false) , by default it will be true.
•if you want to run only one test case over 100 test case, in testng.xml
use
<method>
<**include** name="test"/>
</method>
to exclude in testng.xml
use <**exclude**> instead of <include>

timeout:
to execute the method with in particular time
use @test (timeout= 300)
we can use that in testng.xml file too
next to test
and for suite level also give next to suite, it is for that all test in that suite

Annotations in TestNG

@BeforeSuite: The annotated method will be run before all tests in this suite have run.
@AfterSuite: The annotated method will be run after all tests in this suite have run.
@BeforeTest: The annotated method will be run before any test method belonging to the classes inside the tag is run.
@AfterTest: The annotated method will be run after all the test methods belonging to the classes inside the tag 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.
@BeforeClass: The annotated method will be run before the first test method in the current class is invoked.
@AfterClass: The annotated method will be run after all the test methods in the current class have been run.
@BeforeMethod: The annotated method will be run before each test method.
@AfterMethod: The annotated method will be run after each test method.
@Test: The annotated method is a part of a test case.

Parallel Execution in Test NG:

set the '**parallel**' attribute to '**tests**' in the above used xml and give a run again. This time you will notice that your both browsers will open almost simultaneously and your test will run in parallel.

<suite name="Suite" parallel="tests"**>**
<test name="FirefoxTest"**>**
<parameter name="browser" value="firefox" **/>**
<classes>
<class name="automationFramework.MultiBrowser" **/>**
</classes>
</test>
<test name="ChromeTest"**>**
<parameter name="browser" value="Chrome" **/>**
<classes>
<class name="automationFramework.MultiBrowser" **/>**
</classes>
</test>

Types of assert:

Hard Assert: same as assert
Soft Assert:
Softassert s =new softassert
s.assertequals/asserttrue
even exception occurs it will skip that and pass the test case
if you want to execute that exception, use s.assertall
at the end it will execute the exception
soft assert is best

TestNG DataProviders

When you need to pass complex parameters or parameters that need to be created from Java (complex objects, objects read from a property file or a database, etc…), in such cases parameters can be passed using Dataproviders. A Data Provider is a method annotated with *@DataProvider*. A Data Provider returns an array of objects.

```
@DataProvider(name = "Authentication")
 public static Object[][] credentials() {
     return new Object[][] { { "testuser_1", "Test@123" }, { "testuser_1", "Test@123" }};
 }
```

And

```
// Here we are calling the Data Provider object with its Name before a method in @test annotation

 @Test(dataProvider = "Authentication")
```

Parameters.

TestNG lets you pass parameters directly to your test methods with testng.xml.

```
<suite name="Suite">
   <test name="ToolsQA">
         <parameter name="sUsername" value="testuser_1"/>
         <parameter name="sPassword" value="Test@123"/>
                  <classes>
                     <class name="automationFramework.TestngParameters" />
                  </classes>
   </test>
</suite>
```

And use in @ test annotation

```
@Test
 @Parameters({ "sUsername", "sPassword" })
```

Retry the failed Test Cases:

Use Interface IRetryAnalyzer implement it and override a method retry
And run failed test cases.
This method decides how many times a test needs to be rerun.
TestNg will call this method every time a test fails. So we can put some code in here to decide when to rerun the test.
Note: This method will return true if a tests needs to be retried and false it not.

And **Specifying retryAnalyzer during runtime**
implement ITestAnnotationTransformer interface. ITestAnnotationTransformer interface falls under a broad category of interfaces called TestNG Listeners. And override transform method. add it as a listener in the testng run xml

Refer: http://toolsqa.com/selenium-webdriver/retry-failed-tests-testng/

Cucumber

Cucumber :

Cucumber is a testing approach which supports Behavior Driven Development (BDD). It explains the behavior of the application in a simple English text using Gherkin language.
Cucumber tool is based on the Behavior Driven Development framework that **acts as the bridge** between the following people:
Software Engineer and Business Analyst.
Manual Tester and Automation Tester.
Manual Tester and Developers.

Cucumber makes **it easy to read and to understand the application flow.**
Cucumber also **benefits the client to understand the application code** as it uses Gherkin language which is in Plain Text.

3 Important Files needs to run cucumber framework:
Cucumber Work Flow:
Write Gherkin Script in feature file and run junit runner class get snippet add in step definition file and again run junit runner class to execute your script

1. Feature file:
A ***Feature File*** is an entry point to the *Cucumber* tests.
This is a file where you will describe your tests in Descriptive language (Like English).
It is an essential part of Cucumber, as it serves as an automation test script as well as live documents.
A feature file can contain a scenario or can contain many scenarios in a single feature file but it usually contains a list of scenarios.

2. Step Definition
A Step Definition is a small piece of *code* with a *pattern* attached to it
A Step Definition is a java method in a class with an annotation above it.
An annotation followed by the pattern is used to link the *Step Definition* to all the matching *Steps*, and the *code* is what *Cucumber* will execute when it sees a *Gherkin Step*.

3. Junit Runner File:
As *Cucumber* uses *Junit* we need to have a ***Test Runner class***.
This class will use the *Junit annotation* **@RunWith(),** which tells *JUnit* what is the *test runner class*. It more like a starting point for Junit to start executing your tests.
In the src folder create a class called ***TestRunner*.**

```
@RunWith(Cucumber.class)
@CucumberOptions(
                features = "Feature"
                ,glue={"stepDefinition"}
                tags={@tagname}
                )
```

Options Type	Purpose	Default Value
dryRun	true: Checks if all the Steps have the Step Definition	false
features	set: The paths of the feature files	{}
glue	set: The paths of the step definition files	{}
tags	instruct: What tags in the features files should be executed	{}
monochrome	true: Display the console Output in much readable way	false
format	set: What all report formaters to use	false
strict	true: Will fail execution if there are undefined or pending steps	false

Example runner class:

Examples with Given, When , Then Keyword:
Feature: Defines what feature you will be testing in the tests below
Given: Tells the pre-condition of the test
And: Defines additional conditions of the test
Then: States the post condition. You can say that it is expected result of the test.

Gherkin Keywords:

Gherkin Keyword	Descriptiom
Feature	*Feature* defines the logical test functionality you will test in this feature file. For e.g if you are testing a payment gateway your *Feature* will become *Payment Gateway*
Background	***Background*** keyword is used to define steps which are common to all the tests in the feature file.
Scenario	Each Feature will contain some number of tests to test the feature. Each test is called a ***Scenario***
Given	***Given*** defines a precondition to the test.
When	***When*** keyword defines the test action that will be executed. By test action we mean the user input action.
Then	***Then*** keyword defines the Outcome of previous When step or Verification purpose.
And	***And*** keyword is used to add conditions to your steps
But	***But*** keyword is used to add negative type comments.
*	This keyword is very special. This keyword defies the whole purpose of having Given, When, Then and all the other keywords. Cucumber doesn't care about what Keyword you use to define test steps, all it cares about what code it needs to execute for each step.

Cucumber Reports:

Format
Format Option is used to specify different formatting options for the output reports. Various options that can be used as for-matters are:

Format Options	Description	Syntax
Pretty	Prints the *Gherkin* source with additional colours and stack traces for errors	***format = {"pretty"}***
HTML	This will generate a HTML report at the location mentioned in the for-matter itfsel.	***format = {"html:Folder_Name"}***
JSON	This report contains all the information from the gherkin source in JSON Format.	***format = {"json:Folder_Name/cucumber.json"}***
JUnit	***:***This report generates XML files just like Apache Ant's JUnit report task. This XML format is understood by most Continuous Integration servers, who will use it to generate visual reports.	***format = { "junit:Folder_Name/cucumber.xml"}***

Parameterization:
Cucumberi nherently supports ***Data Driven Testing using Scenario Outline***.
Scenario Outline – This is used to run the same scenario for 2 or more different set of test data
Examples – All scenario outlines have to be followed with the Examples section. This contains the data that has to be passed on to the scenario.
4 Types of Parameterization:

Parameterization	Syantax
without Example Keyword	And User enters "testuser_1" and "Test@123"
with Example Keyword	And User enters "<username>" and "<password>" Examples: // below the scenario | username | password |
Parameterization using Tables	**And** User enters Credentials **to** LogIn | testuser_1 | Test@153 |
Maps In Data Tables	**And** User enters Credentials **to** LogIn | Username | Password | | testuser_1 | Test@*153* |

Example for parameter using data tables in step definition file*:*

```
@When("^User enters Credentials to LogIn$")
public void user_enters_testuser__and_Test(DataTable usercredentials) throws Throwable {

//Write the code to handle Data Table
List<List<String>> data = usercredentials.raw();
//This is to get the first data of the set (First Row + First Column)
driver.findElement(By.id("log")).sendKeys(data.get(0).get(0));
//This is to get the first data of the set (First Row + Second Column)
   driver.findElement(By.id("pwd")).sendKeys(data.get(0).get(1));
   driver.findElement(By.id("login")).click();
}
```

Example for parameter using map in data tables in step definition file:

```
@When("^User enters Credentials to LogIn$")
public void user_enters_testuser_and_Test(DataTable usercredentials) throws Throwable {
//Write the code to handle Data Table
List<Map<String,String>> data = usercredentials.asMaps(String.class,String.class);
driver.findElement(By.id("log")).sendKeys(data.get(0).get("Username"));
driver.findElement(By.id("pwd")).sendKeys(data.get(0).get("Password"));
driver.findElement(By.id("login")).click();
 }
```

What are Hooks in Cucumber?

Cucumber supports ***hooks***, which are blocks of code that run ***before*** or ***after***each scenario. You can define them anywhere in your project or step definition layers, using the methods ***@Before*** and ***@After***. ***Cucumber Hooks*** allows us to better manage the code workflow and helps us to reduce the code redundancy.

Things to note

An important thing to note about the after hook is that even in case of test fail, after hook will execute for sure.

Method name can be anything, need not to be beforeScenario() or afterScenario(). can also be named as setUp() and tearDown().

Make sure that the package import statement should be ***import cucumber.api.java.After;***
& import cucumber.api.java.Before;

what is regular expressions?

A regular expression is a pattern describing a certain amount of text. The most basic regular expression consists of a single literal character

Refer more for cucumber : http://toolsqa.com/cucumber/

Data Driven Framework

Data Driven Test Framework

In data driven framework all of our test data is generated from some external files like Excel, CSV, XML or some database table.

XSSFWorkbook: Is a class representation of XLSX file.
HSSFWorkbook: Is a class representation of XLS file.

Interface:

Workbook – XSSFWorkBook implements Workbook Interface
and
Sheet, Row, Cell – Interface

sheet.getPhysicalNumberOfRows() – to get no of rows.
row3.getPhysicalNumberOfCells() – to get no of cells
cell.getStringCellValue() – convert cell to string.
cell2.getNumericCellValue() – convert cell to numeric

```
public void method() throws Throwable {
// Import excel sheet.
File fileExcel =new File("path");
//Create an object of FileInputStream class to read excel file
FileInputStream stream= new FileInputStream(fileExcel);
//create object of XSSFWorkbook class
Workbook wbook =new XSSFWorkbook(stream);
//Read sheet inside the workbook by its name
Sheet sheet = wbook.getSheet("Demo");
//Find number of rows in excel file
for (int i = 0; i < sheet.getPhysicalNumberOfRows(); i++) {
Row row = sheet.getRow(i);
for (int j = 0; j < row.getPhysicalNumberOfCells(); j++) {
//find the cell using iteration
Cell cell2 = row.getCell(j);
int cellType = cell2.getCellType();
if (cellType==0) {
double d = cell2.getNumericCellValue();
long l =(long) d;
String string = String.valueOf(l);
System.out.println(string);
}else if (cellType==1) {
String string = cell2.getStringCellValue();
System.out.println(string);
}}}}
```

```
        public static String getDataExcel(int r, Object c) throws Throwable {
            List<LinkedHashMap<String, String>> mapdatalist = new ArrayList<LinkedHashMap<String, String>>();
            File loc = new File("C:\\Users\\ELCOT\\eclipse-workspace\\MavenPackage\\excel\\TestNg.xlsx");
            FileInputStream stream = new FileInputStream(loc);
            Workbook book = new XSSFWorkbook(stream);
            Sheet sheet = book.getSheet("Test");
            Row headerrow = sheet.getRow(0);
            for (int i = 1; i < sheet.getPhysicalNumberOfRows(); i++) {
                LinkedHashMap<String, String> mapdata = new LinkedHashMap<String, String>();
                Row row = sheet.getRow(i);
                for (int j = 0; j < row.getPhysicalNumberOfCells(); j++) {
                    Cell cell = row.getCell(j);
                    int cellType = cell.getCellType();
                    if (cellType == 0) {
                        String key = headerrow.getCell(j).getStringCellValue();
                        double d = cell.getNumericCellValue();
                        long l = (long) d;
                        String string = String.valueOf(l);
                        mapdata.put(key, string);
                    }else {
                        String key = headerrow.getCell(5).getStringCellValue();
                        String string = cell.getStringCellValue();
                        mapdata.put(key, string);
                    }}
                mapdatalist.add(mapdata);
            }
            return mapdatalist.get(r).get(c);
        }

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

        String dataExcel = getDataExcel(1, "Input");
        System.out.println(dataExcel);

    }
```