**Core JAVA**

**@. Function and Method**

In java a method is similar to function that should be written inside the class only. We do not have functions in java instead of this we have methods. In C++ they are called ‘member functions’ not methods.

**@. Is String class or data type**

String is a class in java.lang package but in Java, all classes are considered as ‘user-defined data types’ so we can take String as a data type also. String is used to store sequence of characters.

**@. Can we call class as data type** -- Yes, we can call class as ‘user-defined data type’ this is becz user can create a class.

**@. Object Reference** SCB -- System Clip Board

Object Reference is a unique and Object Reference (myDog) is a way to keep memory address info of an **Object** which is stored in SCB. (@**SCB** is a computer memory and it’s nothing but RAM or ROM). Object Reference is points to an object. An object can have any number of references.

**@. Object / Instance of class**

An Instance of class is called object. Object is created using new keyword. Object can have its own state, behaviour and identity. Here state represents data (value (100, 500...Rs)) of an object, behaviour represents functionality of an object such as deposit, withdraw, and Identity represents the unique name of an object. An Object can exist physically (that means physical presence of the object in memory). In java **object** cannot be declared as **static**. To access any non-static method of any class, we need to create an object to that class so that, all non-static methods can able to access. An object can have any number of references Dog dg1 = new Dog(); Dog dg2 = new Dog()…(There is no way to find the actual size of an object on the heap). Object occupies memory space.

**@. ==(assignment) and equals() while comparing strings. which one is reliable** ---- Ref: Java\_Examples\AssignmentOperator\_nd\_equals/ AssignmentOperator2

**==** is an operator and **equals()** is a method both are used to compare to check the equality of two objects. Whenever we create an object using new operator, it will create a new memory location (address) for that object. Here **==** operator is used to check both **address (object references)** and **content** comparison (i.e. address of two objects are the same or not) and equals() method for **only** **content (values of objects) comparison.**

**N:** ‘new’ kw is used to create new instance/object of class so that will allocate new memory for an object on heap space

**@. String constant pool**

String constant pool is a place in heap memory and string objects are held by JVM (Java Virtual Machine). When an object is created to a string class (i.e. String s1 = “Hello”), then it is stored in string constant pool.

**@. Declaration, Initialization & Assignment statements** -- dia

**Declaration** is not to declare value to variable here declares the variable to datatype only. **EX**: int a, b; //Declaration

**Initialization** is the assignment of value to a variable at the time of declaration. **EX**: a = 10; //Initialization

**Assignment** is simply storing the new value to a variable (Throwing away the old value of a variable and replacing it with a new one). **Ex:** int c = a+b; (**or**)

**EX**: int[] arr = { 1, 2, 3 }; // declaration with initialization, allows special shorthand syntax for arrays

arr = { 4, 5, 6 }; // doesn't compile, special initializer syntax invalid here

arr = new int[] { 4, 5, 6 }; // simple assignment, compiles fine

**@. 1. String s = ”Hello” 2. String s = new String (“Hello”)**

In the first statement, equal operator is used to assign the string literal “Hello” to the string variable “s”. In this case, JVM first checks whether the same object is already available in the string constant pool. If it is available, then it creates another reference to it. If it is not available, don’t create any other reference and stores it into the string constant pool. In the second statement, new operator is used to create the string object. in this case, JVM always creates a new object without looking the string constant pool.

**@. String and StringBuffer classes ----** Ref: Java\_WD\_Ex\Java\_Examples2\Mutable\_nd\_Immutable\_Objects

|  |  |
| --- | --- |
| **String** | **StringBuffer** |
| String class is immutable (Immutable objects cannot be modified after creation) | StringBuffer class is mutable |
| String is slow and consumes more memory while concatenate too many strings because every time it creates a new instance. | SB is fast & consumes less memory while concatenate too many strings |
| String class Overrides the equals() of Object class. So, we can compare the contents of two strings by equals(). | SB class doesn't Override any equals() of Object class. |

**N:** If String and StringBuffer both represents string objects, we can’t compare them with each other although if we try to compare them we will get an error.

**@. String, StringBuilder, and StringBuffer**

**String** class is Immutable (**Reason for Immutable:** to maintain security, caching, synchronization, and class loading). Immutable objects are used to store sequence of characters. Ref: Java\_Examples\String\_ImmutableObjsUsedToStoreSeqOfChars

**StringBuilder** class (mutable, non-thread-safe) is used to create and manipulate mutable sequences of characters. It is more efficient than concatenating strings using the ‘+’ operator, as it avoids unnecessary object creations.

**StringBuffer** class (mutable, thread-safe) is used to create and manipulate mutable sequences of characters similar to StringBuilder class. However, StringBuffer is synchronized and thread-safe, making it suitable for multithread environments.

**@. Reason for String is immutable**

String is widely used in security-sensitive operations like authentication, database connections, network URLs, and file paths. If String is mutable, anyone could modify sensitive data.

**N:** To create immutable class: Declare the class as final, make all fields private & final, provide only getter methods, no setter methods.

**@. Are there any other classes whose objects are immutable?** PDT – Primitive datatypes (wrapper classes)

Yes, ‘wrapper classes’ like PDT (boolean, char, byte, short, int, long, float & double) objects are created as ‘immutable’.

**@. Type conversion**

**Implicit Conversion:** Automatic type conversion by the compiler. **Ex:** int a = 10; ==> double b = a;

**Explicit Conversion (Casting):** Manually converting one datatype to another. **Ex:** double x = 10.5; ==> int y = (int)x;

**@. Wrapper Class** Ref: Java\_Examples2\WapperClass

Classes of DTs is called **Wrapper Class**. These are final and immutable. It is a mechanism to converts the PDT (int, char, boolean, etc) into corresponding wrapper class objects (Integer, Character, Boolean, etc) and wise versa. It provides utility methods for performing operations on PDTs those are valueOf(), ParseInt(), etc. To provide additional functionality, type safety and flexibility. It provides feature of:

**Boxing:** Converting a primitive data type into corresponding wrapper class object (E: Integer obj = Integer.valueOf(10);)

**Auto-Boxing:** Automatic conversion from primitive data type to wrapper class object (Ex: Integer obj = 10;).

**Unboxing:** Converting wrapper class object into primitive value (Ex: int val = obj.intValue();).

**Auto-Unboxing:** Automatic conversion from wrapper class object to primitive values(Ex: int val = obj;).

**N:** Data type starts with Upper Case means **WC** and Data type starts with Lower Case means **Primitive Data Type**.

**@. float and double ---- float** can rep. up to 7 digits and **double** can rep. up to 16 digits accurately after decimal point.

**@. StringBuffer and StringBuilder classes**

|  |  |
| --- | --- |
| **StringBuffer** | **StringBuilder** |
| SB is synchronized i.e., thread safe. It means two threads can't call the methods of SB simultaneously | SB is non-synchronized i.e., not thread safe. It means two threads can call the methods of SB simultaneously |
| StringBuffer is less efficient | SB is more efficient than String for String manipulation |

**N:** Both are mutable so we can change without creating new object. StringBuffer is suitable for concurrent prog’g.

**@. Class? Class and Object** ----mdcci Ref: Java\_Examples\ReadOnly\_Class Vs & Ms – Variables & Methods

Class keyword is reserved in java. Class is created using class keyword. Class is a blue print or template and it is a model for creating object and object is an instance of class. Class describes attributes (Vs & Ms) and behaviour of object. Class is a collection of objects. It is a combination of variables and methods. All Java codes are defined inside the class. Class doesn’t occupy any memory space.

A**Variable** is a name of memory location. It acts as a container which holds (store) the value while java prog is executing. Value can assign to variable. Variables are created in the stack memory so that when the variable is out of scope those variables get garbage collected. The three main types of variables in Java are local, instance, and static:

**Local variables** are declared inside a method or block and these can only be accessed within that method or block.

**N:** **Can LVs have default value?** LVs are not initialized to any default value, neither primitive nor object references.

**Instance variables** (non-static variables) are declared inside a class but outside the method. These are associated with instance of the class.

**Static variables** are declared as static and these are shared among all instances of a class.

**N:** No variable can’t be static as well as local at the same time. Defining local variable as static it gives compilation error.

A **Method** is a program module used to perform some operations. It contains a set instruction to satisfy the particular requirement. A class can contain unlimited number of methods and each method can call unlimited number of times. Method is used to exhibits functionality of an object. There are 2 types of methods in java those are,

**Built-in / PD Methods:** Defined by the language (length(), equals()) and **User-defined Methods:** Defined by the user

**N:** Java provides Access modifiers to make **Read-only class**. By this we are providing only the get method and not providing any set method so that we can get the values of class members but we are not allowed to set the values.

**@. Private and Final methods in Java**

**Private:** Private methods are not accessible from outside of the class / from child class and these cannot be Overridden.

**Final:** When a method is marked as a final (final keyword before the method name) in the parent class, then any child class cannot override or modify the final method in java.

**N: Constant Variables** are declared as ‘**final’**, so during execution we can’t change their value once they are fixed.

**N:** To store data in a .properties file in Java, we need to use the java.util.Properties class.It is used to store project configuration data (like environment variables ex: Browser details, URLs, username & pass, etc.) in key and value pair format externally from code for easy updates and then initialize this file into setup methods. Common exceptions we have faced while working with properties file is: FileNotFoundExcep, NullPointerExcep, IOExcep.

**@. for loop and for-each loop (Enhanced for loop)**

* Both are used to executes a single stmt or block of statements repeatedly until a specified expr evaluates to false.
* If certain amount of iteration is known we can proceed with **for** loop.
* **Foreach** loop repeats a block of statements for each element in an array (nothing but object collection).
* **For** loop is more flexible and it does not necessarily involve an Array.
* The performance of normal **for** loop is slightly better because there is no function call to each element involved.

**N:** A for statement can loop indefinitely in Java if there is no terminating condition or a condition that always evaluates to true. **Ex:** For(;;) Ref: Java\_Examples\ForStatementCanLoopIndefinitly

**Nested for loop:** Loop inside another loop is present.

**@. Reflection in Java**

Reflection is an API which is used to examine or modify the behaviour of methods, classes or interfaces at runtime. It comes under java.lang.Reflect package. It provides classes like Class, Method, and Field to analyse and manipulate classes, methods, and fields. Reflection gives us information about the class to which object belongs and also the methods of that class.

**Ex:** In test automation frameworks Reflection is commonly used for loading classes and finding methods dynamically or parameterized test execution using Reflection. Ref: Java\_Examples2\Reflection\Parameterized\_Test\_Execution\_usingReflection

**@. Use of java.lang.Class**

The Class class in java is used to represent classes and interfaces during runtime. It provides methods to examine and access class metadata.

**@. Object-oriented approach? OOP languages and OBP languages** WC – Wapper Class

OOP approach is a programming methodology to design computer programs by using classes, interfaces, methods and variables. OOP languages follow all the features of OOP system (OOPs). C++, Java, python, etc. are examples for OOP languages. These are secure and reusable. Object based programming languages follow all the features of OOPs except Inheritance and Polymorphism concepts. For ex, JavaScript and VBScript (Visual basic).

**N:** Java is not 100% object oriented because PDTs (int, float….) are not objects and due to having the WCs. **N:** A pure OOP language treats everything as an object. Challenges of using OOP in java: OOP can make code more complex

**@. Pair Class**

Pair class is used to store the values in the form of <Key, Value> pair format by using parameterized constructor provided by the pair class. It is used in [HashMap and TreeMap](https://www.geeksforgeeks.org/hashmap-treemap-java/).

**Methods provided by the javafx.util.Pair class:**

* **Pair(key, value):**Creates a new pair
* **Boolean equals(object obj):**It is used to compare two pair objects. It compares on the basis of <Key, Value>
* **toString()**is used to return a string representation of an object. Ref: Java\_Examples\toString\_Ex
* **getKey()** returns key for the pair.
* **getValue()** returns value for the pair.
* **int hashCode()** returns an integer hash code value for an object.

**N:** An **API document** is a html file that contains description about all the features of a software or a product or a technology. API document is helpful for the users to understand how to use the software.

**@. #include and import statement**

#include directive makes the compiler go to C/C++ standard library, #include directive copy the entire code from header file contents that has to be compiled. As a result, the program size increases, thus wasting memory and processor’s time. Import statement makes the JVM go to Java standard library, import will paste the compiled output from the header file contents instead of entire code copied. execute the code here and substitute the result into the program. Here, no code is copied so no waste of memory and processor’s time. So, import is efficient mechanism than #include.

**@. print(), println() and printf() methods**

Both methods are used to display the results on the console. print() method displays the result and then retains the cursor in the same line to the end of the result. println() displays the result and then throws the cursor to the next line. With printf we can use format identifiers too.

**@. If ‘String[] args’ is not written in main() method / Significance of main method in java**

Main method is a starting point of execution for all java programs. JVM invokes main() to start the execution of java program. When main() method is written without ’String[] args’ i.e., Public static void main() only - The code will compile by the compiler but JVM cannot run the code because it cannot recognize the main() method from where it should start execution of the Java program. Remember JVM always looks for main() method with string type array as parameter.

**N:** If Main method is not containing ‘static’ we didn’t get any error the code will compile by the Compiler but JVM won’t execute the java program. If Main method is containing ‘static’ the JVM can call the Main method just by using the class name as reference.

(To call a method we require object, sometimes it may be required to call a method without the help of object. Then we declare that method as static. JVM calls the main() method without creating object by declaring keyword static)

**@. Unicode system**

Unicode is used for internal representation of characters and strings. Unicode system is an encoding standard that provides a unique number for every character no matter on what the platform or programming language using. Unicode uses 16 bits to represent a single character. For example, Unicode standard defines are UTF-8, UTF-16 & UTF-32.

**@. Control statements and it’s use**

Which alters the flow of execution of program and provide better control to the programmer on the flow of execution. They are useful to write better logic for complex programs. Java provides 3 types of control flow statements, those are:

i) Conditional / Decision Making statements: if-else, switch statements and ternary operators.

ii) Loop statements: while, do-while, for loop & for-each loop.

iii) Jump statements: break, continue and return.

**@. Out of do-while and while – – which loop is efficient**

In do-loop, the statements were executed without testing the condition on the first time. From the second time onwards, the condition will observe. In while-loop, first the condition will test and then only the statements were executed. Hence, while loop is more efficient than do-while loop. Depends upon the usage these loops will use.

**@. JAVA** ----ohpc

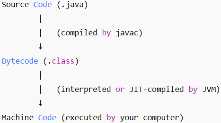
Java is object oriented, high-level programming language, platform independent and Java is a collection of objects. Java was originally developed by **James Gosling** at Sun Microsystems (later acquired by Oracle) in 1991 and released as a core component of Sun Microsystems’ Java platform. Lot of applications, websites and games that are developed using Java. There are three components are available in java those are:

Java SE (Java Standalone /Desktop Edition) **&&** Java EE (Java Enterprise / Web Edition) **&&** Java ME (Java Micro Edition)

**@. JAVA / JAVA API / Features in JAVA / OOPS concepts / Principles of OOP / 4 pillars of an OOPL / Y java is platform independent / W is byte code in java** ---- IEPA

Java is High-level, OOP Language. It is a collection of pre-built classes and packages that provides commonly used functionality and it has some features like: Variables and Data types, Basic operations, Conditional statements (if-else, ternary operators), Loops (for), Arrays, different classes & methods, Garbage collection, Exception Handling, etc.

**OOPS core concepts / principles:** Under OOPS concepts **I**nheritance, **E**ncapsulation, **P**olymorphism and **A**bstraction

* **High Performance:** **JIT** (Just In Time compiler) is a part of JRE. JIT enables high performance in Java. **JIT** runs after the program has started executing on the fly. It accesses runtime information and makes optimizations of the code for better performance. JIT converts the bytecode into machine code at runtime and then JVM starts the execution.
* **Platform independent:** Bcz of this feature a single program can works on different platforms without any modification.
* Reason (H Java works): When application is developed with java programming language the first step is java compiler (javac) takes java file and after successful compilation byte code/.class file is generated by java compiler. It is an intermediate code and platform independent we can’t execute directly on computers so JVM will converts bytecode into low level code which can executed on computer. Here java is platform independent but JVM is different for different platforms (i.e. platform dependent).
* .java file (Source code) **-->** byte code/.class file **-->** machine level code/low level code
* **Multithreading** **or Thread-based Multitasking** is a process in which java program is divided into multiple smaller parts called threads, these threads run parallelly and simultaneously, and allowing tasks to be performed concurrently. JVM creates a thread is called ‘main thread’. The user can create multiple threads **By Extending the thread class** or **By Implementing Runnable interface**.

**Use of Multithreading:** The main purpose of multithreading is to provide simultaneous execution of two or more parts of a program that can run concurrently. Generally, a multithreaded program contains two or more parts each part of a program called ‘thread’. Each thread represents an independent flow of execution.

**Main feature of java:** is platform independent and it can be achieved through the JVM.

**Interpreted:**

**@. Inheritance / Achieve / Use of inheritance**

Create a new class from existing class so that the new classes will acquire all the existing properties from parent class to child class with the help of ‘extends’ keyword is called Inheritance. Inheritance can be achieved using ‘extends’ keyword. **Adv:** For Method Overriding (so that Runtime polymorphism can be achieve), For Code Reusability. No Code Duplication and Simplifies updates.

**In FW level Inheritance** allow to share common functionalities (methods) across different classes by extending BC / PC.

**Ex:** ‘BaseTest’ manages browser setup(), teardown() and logging these are inherited by all tests.

**Limitations of using Inheritance:** Inheritance inherits everything from the super class and interface, it may make the subclass too clustering and sometimes error-prone when dynamic overriding or dynamic overloading in some situations.

**N:** **Private** members and **Constructors** are not inherited by child class because constructors are not considered as class members so they are not inherited by subclasses. But superclass constructor can be invoked from subclass by using super() keyword.

**@. Encapsulation / Data hiding**

Encapsulation is a mechanism for the process of binding data members and corresponding methods together into a single unit (i.e. class)*.* for ex, if we take a class, we write the variables and methods inside the class. That class is binding them together so, class and package is a good example for Encapsulation. Encapsulation is mainly used to hide data members with the help of private keyword. To implement Encapsulation, declare the Global variables as private so that this variable cannot be accessed directly by anyone from outside of the class. By providing public non-static (because, we cannot use ‘this’ keyword inside the static content) getter and setter methods in the class to access/get the values and to set/modify the values of the private variable/field. So that other methods can access getter & setter and getter & setter will access the private members. If we don’t define the setter methods in the class then the field will act as read-only and if we don’t define the getter methods in the class then the field will act as write-only.

**Adv:** The main purpose of Encapsulation is for data hiding (i.e. by preventing the other classes to access the private fields directly for security point of view or hacking prevention point of view), it makes the code readable and maintainable, Controlled Access (Provides validation via getter/setter methods).

**Dis-adv:** It increases the length of the code.

**In Framework level Encapsulation** is mostly used in POM to keep locators as private and expose only necessary actions for that allow access via public methods. and to create json request body in RestAssured API by POJO.

**Ex:** Login page methods like ‘enterUserName()’ and ‘clickLogin’ prevent direct locator access.

**@. Polymorphism ----** Ref: Java\_WD\_Ex\OOPS\_Polymorphism\Static\_nd\_Dynamic\_Bindings

**Polymorphism is a** feature to achieve different implementations with the same name. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. There are 2 types of polymorphism in Java those are Compile-time/Static polymorphism / Early/Static/Compile-time binding. Compile-time polymorphism can be achieved by Method Overloading and Constructor Overloading. Compile-time polymorphism means method invocation is decided by the Compiler and same gets executed by JVM at runtime or type of object is determined at compile-time. Because of early binding compile-time polymorphism provides fast execution. **R**untime / Dynamic polymorphism / Late/Dynamic/Runtime binding. Runtime polymorphism can be achieved by Method Ove**rr**iding and Upcasting. Runtime polymorphism means method invocation is decided by JVM (during the program execution) not by the Compiler or type of object is determined at run time. Because of late binding RTP provides slow execution compare to CTP. With the help of parameter Compile-time polymorphism itself it can identify. Depends upon the reference we will get the result in Runtime polymorphism. Runtime polymorphism is a process to call an Overridden method through the parent class reference. Binding is nothing but a calling the methods. **Adv:** Code reusability, flexibility and maintainability. For FW level use: In base class to launch the browser. (for more ref MOL and MOR section)

**N:** PC reference variable refers / store the object of CC is called **UC**. It is done by compiler automatically (implicitly). **Ex:** PC pr = new CC(); (implicitly). UC using in Framework: WebDriver driver = new FirefoxDriver();

**N:** CC reference variable refers / store the PC reference by typecasting CC is called **DC**. It is done programmatically (explicitly). **Ex:** CC ch = (CC)pr; (explicitly).

**Adv:** UC allows to write generic code, DC allows to access specific feature (methods & fields) of subclass. Both are used to implement polymorphism. --- Ref: Java\_Examples2\TypeCasting

**@. Early and late binding**

Early binding refers to the assignment of values to variables during design time, whereas late binding refers to the assignment of values to variables during runtime.

**@. Why goto statements are not available in Java**

goto statements leads confusion for a programmer. Especially in a large program, if several goto statements are used, the programmer will get confuse while understanding the program flow, from where to where the control is jumping.

**N:** If **Operator (+, -, \*, /, %…) Overloading** is possible in java the programmer will get confuse so it is not possible.

**@. On which memory, arrays are created & stored in Java**

Arrays are **created** in dynamic memory by JVM. Everything (it’s nothing but Arrays, Variables, Objects ….) is created in dynamic memory only there is no question of static memory in Java.

Arrays are **stored** in Heap space, because array is a class so we have to create object to that class, all the objects are stored in heap space and it is a dynamic memory.

**@. Memory management in java / Stack and Heap memory/area**

Java memory is divided into different regions in the Java Virtual Machine (JVM):

**Stack memory** is small and fixed-size area of memory to store Local Variables (like Primitive variables int, char, etc.) and method calls. It is allocated automatically when a method is called and it is deallocated when the method is returns. Memory access is fast.

**Heap memory** is large and dynamic-size area of memory to store objects and their instance variables. It is allocated dynamically when an object is created and it is deallocated when an object is no longer needed. It is managed by the Garbage Collector. Memory access is slow. **N:** LVs doesn’t store in Heap memory.

**Method Area (Part of JVM Memory)** – Stores: Class metadata, Static variables, Static methods, Constant pool

**Java IDE’s:** An IDE is nothing but a software application which enables users to more easily write and execute the programs. Many IDEs provide some features like syntax highlighting, error displaying, debugging….…etc which help the users to write code more easily. Eclipse, visual studio and NetBeans are the Java IDEs.

**@. Concrete class and Abstract class**

A **Concrete class** can be instantiated (means we can create object) and it requires to implement all the methods.

An **Abstract class** cannot be instantiated and at least one method has not been implemented.

**@. Abstraction**

Hiding the internal implementation details (complex logic) from the user and expose only the functionality is of interest of the user. The main idea behind abstraction is to reduce complexity and increase the efficiency of code. To achieve Abstraction, we have to use Abstract class or Interface. A good example for Abstraction is ATM machine: at ATM machine we can see only the functions (withdraw, deposit, balance enquiry….) but how ATM internally working is hidden from user. **Adv:** Allow easy browser switching, Reduces test script dependency.

**In FW level Abstraction Ex** is ‘WebDriverManager’ handles browser selection internally so Test Cases use ‘getDriver()', ListnerClass implements ItestListner{ }

**@. Java Collections Framework**

Java **Collections Framework** provides a set of interfaces and classes that helps in storing, processing, and manipulate the group of objects efficiently. It provides ready-to-use data structures like List, Set, Queue, and Map.

**@. Java Collections class**

Java **Collections class** provides utility methods for working with collections, such as sorting, searching, and shuffling.

**@. Collection, Collections and Collector** ---- Ref: Java\_Examples2\Collections\_Collection\_Collector, Java\_Examples\Collections\_Ex A - Array, AL - ArrayList

* Collection is a root interface of the Java ‘Collections Framework’ like List, Set, Queue, (excluding Map). This comes under java.util package. It is used to store the list of objects in a single object. Not all the methods inside Collection is static. Collection interface provide following methods to perform certain basic operations: add(), addAll()(Add all the elements in the specified collection to existing collection), clear(), contains(), containsAll(), isEmpty(), remove(), removeAll(), retainAll(), size(), toArray(), iterator(), spliterator(), stream(), parallelStream(). It is dynamic mem alloc.
* Collections is a utility class that comes under java.util package. It is used to operate on a collection. Collections has all static utility methods. Collections class provide following static utility methods for searching, sorting, and modifying collections: min(), max(), sort(), reverse(), addAll(), shuffle(), fill(), copy(), synchronizedCollection/List(to make array list as synchronized)/Set/Map, unmodifiableCollection/List/Set/Map, singleton().
* Collector is an interface under java.util.stream package used for collecting stream elements into collections like List, Set, or Map. i.e. to accumulate elements into a final structure like a List, Set, or Map. Common Collectors: toList(), toSet() (Collects elements into a Set), toMap(), joining() (Joins elements into a String)

**N:** A **read-only Collection** means, a Collection where we cannot perform modifications like add, remove or set. We can only read from the Collection by using get method or by using Iterator. Common Collection methods to manipulate, access and organize elements within collections are add(), remove(), clear(), toArray(), contains(), size() and iterator().

**@. Abstract Class** ------ cmv sdms co

Abstract class is used to achieve partial abstraction (i.e., 0(not mandatory to have AMs inside AC)-99% abstraction). An abstract class can have both Abstract methods and non-abstract methods. In Abstract class at least one method should be in abstract nature. By using extends keyword in child class we can acquire all properties from Abstract class. In selenium WebDriver ‘RemoteWebDriver’ is an abstract class. It implements the WebDriver interface. It acts as a base class for browser-specific drivers like ChromeDriver, FirefoxDriver, etc.. Rules for Abstract Class: ---- cmv sdms co

* Abstract class must be declared with an abstract keyword before the class name.
* Abstract method is must be declared with an abstract keyword in method signature.
* In Abstract class we can declare **any** kind of variables like static, non-static and final variables.
* Abstract class can have at least single abstract method, then that class will become abstract class.
* Abstract methods can declare with public, protected and default. And cannot be declared with private and final (because we have to Override AMs from subclass).
* Abstract class can have both abstract methods and non-abstract methods. ------ cmv sdms co
* Abstract methods cannot be declared as a static (because we have to Override abstract methods from subclass so Overriding is not possible with SMs and SMs must have method body).
* In Abstract class we can define Methods and Constructors (it doesn’t give any compilation error but we can’t instantiate Abstract class so there is no use of creating constructor for abstract class). --- Ref: OOPS\_Abstraction\AbstractClass\XExOnAbstractClass
* Object cannot be created to both Abstract class and Interface because these are reference types in java and Abstract methods are not fully implemented in superclass but Abstract methods can call by subclass object.

**Abstract method:** Is only prototype that means the method which has only declaration not implementation is called the “Abstract method”. Declarations are ends with semicolon.

**Variables & Constants:** The main difference between variables and constants is that variables can change their value at any time but constants never change their value. (i.e., the constants value is locked for the duration of the program execution). Pi for instance is a good example to declare a constant.

**@. Interface / Y we use Java Interface / H to declare an Interface** ----- mv ic Ref: OOPS\_Abstraction\Interface2

If one class extends more than one class then it is called Multiple inheritance. Multiple inheritance cannot be achieved in java to overcome this problem Interface concept was introduced. An interface is a template and it is a blueprint of class. An interface is declared by using the interface keyword. Interface is used to achieve full (100%) Abstraction, MI, Polymorphism and Loose Coupling. Here we can declare only abstract methods so that it has only method declarations not the method implementations so it cannot have a method body. While implementing Interface no need to write abstract keyword before the method name because interface keyword is used in interface signature so by default it will consider as all the methods in Interface is Abstract nature. By using implements keyword, we can override the abstract methods to child class. In real time we use interface concept at the creation of ‘WebDriver’ instance. Rules for interface:

* Abstract methods must be non-static inside the interface (these are by-default public abstract type). ----- mv ic
* We can declare only ‘static‘ and ‘default’ implemented methods inside the Interface. (reff^)
* In Interface we can declare public, static and final type variables.
* Constructors are not allowed inside an Interface.

**LC** means reducing dependencies of a class that use different class directly (the classes are independent to each other)

**TC** means classes and objects are depending on one another. Ref: Java\_Examples2\Coupling

Dis-adv. of Tightly coupled code is **it takes huge effort** & **testing time** so is reduced by writing loosely coupled code.

**N:** Default methods are allowing from java 8 and these allow to declare concrete methods in Interface. These allow us to add new functionality to existing interfaces without breaking the code of existing implementations.

**N:** Static methods are allowing from java 8 and these allow to declare static methods in Interface. These methods are not inherited by implementing classes and this can only be accessed using the Interface name.

**N:** An Interface cannot extends’ a class because Interfaces is in abstract nature and classes is in concrete nature.

**N:** If two Interfaces have same method name and signature and class implements both Interfaces then ‘the method must be overridden in the implementing class’. Ref: OOPS\_Abstraction/Why\_Interface\_Intraduced/TwoInterfacesHaveSameMethodName\_nd\_ClassImplementsBothInterfaces

**@. Constructor** ----- Nrao Ref: Constructor\AA\_Types\_of\_Constructors / Default\_nd\_ParameterizedConstructor

Constructor is a block of code similar to method it has the same name as class name. Constructor is called (invoked) by just creating object to the class. Constructors is used to initialize the objects or state of an object or initialize non-static members inside the objects (Ex: Initialize the elements in POM design pattern). There are 2 types of constructors in java those are Default Constructor (doesn’t not have any parameter and it is implicitly created by java compiler in .class file every java class have a default constructor if no other constructor is explicitly defined and these can have package level access) && User Defined Constructor (Parameterized (will have one or more parameters. It is explicitly defined by the programmer. It is mainly used to initialize instance variables to specific values at the time of object creation so that we can pass different values to a parameterized constructor at the time of object creation, but values should match the constructor signature) & non-parameterized (created by user)). && Copy Constructor (creates a new object by copying the properties of an existing object. It is used to copy the details from another constructor. It takes an instance of same class as an argument and initializes the new object with the same values.Ref:ConstructorConcept\Copy\_Constructor) && Private constructor Some other constructor properties are: (Ref: API FW api.utility/XLUtility class).

* Constructors are always non-static (So that Constructor is invoked by just creating object to the class, so no use of the static constructor)
* Constructor doesn’t return any value but does not accept the void in method signature. (Reason: Constructor is not directly called by code. It is called by object initialization at runtime)
* Constructor allow public, private, default, protected Access modifiers.
* A constructor can Overload but cannot Overridden (not inherited (Bcz Constructors are not considered as members and Constructor name is same name as class name)).

**@. Singleton class** Ref: Constructor\Singleton\_Class\SingletonDemo1 **/** LazySingletonClass

A private Constructor (prevents external instantiation), private static instance variable (holds the single object) and public static method (provides access to the instance) is/ are the basic components used to create a Singleton class (A Singleton class is a specific class that implements by the singleton design pattern that allows to create only one instance and provides a global point of access to that instance). It can be accessed within the class itself. Private constructor is used if we don’t want other classes to instantiate the object and to prevent sub classing.

**Adv:**

* Saves memory by avoiding unnecessarily object creation.
* Provides global access to an instance.
* Useful for classes managing shared resources (e.g., DB connection, Logger, WebDriver in Selenium).

**Dis-adv:** Global state can lead to unexpected side-effects in testing or multi-threading.

**N:** Before code compilation .java file after code compilation .class file.

**Instantiate/Instantiation:** To call any method of any class we need to creates an instance or object of that Class is called ‘Instantiation’. Instantiation allocates initial memory for the object and returns a reference.

**@. Constructor and Method / H Constructor is different from Methods-----** oirdnf

* Constructor is used to initialize the state of an object where method is used to perform actions (i.e. to expose functionality of an object).
* Constructor is invoked implicitly whereas m**e**thod is invoked **e**xplicitly (i.e., with the obj ref we can access method).
* Constructor doesn’t return any value where the method may or may not return the value.
* In case constructor is not present, if require a default constructor (hidden constructor) is provided by java compiler. In case of method, no default method is provided.
* Constructor should be the same name as class name. Method name should not be the same name as class name.
* Constructor cannot be marked as a static, final, abstract whereas method can be marked as a static, final, abstract.

**@. Abstract class and Interface** ------ kai edm bvm pim CC-concrete class, AC – abstract class

|  |  |
| --- | --- |
| **Abstract Class** | **Interface** |
| Abstract class is created using abstract keyword | Interface is created using interface keyword |
| Abstract class is used to achieve partial (0-100%) abstraction | Interface is used to achieve Full (100%) abstraction |
| MI is not supported through Abstract class | MI is supported through interface |
| Abstract class can extend only one CC or one AC at a time | Interface can extend any number of interfaces at a time |
| Abstract class have a default constructor. | Interface doesn’t have any default constructor. |
| It Contains both Abstract and non-Abstract (concrete) mtds | It Contains only Abstract methods. |
| Abstract class can have static and non-static initializer blocks | Interface can’t have static or non-static initializer blocks |
| Abstract Class can have instance variables. The variables of an Abstract class contain static, non-static and final variables | Interface can’t have instance variables. The variables of an interface contain default public, ‘static’ and ‘final’ variables. |
| The class which implements the Abstract class should not require to implement all the methods, only non-Abstract methods need to be implemented in the concrete subclass. | The class which implements the interface should not require the implement all the methods. |
| Performance wise it is faster than interface kai edm bvm pim | Performance wise it is slower than Abstract class because in interface if we try to access any method first will check where exactly that particular method is available inside the interface (because in interface method declaration in one class and method implementation in another class) |
| It cannot be instantiated (means object cannot be created) | It cannot be instantiated (means object cannot be created) |
| Abstract method can declare with public and protected | Class members are public by default in interface |

**@. Types of variables** Ref: Java\_Examples\TypesOf\_Variables

**Local var’s** are **d**efined inside the method, constructor, or block and these can access within that scope. Accessible only within the method or block where declared. Must be initialized before use. Ex: int n=90;

**Instance variables (non-static):** Class variables without a static kw is known as IVs. These are defined inside a class but outside any method. Belong to each object of the class. **IVs** are Created when the object is created, and destroyed when the object is destroyed. **IVs** can access from anywhere by using object reference. Ex: String instanceVar = “abcd”;

**Class variables (static variables))** are declared with the static keyword. SVs are **d**efined inside the class and outside the method, constructor, or block and these shared among all objects of the class i.e. these can access all methods, constructors, or blocks of the class. SVs belongs to the class, not to any object. Created once when the class is loaded into memory. **SVs** can access from anywhere by using class reference.

Ex: public static int staticVar = 90; ref for below N: Java\_Examples/VariableHiding

**N:** A variable can’t be local as well as static at a time bcz by defining a local variable as static it gives compile-time error.

**@. Where Encapsulation and Abstraction used**

Encapsulation implemented in POM by keeping WebElements and methods to interact with them within the page class.

Abstract class or Interface define common behaviours that are implemented in concreate classes and hiding complex implementation from the user.

**@. Method Overloading / Constructor Overloading / Ways to achieve Overloading**---- npn’s

Inside a class more than one method having the same name with Different parameters or argument list is called Method Overloading. Best example of method overloading is we will use implicit wait to make the page wait for some specific time interval here we can provide different timestamp like SECONDS, MINUTES.…. and in selenium TestNG Assert class i.e., Assert.assert(Boolean\_cond, String\_msg); or Assert.assert(Boolean\_cond); from this both method names are same but different parameters. Following are the mandatory conditions for Method Overloading:

* Same method name
* Different parameters or arguments in signature (i.e. change in datatype)
* Number of arguments is different
* Sequence of arguments is different

And if both methods follow the above mandatory rules, then they may or may not:

* Have different access modifiers.
* Throw different checked or unchecked exceptions.

**Adv:** If we perform different operations using same method name then it will improve the reliability, readability, flexibility of program and reduce the code duplication.

**N:** Method Overloading is not possible by changing the return type in method signature because method name and parameters are same so there will confuse while passing the parameters.

**@. Method Overriding** Ref: OOPS\_Polymorphism\RuntimeP\_or\_MethodOverridingE5**\**E6\_RealTimeEx

Overriding a parent class method in child class and provide its own implementation to an inherited method without modifying the parent class code is called Method Overriding. A method can Override but data members Overriding is not possible. In selenium we use this concept by creating the WD driver = new ChromeDriver(); (RTP concept) then only we can execute the script on any browser. In framework level it is useful: Overriding a capturesceen() in onTestFailure() on ExtentReportManager class (Ref: Openkartv121/utilities/ExtentReportManager). We have to follow some rules to Override a method:

* We have to use exact method declaration as it is in child class (name, para’s, return type, static & non-static pov)
* We can override only non-static methods
* Relationship is important (i.e., extends: kw is used to establish an inheritance relationship between classes)

**Adv:** A class can give its own implementation to an inherited method without modifying the parent class code, used to achieve Runtime polymorphism, code reusability and code maintainability, to establish parent-child relationship.

**N:** **private, static** and **final** methods Overriding is not possible from child class.

**In Framework level Method Overriding** is mostly used to Overriding the ‘findElement’ Method, Overriding the ‘setUp’ Method. Ref: OOPS\_Polymorphism\RuntimeP\_or\_MethodOverridingE5**\**E6\_RealTimeEx

**@. Method Visibility in java /** **AMs: Public, Private, Default and Protected** ---- Reff: Java\_WD\_Ex\AccessModifiers <https://youtu.be/dkUSSdUpmyY?si=A0DzFJW1DDZgnIVG> --- 30:02

Methods and variables are known as class **members**. MV is nothing but accessibility of methods from another classes.

**Public** members are **v**isible in the same package as well as outside the package (i.e., other packages).

**Private** members are **v**isible in the same class only and not for the other classes in same package as well as classes in outside the packages.

**Default** members are declared in a class without any AM’s. These members are **v**isible with in the same package (include all classes in package) only.

**Protected** members are **v**isible with in the package (include all classes in package) and outside the package subclasses (of package) only can access.

**@. Multiple Inheritance is not supported through class in java, but H it is possible by an interface / Diamond Problem**

If one class inherits the properties and behaviour of more than one class (there will be compile-time error and priority problem) then it is called **MI**. If multiple parent classes have same method names and arguments then priority problem will occur so JVM gets confuse which method has to call and it leads to ambiguity. So, because of this diamond problem Java does not allow MI through classes to reduce complexity and maintain simplify the language. However, it is supported in case of an interface because method implementation is provided by the implementation class so, there is no ambiguity. Ref for **priority** **problem**: OOPS\_Abstraction/Why\_Interface\_Intraduced/TwoInterfacesHaveSameMethodName\_nd\_ClassImplementsBothInterfaces

**@. Wr Multiple Inheritance use in our automation framework** Ref: Java\_WD\_Ex\OOPS\_Inheritance\Multiple\_Inheritance\MI\_use\_In\_our\_AutomationFW1/2/3/4

1. To use Reusable Utility Interfaces (so that provide different reusable methods across the framework)

2. Page Object Model with Utility Methods (In POM we can use multiple interfaces to define common functionalities for page actions, validation utilities, and reporting).

3. Implementing TestNG Listeners with Multiple Inheritance

4. Creating a Hybrid Framework (Keyword + Data-Driven)

**@. Array and ArrayList / Array and Collection ------** nL his pm

### Array and Collection are somewhat similar regarding storing the references of objects and manipulating the data but they differ in many ways. Collection framework provides a set of classes or interfaces to store and manipulating the collection objects. It includes Map, List, Set and Queues. The main diff bn the Array and Collection are defined as:

|  |  |
| --- | --- |
| **Array** | **ArrayList** |
| Arrays can store similar type objects (having same data type). These are in static nature. ------- nL his pm | Collection can store different type objects (having different data type). These are in dynamic nature. |
| Arrays are fixed length data structures so we can’t change their size once they are created i.e., Size should be given at the time of array declaration.  **Ex:** String[] name = new String[2] | ArrayList size is automatically increase if we add elements beyond its capability i.e., Size may not be required. It changes the size dynamically.  **Ex:** ArrayList<String> name = new ArrayList<String>(); |
| In Java Arrays can hold primitive (Int, Char, Float, etc.) and non-primitive data types (Object). | ArrayList can hold only non-primitive data types (Object)  ------ Java\_Examples\Iterator |
| **for** or **for each** loop will use to iterate an array | Iterator, for, for-each will use to iterate an ArrayList |
| Size of Array can check by using **length** attribute | Size of ArrayList can check by using **size()** attribute |
| To put an object into array we need to specify the index. **@Ex:** name[1] = “book” | No need to specify the index instead of this we simply use **add()** method **@Ex:** name.add(“book”) |
| Array is multi-dimensional (Ref: XLUtility class) | ArrayList is single-dimensional |
| Array allow duplicate values | AL allow duplicate values and these are in insertion order |

**Ways to initialize Array:** Array[] arr = new Array[size]; Array arr[] = {val1, val2….};

**Adv and Dis-adv of Array:** Adv: In a single variable we can store multiple values. Dis-adv: supports only similar DTs, size fixed at compile-time, memory wastage is high.

**N:** Arrays cannot provide any readymade methods for user requirements as searching, sorting, manipulation, insertion, deletion, etc. but Collection includes readymade methods to use.

**Sorted Array:** Array which contains duplicate elements is called Sorted Array.

**@. HashSet, HashMap, TreeSet, TreeMap and HashTable** ------- Ref: Java\_Examples/Hashset\_HashMap

|  |  |
| --- | --- |
| **HashSet** | **HashMap** |
| HashSet is an implementation of Set interface | HashMap is an implementation of Map interface |
| HashSet stores the data in the form of objects | HashMap stores the data in the form of key-value pairs |
| To put data into HashSet we need to use .add() | To put data into HashMap we need to use .put(k, v) |
| HashSet internally uses HashMap to store its elements | HashMap internally uses an array of entry <K, V> objects |
| HashSet doesn’t allow **duplicate** elements | HM allow duplicate values, but doesn’t allow duplicate keys |
| HashSet allow one **null** element | HM allow one <null key, null value> and multiple null values |

**N:** HM and HT are used to store elements in the form of key-value pairs, HM is not synchronized and allows null keys and values, while HT is synchronized but doesn’t allow null keys or values.

**TreeSet** is an implementation of Set interface that uses a Red-Black tree for storage. It provides ele’s in sorted order.

**@. Enumerator, Iterator and ListIterator**

Enumerator is older and it is used to iterate only legacy class and interface. It is there from JDK1.0 and Iterator was introduced later. Iterator can be used to iterate collection classes (List, Set and maps) to traverse the objects from collection. By iterator we can read and remove the elements while traversing the collection. By Enumerator we can only read the elements while traversing the collection. Only forward direction iteration is possible with both Enumerator and iterator. Iterator is an interface and it is more secure and safe as compared to Enumeration because it does not allow other thread to modify the collection of objects while some thread is iterating over it, it throws ‘ConcurrentModificationExcp’. Iterator iterate by using HashMap.keySet() and HashMap.entrySet(). ListIterator is an interface used for iterating List type classes. Here remove method is available. Both forward & Backward dir’s are poss.

**Avl mtds in Iterator:** hasNext(), next(), remove() && ListIterator - hasNext(), next(), remove(), hasPrevious(), previous()

**N:** In Java, an **enum** (short for enumeration) is a special data type used to define fixed set of constants and used to call an enumeration. enum can have methods, constructors and Instance variables. It is more powerful than using traditional constants. It provide features like type safety, iteration, and also used to represent a collection of predefined values, such as days of the week, directions, or status codes. Ref: Java\_Examples2\EnumEx

**Interfaces:** List, Set, Map, Queue

**Classes:** AL, LL, HS, LHS, TS, HM, LHM, TM, PriorityQueue, Deque.

**Utility Classes:** Collections (searching, sorting, synchronizing), Arrays (for Array-to-list conversions, sorting)

**@. Classes and Interfaces are avl in the Collection FW / Interfaces in the Collection FW / Java collections hierarchy**

**d**At the top level, we have two root interfaces in java.util:

1. Collection<E> (represents a group of elements)

2. Map<K,V> (represents k-v pairs, not a true subtype of Collection)

**Types of Classes:** Normal /Concrete classes and Abstract classes

**Types of Interfaces:** Normal, Functional & Marker Interface

**N: Priority Queue** Implements Queue Interface.It is used to hold the elements in the collection and it follows the **FIFO** (First-In-First-Out) principle to inserting elements at the end of the list and deleting elements from the start of the list.

**N: Vector** is synchronized and thread safe. ArrayList is not synchronized and not thread safe. Vector class is used to provides the ability to implement a growable array of objects.

**@. ArrayList and LinkedList** Ref: Java\_Examples\ArrayList\_LinkedList\_Diff / Java\_Examples2\LinkedListEx / Java\_Examples\Reverse\_LinkedList

An **ArrayList** internally uses dynamic Array to store the elements. ArrayList allows fast storing/searching/retrieving the elements using indexes but slower insertion (at beginning) and deletion of elements. And fast insertion at end.

A **LinkedList** is implemented from doubly LinkedList to store the elements. LinkedList allows fast insertion (at end & beginning) and deletion but slower on storing/searching/retrieving the elements.

**N:** ArrayList is a better choice if our prog. is thread-safe. Vector & ArrayList req. more space when more ele’s are added.

**@. Ordered, Sorted and Contrast in collections**

An **Ordered collection** means the elements of collection have a specific order. The order is independent of the value of the element. **Ex:** [List](https://docs.oracle.com/javase/7/docs/api/java/util/List.html).

A **Sorted collection** means the elements of collection have a specific order. The order is dependent on the value of the element. **Ex:** [Sorted Set](https://docs.oracle.com/javase/8/docs/api/java/util/SortedSet.html).

A **Contrast** means the elements of collection have **without any order**. **Ex:** [Set](http://docs.oracle.com/javase/7/docs/api/java/util/Set.html).

**@. Exception**

An Exception is a problem that can occur during the normal flow of execution. When something wails at runtime the method throws an exception, if couldn’t handle that exception, the execution gets terminated before it completes the task. If we handled the exception, the normal flow gets continued. Exceptions are the subclass of Throwable classes.

**@. Math class**

**PI** is a built-in math function that's part of the overall Math class. Math classes are included in java.lang.Math.\* package. Math class in java provides various methods for performing mathematical operations such as square root, trigonometric functions, exponential calculations, rounding and more. (Ex: Math.pow(i, n), Math.sqrt()…)

Math.randon() returns a random double value between 0.0 (inclusive) and 1.0 (exclusive). It is often used to generate random numbers or to simulate random behaviour.

**@. Comparable and Comparator Interface ----** Ref: Java\_Examples2\SortArrayListUsing\_Comparable\_nd\_Comparator

|  |  |
| --- | --- |
| **Comparable Interface** | **Comparator Interface** |
| Comparable define natural ordering | Comparator define custom ordering |
| Comparable interface provides single way of sorting | Comparator interface provides different ways of sorting |
| Comparable allows to sort the collection on the basis of a single element such as id or name etc | Comparator allows to sort the collection on the basis of multiple elements such as id and name etc |
| Actual class is modified with Comparable | Actual class doesn't modify with Comparator |
| It comes under java.lang package | It comes under java.util package |
| It overrides compareTo() method to sort elements | It overrides compare() method to sort elements |

**@. Types of Exceptions**

While code compiling time or code executing time there are two types of Exceptions will occur in java those are.

**Checked Exception:** Checked Exceptions are coming under user control that means user has to verify during the coding. These types of exceptions can handle by the Compiler with the help of try and catch blocks.

**Ex:** IOException, ClassNotFound Exception, FileNotFoundException ------ Compile time Excp’s

**Unchecked Exception** (Runtime Exceptions)**:** Unchecked Exceptions are not known until they are executed. These types of exceptions can handle by JVM, so we have to correct them programmatically.

**Ex:** ArithmeticExcp, ArrayIndexOutOfBoundsExcp, ClasscastExcp, DivideByZeroExcp, NullPointerExcp --- Run time Excp’s

**@. Can we use catch statement for checked exceptions** Ref: Java\_Examples2\ExceptionHandling\UsingCatchStatementForCheckedExceptions

Yes, we can use a catch statement or declare with throws for checked exceptions in Java.

**@. Ex for java and WebDriver Exceptions / Built-in Exceptions** ---- https://www.geeksforgeeks.org/built-exceptions-java-examples/

There are two categories of exceptions will occur in java those are **Built-in** and **User-defined Exceptions.** AllChecked (are checked during compile time) and Unchecked (are thrown during run time) excp’s are comes under Built-in excp’s.

**ArithmeticException** is thrown to indicate that when an exception has occurred in an arithmetic operation.

**ArrayIndexOutOfBoundsException** is thrown to indicate that an array has been accessed with an illegal index. i.e., the index is either negative or greater than or equal to the size of the array.

**ClassNotFoundException** is thrown when we try to access a class whose definition is not found

**ElementNotVisibleExcp**: Although element is present on the DOM but not visible on the page so we are not able to interacted with that element. It occurs due to the element is hidden or overlapped by another element, out of viewport, or timing issue.

**ElementNotSelectableException:** Although element present on the DOM but cannot be selected. It occurs due to the element is Disabled (disabled attribute is set). It occurs due to dynamic changes. If an element is hidden (display: none or visibility: hidden), it typically causes ElementNotInteractableException, not ElementNotSelectableException.

**ElementNotVisibleExcp (**Deprecated -> **ElementNotInteractableExcp** (Selenium 3+)**):** When an element is present in DOM but not visible or not interactable with (Ex: Trying to click an element that is hidden, typed into) at the moment by hidden or overlapped. Fix: Wait for element visibility, or use JS click. Debug: Check display none or hidden CSS prop’s.

**ElementNotClickableException:** Although element is present on the page but we are not able to click on that element due to the page loading issue, overlapped, or out of viewport. Most of the times it can handle by refreshing the page.

**FileNotFoundException** is thrown to indicate that when a file is not accessible or does not open.

**IOException** is thrown to indicate that when an input-output operation failed or interrupted. (Ex: FileNotFOundExcp, MalformedUrlExcp, NotSearializableExcp)

**InterruptedException** is thrown when a thread is waiting, sleeping or doing some processing it’s get interrupted.

**IllegalArgumentException** is thrown to indicate that when an argument is illegal for Ex Thread.sleep(-1000);

**NoSuchFieldException** is thrown to indicate that when a class does not contain the specified field (or variable)

**NoSuchMethodException** is thrown to indicate that when accessing a method which is not found.

**NoSuchElementExcp** is thrown to indicate that when findElement() can’t able to find the element using specified locator. Fix: Add Explicit waits, or verify locator accuracy (Debug tip: use browser tools).

**NullPointerException** will occur when accessing or modifying the null object reference. Null rep. nothing.

**NumberFormatException** is raised when a method could not convert a string into a numeric format.

**RuntimeException:** Any exception which occurs during runtime (except **IllegalArgument** above all are RT exceptions).

**StringIndexOutOfBoundsException** is thrown by the String class methods to indicate that an index is either negative than the size of the string.

**StaleElementExcp / StaleElementReferenceExcp:** There are two causes for this exception. Element in the web page that was a part of the current instance has been refreshed or navigated to another web page so that element is destroyed or reconstructed. The second reason is that the element is no longer attached to the DOM. Ref: WD\_Examples/EX23\_Synchronization\_options

**To handle this**, refresh the webpage using the navigate command **or** use a try-catch block within a for loop if element is not attached to the DOM **or** Another option is to use WebDriverWait with ExpectedConditions. There is a specific ExpectedCondition called stalenessOf(), which allows to wait for the staleness of an element to disappear before starting interaction with the WE. **Ex:** wait.until(ExpectedConditions.refreshed(ExpectedConditions.stalenessOf(ele)));.

We can handle the StaleElementReferenceException by:

* Re-locating the element before interacting with it (i.e. Re-locate the element inside a loop)
* Using WebDriverWait with ExpectedConditions

Ex: ExpectedConditions.refreshed(ExpectedConditions.visibilityOfElementLocated(By.id("dynamicElement"))));

* Implementing custom retry logic
* Using try-catch blocks to catch and handle the exception

**N:** The Document Object Model (**DOM**) is a programming API for HTML and XML documents. It defines the logical structure of documents and way of document is accessed and manipulated.

**TimeoutException** is thrown when a expected condition is not met within the specified time or command did not complete in enough time. Fix: Increase the wait duration, or Refine the condition (Ex: Wait for clickable instead of just visible).

**User-Defined / Custom Exceptions**: The following steps are followed to the creation of User-defined Exception.

* First, create a Class that extends **Exception** class and **Exception** class is a sub class of Throwable class.
* Create a parameterized constructor to take string as a parameter to store the exception details and call super class constructor from this and send the string parameter. Ref: Java\_Examples\UserDefinedException
* Finally, to raise exception as user-defined type, create an object to our excp class and throw it using throw clause.

**@. Advantages of Exception Handling -----** nct

* If Exception got handled the normal flow of the execution won’t terminated.
* We can identify the problem by using **catch** declaration with the help of **e.printstacktrace()** method.
* Think of situation that means, if we got any exception, we want to print some custom message in our logs so that, it can be understandable by the whole team.

**@. Exception Handling / Exception Handling Keywords / Handle Exception / Ways to Handle Exceptions**

**Exception Handling** is the process of handling the **Exceptions** when a program runs so that the normal flow of the application can be maintained. Java provides object-oriented way to handle the exception scenarios**.**

**throw Keyword** is used to throw an exception explicitly (create an object to our exception class and throw it using throw clause) from a method. It is used inside the method. It can throw only one exception at a time.

**throws clause is used to** throw an exception in runtime and handle it. It is used in method signature. If an exception occurs in a method, the process of creating the exception object and handling it to runtime environment is called **“Throwing the Exception”** and java run time system starts processing to handle them.

**🡪** At the end of the method signature, we declare the Exception type using ‘throws’ keyword. By declaring this, the caller program knows the exceptions is thrown by the method. Here we can declare multiple exceptions in this throws clause and it can be used with main method as well as sub method also.

**By using try-catch:** A method catches an exception using the combination of the **try**and **catch**keywords. try is the start of the block and Catch is at the end of try block to handle the exceptions. A risky code is surrounded by try block and If an exception has occurred catch block catches the exception thrown by the preceding try block. try can be followed either by catch (or) finally (or) both. But any one of the blocks is mandatory. Must and should **catch** is followed by try block. **Perfect Ex for try catch usage:** Ref: WD\_Examples\EX32c\_Check\_If\_ele\_Is\_Inside\_iframe\_r\_not\_Using\_tryCatch

**🡪** A try block doesn’t necessarily need a catch block. It can be followed by either a catch block, a finally block, or both.

**🡪**A single **try** block can be followed by multiple **catch** blocks for Ex, if there are multiple exceptions **or** we can expect more than one type of exception in a single block of code. There is no restriction on the number of the catch blocks. We cannot have multiple try blocks, there can be only one try block before catch/finally block but we can have try block inside try block followed by the catch block.

**🡪** Working of multiple catch blocks is pretty simple, if an exception occurs in the protected code, the exception is thrown to the first catch block in the list. If the **exception thrown** matches with the **ExceptionType1**, it gets caught there and executes the code which is under the same exception block. If not match, the exception passes down to the second catch statement and goes on like this.

**🡪**In case, the exception does not match with any **Exception type** after falls through all catch blocks, the current method stops the execution and will throw the exception. That’s why it is advisable to include **Default Exception** at the end so, in case if the exception falls through it, it can be handled by the default one.

**🡪**We can catch more than one exception in a single catch block. Ref: Java\_Examples2\ExceptionHandling\CatchMoreThanOneExcep\_in\_a\_SingleCatchBlock

**🡪**We can rethrow the checked exception from catch handler. For that we need to declare that exception Ref: Java\_Examples2/ExceptionHandling/Rethrow\_TheSameExceptionFrom\_catchHandler

**finally:** This is followed either by try block or catch block. Inside of this block statement gets executed regardless of an exception. So generally, clean up codes are provided here. It is used for cleaning up resources such as closing connections & sockets, quitting WebDriver. We cannot have multiple finally blocks in a single try-catch structure. If an exception is thrown from finally block it will throw RunTimeExcption. Finally block will executed even if no catch block handles the exception. Finally block will not be executed whenever JVM shutdowns, or if we use System.exit(0) in try statement. Ref: Java\_Examples2\ExceptionHandling\ThrowingExceptionFrom\_Finally\_block

**N:** A catch block without an exception type catches all types of exceptions. It generally used as a final catch-all to handle any unanticipated exceptions.

**@. Explain situation where finally block statement will not be executed**

If finally block is present it will not be executed whenever JVM shutdown (use System.exit(0); in try statement)

**N: Try-with-Resources** statement is used to close resources (like file streams, DB connections) automatically after use or when they are no longer needed or that are opened in try block. Ref**:** Java\_Examples2\ExceptionHandling\try\_wyth\_resources

**@. Explicit and Implicit Exceptions**

|  |  |
| --- | --- |
| Explicit Exceptions are thrown once explicitly checked by JVM **Ex:** ArrayIndexOutofBondExcep (eve Array indexing ope is ex) | Runtime throwing exceptions are called Implicit excp’s **Ex:** NullPointerException & StackOverflowError. |

**@. Exception Propagation**

When an exception occurs, it tries to locate the matching catch block, if the matching catch block is located, then that block is executed. Otherwise the Exception propagates through the method call stack and goes into the caller method. This happens until the matching catch block is found. In case the match is not found, the program gets terminated in main method.

**Call stack** is a mechanism for interpreter (like JavaScrit interpreter in a web browser) to keep track of its place in a script and that calls mul [fun’s](https://developer.mozilla.org/en-US/docs/Glossary/function) (to know that what fun is currently being run and what fun’s are called from within that fun, etc).

**@. Final keywords** ----- Java\_Examples/Final\_Variable\_Initializations

final keyword is used to create classes, methods and variables to restrict the modification of these. It can be applied to classes to preventing the sub classing, methods to preventing overriding and variables to create constants.

**Final variable:** If a variable is marked as a final then the value of the variable could not be changed. It’s like a constant.

**Ex:** final int n = 12;

**Final method:** If method is marked as a final (final keyword in method sig), then it cannot be Overridden by subclasses.

**Final class:** If class is marked as a final, then it cannot be extended by other classes (when we want to restrict a class from being extended then we make class as final).

**@. Thread / To make a thread in Java / Runnable interface / To implement Runnable interface** Ref: Java\_Examples2\Thread

In Java, the flow of execution is called **Thread**. Every java prog has at least one thread called MT it is created by JVM. The user can create their own threads in 2 ways those are **By** **ETC** (or) **By IRI**. Threads are executed concurrently.

|  |  |
| --- | --- |
| **a) By Extending Thread Class:** Stepsto ETC | **b) By Implementing Runnable Interface:** Steps to IRI |
| * Create a class that extends Thread class * Override the run() method and provide own implementation to this method. This is executed by thread. The run() method must be public while overriding. * Create an obj to main class in the main() mtd. * Call the start() mtd with main class object (c\_n). | * Create a class that implements Runnable interface. * Override the run() method and provide own implementation to this method. This is executed by thread. The run() method must be public while overriding. * Create the object to main class in the main() mtd. * Create the object to Thread class by passing above main class object as parameter to the Thread class constructor. * Call the start() method with ‘Thread Class’ object (Thread). |

**N:** The main purpose of Thread class is used to create and control the execution of threads. It provides methods for thread initialization, starting, sleeping and synchronization.

**@. Process and Thread**

A process is an independent program with its own memory space.

A thread is the smallest unit of execution within a process, sharing the same memory space.

**N:** A **join()** method is used to wait for a thread to finish its execution before moving on the next task. Ref: Java\_Examples2\Thread\join\_mtd

**@. yield() method of the Thread class do**

yield() method moves the currently running thread or thread back to a runnable state. It allows other threads for execution. Here thread is not sleep(), wait() or block() so that, the threads have equal priority chance to run. It is a static method and it doesn’t release any lock.

**@. To stop a thread in java / sleep (), wait () and block () methods in a thread**

We can stop a thread in java by using sleep (), wait () and block () methods

**sleep()** is used to **sleep** the currently executing thread for the given amount of time. Once the thread is wakeup it moves to runnable state. Ex: **Thread.sleep(2000)**

**wait()** is used to make the thread to wait in the waiting pool. When wait() method is executed during a thread execution wait() method doesn’t give the lock on the object immediately until the currently executing thread complete its task. after completes, immediately the thread gives the lock on the object and goes to the waiting pool and it is mostly used in synchronization. This thread will wake up after we called through notify() (or) notifyAll() methods.

**notify()** is used to send a signal to wake up a single thread in the waiting pool.

**notifyAll()** is used to send a signal to wake up all the threads in the waiting pool.

**N:** The major difference is wait() releases the lock and sleep() doesn’t releases any lock while waiting. Generally, wait() is used for inter-thread communication while sleep() is used just to pause the execution.

**Block():** Which blocks the executing thread until their operation finished.

**@. start() and run() method of thread class**

start() method creates new thread and run() method is executed on newly created thread. If we directly call the run() mtd, the new thread is not create and the currently executing thread will continue to execute the code inside run() mtd.

**@. Synchronization in java &&&& Disadvantage of Synchronization ----** Ref: Java\_Examples2\Synchronizatio\_In\_Java

Synchronization is helps to control or prevent the access of multiple threads simultaneously so that one thread can execute a block of code or method at a time. It is implemented by using synchronized keyword. It is used to declare the block of code is thread-safe (it means that only one thread can execute the single block of code at a time) and also used to prevent thread interference, data consistency issues. The Dis-adv of Synchronization is, it not recommended to implement all the methods because of it, if one thread accesses the synchronized code and next thread should have to wait so, it makes slow perf on the other end. There are 2 ways we can do synchronization in java **a)** Synchronized method (are used to control access to an object) **b)** Synchronized blocks.

**N:** If few lines of code required synchronization then it is recommended to use **Synchronized Blocks**. The main adv over Synchronized methods of this is reduce the waiting time of threads and improve performance of the system.

**N:** If multiple threads need to access and modify a shared resource (like variable, a collection, or a file) then it is recommended to use **Synchronized Methods**.

**@. Synchronized Methods and Synchronized Statements**

Synchronized Methods are methods that are used to control access to an object. A Synchronized Statement can only be executed after a thread has acquired the lock for the object or class referenced in the Synchronized Statement.

**N:** A Daemon thread runs in the background, providing services to other treads. It doesn’t prevent the program from terminating if it’s the only thread running.

**@. Thread life cycle in Java**

**Thread has the following states:** New, Runnable, Running, Blocked (Non-runnable) and Terminated

[](https://cdn.softwaretestinghelp.com/wp-content/qa/uploads/2018/01/Thread-Life-Cycle-in-Java.jpg)

In **New** state the Thread instance has been created but start() is not yet invoked.

**Runnable**: The Thread become runnable state after invoke the start() but before the run() is invoked and thread can also return to the runnable state from wait/sleep by yield(). Thread is ready to run but waiting for CPU scheduling.

**Running**: The thread is in running state after it calls the run() now the thread begins the execution.

**Non-Runnable (Blocked) / Waiting**: Here a thread is waiting indefinitely for another thread to perform a particular action. The thread become not in runnable state with the help of sleep(), wait() or block() methods and it will return to runnable state after some time. In other words.

**Timed Waiting:** Here a thread is waiting for another thread to perform an action up to a specified waiting time.

**Terminated**: Once the run() is completed (i.e. finished execution) then it is terminated.

**Commonly used java Thread methods:** sleep(), start(), run(), join(), getState(), getName()…..

**N:** When thread blocks I/O, thread enters the waiting state.

**@. Class and Interface -----** ommk Ref: Java\_Examples\ToplevelClass\_Declare\_with\_Default

|  |  |
| --- | --- |
| **Class** | **Interface** |
| Class describes the attributes and behaviours of an object. Class is a blue print. A top-level class can create as public and default. | Interface is a template and it is a blueprint of class |
| Class may contain both concrete methods and abstract methods. | Interface contains only abstract methods. |
| Members of a class can be public, private, default or protected. | Members of the interface can be public by default. |
| Extent keyword is used to inherit the other classes. | Implements kw is used to implements the interface through class. |

**N:** private and protected access modifiers can’t be used with Outer class or Interface.

**@. Serialization and Deserialization &&&& Methods are used during Serialization and Deserialization process**

|  |  |
| --- | --- |
| **Serialization** | **Deserialization** |
| For security purposes converting the objects into byte stream (serial bytes) is known as ‘Serialization’. for this, we need to implement java.io.serializable interface. | Deserialization is the process reconstructing object from byte stream. |
| An object is serialized by writing with FOS and OOS. | An object is deserialized by reading with FIS and OIS |

ObjectOutputStream.writeObject**🡪** Used to serialize objects and write the serialized object to a file.

ObjectInputStream.readObject **🡪** Used to reads the file and deserialize the objects.

**N:** To serialize an object must implement the **serializable** interface. If super class implements Serializable then subclass will automatically serializable. **-----** Ref for above: Java\_Examples\TransientKeyword / 3

**Ex:** Under serialization and deserialization, we can consider JSON and XML parsing: here will understand the libraries like Jackson or Gson for serializing and deserializing data in JSON format, which is essential for API testing.

**@. Use of transient keyword (or) Purpose of a transient variables -----** Ref: Java\_Examples\TransientKeyword/2 /3

**Transient** is used in [serialization](http://quiz.geeksforgeeks.org/serialization-in-java/) process to restrict field/variables should not be serialized when the object is serialized. At the time of serialization, it instructs the JVM to exclude that variable from the serialization process. it ignores the original value of the variable and instead of this stores the default value of the variable for Ex, if a program accepts the sensitive fields like user name and pass but we don't want to store the original pass in the file in this scenario we use **transient keyword**. It is not used with static variables. **Ex:** transient int in = 30;

**Adv:** security, performance and data integrity.

**@. Static and Non-static (Instance) variable &&&& method ----** wrwoam – with or without access modifier Ref: Java\_Examples\EX\_static\_nd\_nonstatic\_Variables

|  |  |  |
| --- | --- | --- |
| **Static Variables** | | **Non-static Variables** |
| SVs are declared with static keyword these are also known as Class level variables these are available for all methods | | NSVs are not declared with static kw these are also known as Instance V’s these are restricted. Syn: wrwoam dT varName; |
| We can access static variables using Class Name | | We can access Non-static variables using Class reference |
| Static var’s can access by both static & non-static methods | | Non-static var’s can access only by non-static methods. |
| SVs reduce the amount of memory used by a program and these members are store in java common memory. Here memory allocation is done at the time of class loading. | | NSVs do not reduce the amount of memory used by a prog (i.e., to access NSVs we need to create object so all the non-static stuff will move into this new object(heap)). Here memory is allocation is done when object is created to class. |
| SVs are shared among all instances of a class | | IVs are unique to each instance of a class |
| **Static Method** | **Non-static Method** | |
| Static methods are always accessed with Class Name | Non-static methods are always accessed with Class reference | |
| Static methods can access static variables | Non-static methods can access both static and non-static var’s | |
| **Static** method uses Compile time / early / static binding. | **Non-static** method uses Runtime / Late / dynamic binding. | |
| Due to early binding these cannot Override. | These can Override. | |
| Overloading is possible | Overloading is possible | |
| **Static** method uses less memory for the execution bcz memory allocated only once at the time of class loading. | **Non-static** method uses much memory for execution bcz memory allocated every time when the method is called. | |

**N:** If we try to access static mtds by using obj reff the warning will show as ‘static field should be accessed in static way’.

**@. Y should avoid static method in java**

A static method is declared using static keyword. It belongs to the class, not instances. It doesn’t require it to be instantiated. It can be called using class name without creating the object. This cannot allow “this” or “super” keywords. Because of this, with the static method we cannot fully utilize the Object-Oriented Feature, so it is good practice to avoid static methods. however, some situations we might need to use static methods intentionally.

**@. Purpose of Volatile Variable** ---- Ref: Java\_Examples\VolatileExample / 2

By using **Volatile variables,** the values are always read from the **main memory** not from **thread's cache memory**. It can use mainly during synchronization. Volatile keyword is applicable only for variables to indicate that the variable’s value may be modified by multiple threads asynchronously. It ensures that any read or write operation on the variable is directly performed on the main memory, rather than relying on CPU catches, and also ensure the visibility of changes to variables across threads (i.e. change made by one thread is visible to other threads immediately), preventing data inconsistency issues.

**@. Terms used in Inheritance**

**Sub Class** is a class which inherits the other class it is also called a **c**hild class, **d**erived class or **e**xtended class.

**Super Class** is a class from here the subclass inherits the features. It is also called a **p**arent class.

**Base class** is the most generalized class, and it is said to be root class.

**extends** keyword is used to establish an inheritance relationship between classes.

**Adv:**

* **Reusability (i.e.** Functions can reuse**)** facilitates us to **reuse** the members of existing class by creating a new class. Ex for reusability TCs: Login TC can be referenced in multiple workflows.
* Productivity will increase
* Project development time and testing time is reduced

**@. Types of inheritance** ----- OOPS\_Inheritance/Types\_Of\_Inheritanc**e**

|  |  |
| --- | --- |
| If one class extends the other class is called **Inheritance**. These are 3 types: Single, Multilevel and Hierarchical (one parent and many child class).  Types of inheritance in Java | If one class extends multiple classes then it is known as **Multiple Inheritance**. These are: Multip**le** (when a class inherits more than one class) and Hybrid (combination of more than one type of Inheritance)  Multiple inheritance in Java |

**@. Source Code, Object and Byte Code**

**Source Code is a collection of computer instructions written in human-readable programming language.**

**Object Code is a sequence of steps in machine-level code and it is the output of after** [compiler](https://www.differencebetween.com/difference-between-assembler-and-vs-compiler/)**converts the Source Code in languages like C or C++ with a platform-specific compiler.**

**Byte code** is intermediate code generated after the compiler compiles the source code. It is platform independent and it can be executed by JVM.

**@. Exception Hierarchy in java / Exception and Error**

All the Exceptions and Error types are the sub classes of **Throwable** class it is a superclass. It is the base class in hierarchy. **Exceptions are caused by our program only** and these can handle with the help of try-catch blocks. **Errors are caused due to lake of system resources these should not try to catch,** these are thrown by Java run-time system (JVM). Ex: OutOfMemoryError (When JVM runs out of memory caused by memory leaks, large objects, or infinite loops), StackOverflowError, VirtualMachineError, AssertionError.

**@. H JVM handle an Exception / Default Exception Handling**

If an exception occurred in a method, the method creates an Object is called ‘Exception Object’ and handover it to run-time system (JVM) is called **Throwing an Exception**. The exception object contains a lot of debugging information such as method hierarchy, line number where the exception was occurred, type of exception, name and description of the exception, etc. For default exception handling the following procedure will happen.

* Java run-time system searches the **call stack** (ordered list of the methods) to find the appropriate method that contains block of code this can handle the occurred exception. This block of the code is called “**Exception Handler”**.
* If it finds appropriate handler then it passes the occurred Exception to it. Appropriate handler means the **type of the exception thrown** matches the **type of the exception it can handle**.
* If java run-time system couldn’t find the appropriate handler after searches all the methods in **call stack** then run-time system handover the Exception Object to **Default Exception Handler**, which is part of run-time system. This handler prints the exception information in the following format and terminates the program **abnormally**.

**Ex:** public class ThrowsExecp {

       public static void main(String args[]) {

       String str = null; //default value of string is ‘null’

         System.out.println(str.length());

        } }

**Output:** Exception in thread "main" java.lang.NullPointerException: Cannot invoke "String.length()" because "str" is null

At pac\_name.class\_name.method\_name(class\_name.java:lineNum)

**@. Java Keywords**

**Java Keywords** are predefined and these are the reserved terms in java. They have special usage in java programming language these are used to perform some predefined tasks. For Ex **Keywords** like int, for, class, etc the terms are reserved means they cannot be used as identifiers for any other programming elements including classes, subclasses, variables, methods and objects. They are part of the **Java** programming language syntax and these are care sensitive.

**N:** Identifiers are user-defined names these are used to identify method, variables, classes and other prog’g elements.

**@. Maps / Can a map can have a Null value / Y and when we use Maps -----** ickud Blue - Interface green - class

Java Map is an interface. Map is a part of collections framework. It stores the elements in the form of key-value pair. It maps unique keys to values. Few characteristics of the Map Interface are:

1. Map doesn’t allow dup keys but allow dup values. It allow one <null key, null value> like HM and [LinkedHashMap](https://www.geeksforgeeks.org/linkedhashmap-class-java-examples) but some do not like [TreeMap](https://www.geeksforgeeks.org/treemap-in-java/).
2. The order of a map depends on specific implementations **Map hierarchy**

**Adv:** The maps are used to perform lookup a value by key or when someone wants to retrieve and update elements by keys. Some examples are: Map of error codes with their descriptions. **&&** Map of zip codes with cities.

**Creating Map Objects:** Because of Map is an interface, objects cannot be created to map. We always need a class that extends the map in order to create an object. After introducing Generics in Java 1.5, it is possible to restrict the type of object that can be stored in the Map. Syn: Map hm = new HashMap();

**N:** To iterate map we have to use entrySet() and foreach loop. Ref: Java\_Examples\MapIterationEx

**Map and Set:** Map stores k-v pairs, where each key is unique and Set stores unique elements without any specific order.

**@. Methods used in Map Interface ---- Ref:** **\Java\_WD\_Ex\src\Java\_Examples\AAA\_java\find occurrence of char’s** <https://www.geeksforgeeks.org/map-interface-java-examples/>

1. .put(key, value): is used to insert the elements in Map in the form of <key, value> pair format.
2. .putAll(Map map): Copies all the mappings of the specified map into the new map.
3. .remove(key) | remove(key, value): This method is used to delete an **entry** for specified key.
4. .get(key): is used to #1the value for specified key.
5. .getKey() | .getValue(): is used to return the key and value from Map.
6. .values(): is used to return all the values in map
7. .containsKey(key | value): is used to check whether the specified key | value is avl in Map and it ret Boolean value.
8. Map.Entry<datatype, datatype >: Entry interface enables to work with a map entry
9. keySet(): This method will return all the keys in Map.
10. entrySet(): Creates a set and stores the map elements into them. Return type of entrySet() is Set<Map.Entry<k, v>>

**@. Static keyword in java ---- Ref: Java\_Examples2\Static\_Keywords\_UsedFor**

In Java, **Static keyword** is mainly used for memory management and it is a Non-access Modifier. It can be used to create Class level variables, methods, blocks or nested classes that is belong to the class itself, rather than instance of the class.

**N:** When a variable is declared as a static, then a single copy of var is created and shared among all object at class level.

**@. Data Types** Ref: Java\_Examples2\PrimitiveDataTypes

* **Primitive** data types are predefined in java these include Boolean, char, byte, short, int, long, float and double. A data type is primitive means, if we assign value to the variable that variable actually stores that value. These store only single value with no special capabilities. This size is fixed. Stored directly in memory. These are immutable.
* **Non-primitive / Reference** data types are created by programmers these include String, Arrays, Classes, Interfaces and Objects. They directly don't store the value in memory but stores a reference to that value in memory it is also called address of that value. These store multi values and methods. This size is not fixed. Stored as a reference, Mutable.

|  |  |
| --- | --- |
| Boolean Type:  **boolean -** Rep's true or false values (default value false)  Character Type:  **char -** Stores a single 16-bit Unicode character (**Y char is 2 bytes in java? :** To support Unicode characters)  Integer Types:  **byte -** Stores an 8-bit signed integer (-128 to 127)  **short -** Stores a 16-bit signed integer (-32,768 to 32,767) | **int -** Stores a 32-bit signed integer (-2^31 - 2^31-1) (size 4-bytes)  **long -** Stores a 64-bit signed integer (-2^63 to 2^63-1)  Floating point types:  **float -** Stores a 32-bit floating-point number Ex: float f = 0.25f  **double -** Stores a 64-bit floating-point number (size 8 bytes)  Ex: double d = 5.55; (or) double d = double(f);  **String -** Stores a sequence of characters  **Array -** Stores an ordered collection of values |

**N: toArray(**array\_obj**)** method is used to get an Array from ArrayList which contains all the elements in ArrayList object in proper sequence (from first to last element). Ref: Java\_Examples\ArrayList\_to\_Array\_conv

**@. JDK, JRE, JVM and JIT / Components of java platform**

**JDK** provides the environment to **develop the** Java applications. **JDK**contains JRE along with various Development tools (to provide an environment to develop the java programs) like Java libraries, source compilers, debuggers, etc.

**JRE** is used by end-users to run the java applications. It is an installation package that provides the environment by packages,class libraries and other resources to **run** the java applications on any machine. JRE is the part of JDK. **JRE** is an implementation of the JVM.

**JVM** is for executing byte code. It is an Abstract Computing Machine that provides environment to execute java bytecode (JVM takes byte code and it converts into lower level code line by line). It is very important part in both JDK and JRE because it is contained or inbuilt in both. Whatever Java program is run by using **JRE** it goes into JVM. JVM is a type of **Interpreter** and it is responsible for converting the byte code into machine readable code and **executing the java program line by line**.

**N:** An **Interpreter** is a component of Java Virtual Machine (JVM) that executes Java bytecode line by line. Bytecode is not an executable file to execute a bytecode file, we actually need to invoke a **Java Interpreter**.

**N:** An **Abstract Computing Machine** is nothing but a detailed analysis of how the system works.

**N:** Compiler converts the**:** source/high level code --> object code --> Machine code/lang (0’s & 1’s) **:** Before program run

Interpreter converts the**:** source/high level code --> Intermediate code --> Mach code/lang (0’s & 1’s) **:** During prog run

**@. Interpreter vs JIT Compiler**

Interpreter executes bytecode line by line (slower but requires no additional compilation time).

Just-In-Time (JIT) Compiler converts bytecode into machine code for faster execution.

**@. Can a class be declared as protected**

**Classes** and **Interfaces** are cannot be declared as a **protected** but if Variables, methods and constructors are declared as protected in a class these can visible with in the package (include all classes in package) and outside the package subclasses (of package) only can access.

**@. Can I declare class as static or private** Ref: Java\_Examples2\Static\_Keywords\_UsedFor\Static\_Class /InnerClassRequires\_OuterClassReference

Yes, we can declare a class as static/private inside a top-level class such classes are also known as **Nested** **Classes**. These can create as both static and non-static. **N**on-static **N**ested class are called **N**on-static I**nn**er class. If we want to make a top-level class as static/private, then it's not allowed because we can’t able to create object to static class. Static nested class doesn’t require the Outer class reference/instance and cannot access the Outer class non-static members. **N**on-static Inner classes only require the Outer class reference.

**N:** An Inner class maybe declared as a public, private, protected, static, final, or abstract.

**@. Inner class and Nested class**

**Inner class:** When a class is defined within another class, then it becomes inner class. A (non-local) inner class maybe declared as public, private, protected, static, final, or abstract. Types: static, non-static, local, and anonymous.

**Nested class:** If the access modifier of the inner class is static, then it becomes nested class.

**@. Who can access the class member with a private modifier**

If class members are declared as a **private**, it can access only within the class in which they are declared. Any other class of same package will not able to access these members.

**@. Can a class be default in java** ----- Ref: Java\_Examples2\AccessModifier\DefaultClassEx

A top-level class can declare as public or default (no modifier) but we cannot declare a private or protected.

**@. this & super keywords, this() & super() constructor calls** -- https://www.scaler.com/topics/java/this-and-super-keyword-in-java/ Ref: ConstructorConcept/ExOnSuper ConstructorConcept/ExOnThis3

**this keyword:**

Is used to refers the instance and static variables of current class.

**Is** used to access and modify the instance and static variables of current class.

**Is** used to invoke the current class methods (implicitly)

Can be passed as an argument while method is calling. Ex: mtd(this);

Can be passed as an argument while constructor is calling. Ex: ClassN classn = new ClassN(this);

**I**s used to return the current class instance value. Ex: return this;

**this() constructor call:**

**Is** used to invoke the current class constructor

**super keyword:**

Is used to invoke immediate parent class instance and static variables.

Is used to invoke an immediate parent class method.

**super() constructor call:**

is used to invoke an immediate parent class constructor.

**Similarities:**

* Both **this** and **super** keywords are non-static, so they can't be used inside static context. (we cannot use both the keywords inside the main method in Java). Ref: ConstructorConcept\ProveThisAndSuperisNonstatic
* super() and this() constructor calls can be used in constructor chaining to call another constructor to reduce the code duplication. this() calls the no-argument constructor of the current class and super() calls the no-argument constructor of the parent class.
* super() and this() constructor calls can be use only from constructor.
* this() and super() constructor calls must be the first statements inside a constructor because of this we can call only one function and use only one keyword in child class constructor Ref: Constructor\_Chaining\SimpleConstructoeChaining , \CannotCallConstructorCallsRecursively
* We cannot use **super()** and **this()** constructor calls together (recursively) inside a constructor because if we use together then one will be 1st statement and other will be 2nd statement which will lead to compilation error. Ref: above prog

**@. List and Set**

Both are interfaces.

|  |  |
| --- | --- |
| **List** | **Set** |
| **List** is a type of ordered collection so it maintains the elements in specific order (prints in insertion order). | **Set** is a type of unordered collection so it maintains the elements in any order (prints in random order). |
| **List** allows duplicate elements. Mtds avl in **List** not in **Set**: indexOf(), get(), lastIndexOf() | **Set** doesn't allow duplicate elements. |
| **List** allow multiple null elements. | **Set** can allow only one null element. |
| List is index-based and allow access via indices (list.get(index)). | Set does not support index-based access. |

**Ex: Set:** Hashset, TreeSet, LinkedHashSet (Set<Setring> set = new HashSet<>())

**List:** ArrayList, Vector, LinkedList (List<String> list = new ArrayList<>())

**N:** addAll() is used to convert List to Set (All the elements in List will get added to Set)

**@. Static and Final**

|  |  |
| --- | --- |
| **Static** | **final** |
| static keyword is mainly used for memory management. It is used to **define** class members and these members can overload and cannot be overridden | final keyword is used to **declare** constants these members can overload and cannot be overridden. Ex: for f**inal class** is **String** |
| **In Modification point of view** Static variables can modify. | final variables can’t modify |
| Class object is not required to access **static** members | We can create Object to final class. |

**@. Iterator and ListIterator**

|  |  |
| --- | --- |
| **Iterator** | **ListIterator** |
| Iterator travers the elements in forward direction only | ListIterator travers the ele’s in both the directions |
| Iterator can be used in List, Set and Queue. | **List**Iterator can be used in **List** only. |
| Iterator can only perform remove operation while traversing the collection. | ListIterator can perform add, remove and set operations while traversing the collection. |

**@. Can a Main Method be overridden?**

In Java, A main method can able to **overload** but not **overridden** simply because it is a **static** **method**. Reason for this, Static content will store in java common memory not in new object (because no need to create object for static). And Static methods are associated with class itself and not associated with instance of class (object).

**N:** A Main mtd is entry point in java program so it must be declared as ‘public’ then only JVM can access from anywhere.

**@. Main method**

Main method is a starting point of execution in java program.

public static void main(String[] args) {}

public: This allows the method to be accessible from anywhere, ensuring the JVM can call it from outside the class.

static: It allows the method to be called without creating an instance of the class. Since the JVM invokes main directly, it must be static to avoid the need for an object.

void: The main method does not return any value.

main: This is the predefined name that the JVM looks for as the program's entry point.

String[] args: This is an array of String arguments that can be passed to the program from the command line.

**@. Base class of all java classes / Object class**

Object class is a super base class of all Java classes and its belongs to java.lang package. We can assign any type of Object to Object class (Ex: Object obj = new Fruit();)

**mtds pro by Object Class:** toString() (ret the string rep of this obj), getClass(), equals(Object obj), finalize(), hashCode().

**N:** This concept cannot be generalized to all OOPS lang’s. For instance, in C++ there is no such super class of all classes.

**N:** In java **object** cannot be declared as **static** because static statements are runs as soon as class containing them and class is a model for crating object so it won’t come under static. **N:** static methods cannot be overridden by sub class.

**@. Increment Operators: i++ and ++i -----** Ref: Java\_Examples\IncrementOperators

In java, increment is performed in two ways those are:

**Postfix / Post-Inc (i++)**:If we want to use the current value of **i** and then increase that value of variable **i** by **1**.

**Prefix / Pre-Inc (++i)**: If we want to increment the value of variable **i** by **1** and then use the incremented value in our statement.

**@. Instance and Object ----** am esc rrmd Ref: Java\_Examples2\Object\_nd\_Instance

* **I**nstance means **a**llocating the memory space by JVM and Object means initializing the variables inside the instance.
* Class is a model for creating object and **Instance** is a unique copy of the object (same structure but different data)
* Instance can exist Logically but object can exist Physically (that means physical presence of the object in memory)
* We can’t store an instance but we can store an object in database or file system.
* We can’t create an object for Abstract class and interface (Bcz if our class is not fully implemented then java will not allow to create the object).
* Instance refers to object reference.
* An Object represents set of instances. But instance represents the specific representation. For an example if we take living beings on the earth is a class. human is an object. But you and me are the instances for human.
* A single object can have more than one instance.
* Instance will’ve both class definition (public class Myclass{}) and object definition whereas object will’ve only the object definition.

**@. VirtualMachineError and AssertionError**

[Java.lang.](https://docs.oracle.com/javase/7/docs/api/java/lang/VirtualMachineError.html)**VirtualmachinErrror** is thrown to indicate that an internal error or resource limitation which prevents it from functioning. It is a self-defensive mechanism employed by JVM to prevent the entire application from crashing.

Java.lang.**AssertionError** is thrown while working with Assertion conditions. It’s thrown when an assert statement is fails

**@. Access Modifier and Non-Access Modifier** -----<http://tutorials.jenkov.com/java/access-modifiers.html> Ref: Java\_Examples2\AccessModifier

In older languages like C++ public, private, protected, default is considered as Access / Visibility Specifiers. Except this the remaining all like static, etc are considered as Access Modifiers. But in java there is no terminology of specifiers all are by default considered as Modifiers. By Access Modifiers we can provides access right to our code to other classes, fields, methods and constructors so it represents that whether other classes can access and modify our code or not. For Ex: public, private, default, protected are considered as Access Modifiers.

Proof: If we declare top level class as private will get compile-time error as “modifier private not allowed here”

**Adv:** AM’s are used to achieve Encapsulation (restrict direct access of data members but provide indirect access to set and modify the data members), improve security, make code more maintainable and reusable.

**Non-Access Modifiers** are used to provide additional info about a classes, interfaces, methods or variables to JVM. They do not control the access levels. NAMs are: static, final, abstract, synchronized, volatile, transient, native, strictfp.

**N:** In java **Token** is nothing but a smallest unit of our program like keywords, identifiers, objects, string literals (where the values are used in the program is nothing but a **literal**) (inside “ ” of string content, value of int) and operators.

**N:** In java main class accepts ----public, default, abstract, interface, final keywords

**@. Code and Algorithm**

An **Algorithm** is a sequence of steps for computing a task these are usually executed by computer programs.

A **Code** is a sequence of steps for computing a task and these are usually executed by machines. In many cases, code is composed in a high-level language that is automatically translated into machines understandable code.

**@. equal(), equalsIgnoreCase() and contains()**

Let us consider Actual content as ‘Selenium Project’

equal(Selenium Project) ---- Maintain care sensitive (UpperCase & LowerCase) while comparing string content.

equalsIgnoreCase(SeleNium pRojecT) ---- Ignore care sensitive while comparing string content

contains(Selenium) ---- check only some matching portion of string content

**@. Type Casting / Implicit and Explicit Type Casting** ---- Ref: /Java\_Examples2/TypeCasting

Type Casting is the process of converting value of one data type to another DT. (Ex: int i = (int) 5.5) These are 2 types:

**Primitive Type Casting** is a process of converting one primitive datatype to another primitive datatype. These are 2 types: **Widening / implicit TC** / UCis nothing but a converting smaller datatype to larger datatype. It can be done by both implicitly (automatically) and explicitly by java compiler. And **Narrowing / Explicit TC** / DC is wise versa it can do only programmatically by using typecast operator (i.e. explicitly (manually)).

**WC:** byte -> char -> short -> int -> long -> float -> double **NC:** double -> float -> long -> int -> short -> char -> byte

**Non-primitive Type Casting** is a process of converting one non-primitive datatype to another non-primitive datatype. It can achieve if the class will have **IS-A** relationship. These are 2 types: Up Casting (generalization) & DC (specialization)

**Break** keyword will use in switch, if and loop (for, for-each, while) statements to terminate the loop.

**Continue** keyword will use in loops only to skip the remaining code in current iteration inside a loop and procced to the next iteration in the loop. it doesn’t terminate the loop.

**N:** Break kw is used to exit a loop immediately whereas return kw is used to exit ‘if’ condition Ref: Java\_Examples2\Return\_Keyword\b\_ReturnKeywordInVoidMtds

**@. Types of Relationships ----- Ref:** Java\_Examples2\Relationships\_or\_Associations

There are three types of Relationships are possible between classes that define how they interact those are: **IS-A** Relationship (Inheritance) (here subclass inherits properties from parent class), **HAS-A** Relationship (Association) and **USES-A** Relationship (Dependency). Again HAS-A Relationship (Association) as two types those are:

**Aggregation** is a weaker HAS-A relationship (here two objects have related but both objects can exist independently. One object does not have a full control over the other object so that cannot access & modify its data)

**Composition** is a stronger HAS-A relationship (here there will be a main object and dependent object. The dependent object cannot exist independently without the main object so the main object has full control over the dependent object so that main object can access & modify its data).

**USES-A** Relationship (Dependency) occurs when a class depends on another class to perform a task often passed as an argument to a method or used temporarily within a method.

**@. Types of Arguments / Call By Value and Call By Reference ---** Ref: Java\_Examples2\Pass\_by\_value\_nd\_pass\_by\_reference

There are two ways to pass arguments to a function those are**“call by value”** and **“call by reference”**. A method can call by passing the value is called **Call by value**. Java supports only **call by value**. If any change in value of a variable inside the function, that doesn’t affect the original value of that variable. A function can call by the reference / address of object is called **Call by reference**. If any change in value of a variable inside the function, that affects the original value of that variable. C and C++ supports **call by reference** but Java does not support this because of java does not supports pointers (by using pointer the address of actual parameter passed to formal parameter in **call by reference**).

**@. To initialize Array of Objects** ---- Ref: /Java\_WD\_Ex/Java\_Examples2/ArrayOfObjects

Array stores the values of int, Boolean (default value ‘false’), string, etc., but Array of objects stores class type elements. In 2 ways we can initialize the Array of objects in those are: by using constructor or by using separate member method.

**@. Garbage Collector / H java achieve memory management** Ref: Java\_Examples\GarbageCollector

Java uses automatic Garbage collector to manage memory. **Garbage Collector** will collect / deleting the objects that are no longer being used from java heap memory. It helps memory to free during the program execution so that system will faster. Garbage collector is runs automatically and not necessary for programmer to externally call it.

**N:** A **finalize()** will call by garbage collector before an object is garbage collected. (diff b/n finally and finalized)

**N: Pointer** is a variable that refers the address (memory) these are unsecure so java does not support.

**N: Ternary operator** is used to replace if-else statement to check the condition. It is also known as the conditional operator. **Syn:** condition ? expression1 : expression2; (If condition is true, expression1 is executed or If condition is false, expression2 is executed. Ref: Java\_Examples\Ternary\_Operator3

**N:** Blank / un initialized final variable is called as only **Declaration** not initialization.

**N: toCharArray()** is used to convert String to character Array. Ref: Java\_Examples/AAA\_EOT.java/Split number

**@. Packages**

A package is a collection of related classes, interfaces and sub-packages. These helps to organize the code, to prevent naming conflicts, Readability & Reusability and to provide control access to classes and interfaces. Import keyword is used for importing already created packages, sub-packages (nothing but a Packages that are inside another package). Restrictions: It must appear first line in source code file (eliminating blank lines and comments). There are 2 types of packages:

**Built-in packages (**imp pack’s**)** are provided to reduce the program burden for Ex: Java.lang, Java.util, java.awt, Java.io.

**Custom / User-defined packages** are created by users with package keyword. To create custom packages: Choose name for package (it should be unique and descriptive) and Create a java file for each class in package.

**@. Restrictions are placed on the location of a package statement in source code**

A package statement must appear as a first line in source code file (eliminating blank lines and comments).

**Jump statements:** loop control statements (break, continue) and return Ref: Java\_Examples\Jump\_Statements

**Operators in java:** Arithmetic operators (+, -, \*, %, /), Assignment operators (=, +=, -=, \*=, /=, %=), Bitwise operators (^, |, <<(left shift), >>(right shift)(It carries the sign bit when shifting right), >>>(zero-fill right shift)(zero-fill bits that have been shifted out)), Logical operators (&&, ||, !), Relational operators (==, !=, <, >, >=, <=)

**String methods:** charAt(), length(), format(), substring(),contains(), join(),equals().

**@. Can interface implements another interface**

An Interface cannot be implements another interface.

An Interface cannot extends a class.

An Interface can extends one or multiple interfaces.

A class can implements one or multiple interfaces.

A class can extends only one class at a time.

**Method Chaining** is a process to connect one method with another method in a single line code by using dot operator. **Ex:** obj.m1().m2().m3()...; Ref: Java\_Examples\MethodChaining

**Constructor Chaining** is a process of calling one constructor from another constructor **or** from a subclass constructor to a superclass constructor. It helps to initializing the objects and reusing the code. ‘this()’ is used to achieve chaining in side same class and super() is used to achieve chaining from child class to parent class. Ref: Constructor\Constructor\_Chaining

**Singleton Class** is a class that allows only one instance to be created and provides a global point of access to that instance. It is implemented by using a private constructor, a static instance variable, and a public static method that returns the single instance. Ref: Java\_WD\_Ex\Constructor\Singleton\_Class\LazyInitialization\_SingletonClass

**N: Variable Hiding occurs when a variable in a subclass has the same name as a variable in superclass. So, the subclass variable hides the superclass variable.** Ref: Java\_Examples2/Hiding/variableHiding3

**N: Method Hiding** occurs when a static method in a subclass has the same signature as a static method in superclass. So, the subclass method hides the superclass method. MH is also referred as CT Polymorphism. In method hiding method won’t overridden by other classes. Ref: Java\_Examples2/Hiding/MethodHiding2

**@. Shallow copy and Deep copy** Ref: Java\_Examples2\ShallowCopy\_DeepCopy

Shallow copy and deep copy are two ways of copying objects. The difference in how the object references are handled.

A **Shallow copy** creates a new object, but copies references to the nested/inner objects instead of duplicating them. Changes made to the nested objects in the copy will also reflect in the original (since both point to the same reference).

A **Deep copy** creates a completely independent clone of the object and all nested objects. Changes in the copy do not affect the original.

**N: clone()** is used to create a copy of an object. The class must implements the Cloneable Interface to allow cloning.

**@. Method Overriding and Method Hiding / Shadowing**

|  |  |
| --- | --- |
| **Method Overriding** | **Method Hiding** |
| Both the parent class and child class methods are non-static | Both the parent class and child class methods are static |
| Method resolution is done on the basis of the Object type | Method resolution is done on the basis of reference type |

**@. Parameter** (formal Parameter) **vs Argument** (Actual Argument)

A Parameters are the variables those are declared in method signature. Ex: public void add(int a, int b) {}

An Arguments are the variables/values those passed to a method when it is called. Ex: Add ad = new Add(1, 2);

**N:** Java doesn’t allow default arguments.

**@. Reference Variable and Default value for the reference variable**

A variable which refers to an object is called Reference Variable it stores the location (memory address) of an object. The default value of reference variable in java is ‘null’.

**@. Static and Dynamic method dispatch**

When execution of a method remains same as per binding provided by the compiler then such dispatch is called SMD. It is used for private, final and static methods.

When execution of a method not remain same as per binding provided by the compiler (because method invoking (call) is decided at runtime) then such dispatch is called DMD. It is used for all other methods except above.

Ex: Anima a = new Cat(); //Up casting ==> a.sound();

**N:** A local inner class without name is known as **Anonymous class**. It always extends a class or implements an interface. Mainly it is used to override the methods and used for one-time use where a full class definition is not required. Ref: Interface\OuterClass

**N: Lambda expressions,** introduced in Java 8, It provides a clear and concise way to implement functional interfaces (interfaces with a single abstract method). It enables to write more clean, readable, and functional-style Java code. Syn: (Parameters) -> Expression\_or\_Block Ref: Interface/Z1Ex3\_FunctionalInteface Z1E\_FunctionalInteface\_LambdaEx

**@. Types of Interfaces ----** Ref: Interface/Z1Ex1\_FunctionalInteface SAM - Single Abstract Method

**Normal Interface:** Having two or more unimplemented Abstract methods

In java 8 **Functional Interface (SAM)** can have only one Abstract method and any number of private, static and default methods. Lambda Expression is used to implement Functional Interface in java. It indicates with @FunctionalInterface annotation.

**Marker Interface** is an Interface it has no methods or fields inside it, used to provide metadata to the JVM or compiler (i.e. provides additional information about a class to compiler and runtime environment (JVM). Some of the MIs are:

**Serializable** is empty Interface. It is used to serialize the objects. Serialization is the process of converting an object into a byte stream so that it can be saved to a file.

**Cloneable** indicates objects of the class can be clone.

**Remote Interface** indicates objects of the class can be used remotely.

**@. Class and Abstract Class**

* A class is declared using ‘class’ keyword and Abstract class is declared by using abstract keyword.
* We can create object to Class but we can’t create object to an Abstract class.
* A class can have only concrete methods but Abstract class can have both concrete as well as Abstract methods
* A Class can declared as final but an Abstract class can’t be declared as final because it must be extended.
* A Class may or may not have child class but Abstract class must have child class.
* A Class is not used to achieve Abstraction but an abstract class is used to achieve 0-99% Abstraction.

**@. Class and Interface**

* A class is declared using ‘class’ keyword and interface is declared using ‘interface’ keyword.
* We can create object to class but we can’t create object to an interface.
* A class can have only concrete methods but interface can have only abstract methods (no need to declare abstract before the method name these are by default Abstract in interface).
* A class can declared as final but interface can’t be declared as final.
* A class can have instance variables but interface can’t have instance variables. The variables of an interface by default public, static and final type.
* A class can have constructors but an interface can’t have constructors.
* A class can have static & non-static initializer blocks but interface can’t have static or non-static initializer blocks.
* Multiple Inheritance is not supported through class but it can possible through interface.
* A class is extended by another class but interface is implemented by another class.
* A Class is not used to achieve Abstraction but an abstract class is used to achieve full (100%) Abstraction.

**Method Reference**s are the special type of Lambda Expressions used to call methods by referring them directly using scope resolution :: operator. These are 3 types those are: i) Reference to a static method, ii) Reference to an instance (n-s) method and iii) Reference to a constructor. Ref: Java\_Examples2\MethodReference

**Instanceof Operator** is used to check whether an object is an instance of a specific class or implements a particular interface. It returns Boolean value. Ref: Java\_Examples2\InstanceOf\_Operator\E1/E2/E3\_

**Static / Static Initializer block** is a block of code inside a class that is enclosed in curly braces {} and preceded by the static keyword. It is used to initialize static variables of a class and it is executed when the class is loaded. Ref: Java\_Examples2\InitializerBlocks

**Non-Static / Non-Static Initializer block** is a block of code enclosed in curly braces {} but without static keyword. It is used to initialize instance variables of a class and it is executed an instance of class is created, before the constructor is called.

**@. W Access Modifiers are not allowed with class or interface**

* Nested Classes and Nested Interfaces can allow all AM’s.
* Outer Classes and Outer Interfaces can’t be declared with private or protected AM’s.

**@. C we decrease / reduce the visibility of method in child class while overriding the method of parent class**

No, we cannot reduce the visibility of overridden method in subclass so that the overriding and overridden methods must be same access modifiers.

**Generics:** Generics provide/enable compile-time type safety by allowing a class, interface, or method to operate on specific data type while avoiding runtime errors. Before generics java collections could only hold objects of type Object, it leading to unsafe casts. Generics allow for type safety without sacrificing flexibility. **Generics** enhance the flexibility, type safety, and reusability of code by allowing to create classes, methods, and interfaces that can operate on various data types without loosing type safety. **Generics** allow (Integer, String, … etc., and user-defined type) parameters. **Common Type Parameters:** T - Type, E - Element, K - Key, N - Number and V - Value. Ref: Java\_Examples2\GerericClass **Ex:** List<T> (List<String> list = ArrayList<>(); It ensures the list can only hold String and no other type) or Map<K, V> **Adv’s:**

i) Type-Safety (We can hold only a single type of objects in generics. It doesn’t allow to store other objects)

ii) Type Casting Is Not Req.

**@. H Sub Class is different from Inner Class ----** Ref: Java\_Examples2\SubClass\_nd\_InnerClass

* Sub class is a class that extends or inherit another class. Inner Class is a class that is nested within another class.
* Sub class can be accessed directly. Inner class can only be accessed using outer class reference.

**@. Mutable and Immutable Object ----** Ref: Java\_WD\_Ex\Java\_Examples2\Mutable\_nd\_Immutable\_Objects

* A MO value can be change in memory after it's created and immutable object value can't change in memory.
* In concatenation, MO takes same memory and IMO will create new memory.
* MOs provide a method to change the content of the object. IMOs do not provide any method to change the values
* For ex String is immutable and StringBuilder & StringBuffer is mutable

**@. Set and Map**

* Set is used to construct a mathematical set in java. Map is used to mapping database.
* Set doesn’t allow duplicate values. Map can allow duplicate values but doesn’t allow keys.
* We can easily iterate the Set elements using keySet() and entrySet() methods. Map elements cannot be iterated, we need to convert Map to Set for iterating the elements.
* Insertion order is not maintained in both List and Set.

**Java Built-in Annotations** provide metadata about code and also helps to improve code maintainability and readability. Ex: @Override, @Deprecated, @SupressWarnings.. Whereas Custom annot’ns provide additional features Ex: “skipping”

**@. Structure and Class**

A structure is used for grouping data and the default access type of a structure is public whereas class is used for grouping data and methods and the default access type of a class is private.

Operators not overloaded: Scope Resolution(::), Member Selection(.) & member selection through a pointer to fun (.\*)

**N:** When we declare value to variable, those variables are created in stack memory and if these variables are out of scope those variables get garbage collected.

**N:** **clone()** is used to create a copy of an object. It is part of the cloneable interface, and classes must override it to support cloning.

**N:** We can use functions directly without creating a class by **static** **import**. Ref: AUTOMATION\SimpliiAppium2\_\src\test\java\SimpliiAppAppium2\EX04\_DragAndDrop\_Using\_TouchActionClass

**N: Autoboxing** is the process of converting primitive type to its corresponding wrapper class and **Unboxing** is wise versa.

**N: Scanner class** is used to read inputs from various sources such as console or file. It provides methods for reading different type of data.

**File class** is used to represent and manipulate file paths. It provides methods for creating, deleting and navigating file.

**N:** System.arraycopy() method is used to copy elements from both arrays into the result array.

@. Ways to take input from console ---- Using command line argument, BufferedReader class, Scanner class.

**@. Java coding standards for constants**

In java constants are created using static and final keywords

* Constants are created using static and final keywords
* Constants contains only uppercase letters and it should be nouns.
* If the constant name is a combination of two words it should be separated by an underscore.

**Ex:** MAX\_VALUE, MIN\_VALUE< MAX\_PRIORITY, MIN\_PRIORITY

**@. Java coding standards for variables**

* In java variable name should starts with small letter and variable names should be nouns
* Short meaningful names are recommended
* If there are multiple words every inner word should start with uppercase character.

**N:** Inside interface we can declare abstract methods, default & static implemented methods. Here default implemented method is called using object reference and static implemented method is called using interface name. Ref: OOPS\_Abstraction\Interface\_DefaultMtd\_StaticMtd\_1\Default\_nd\_Static\_mtds

**N:** Event object class and Event listener interface supports events handling / processing

**N:** Reader/writer class is character oriented and inputStream/outputStream is byte-oriented.

**@. Atomic conversion in java**

In java widening conversion comes under atomic type conversion. AC is done by meet the following conditions:

1. When two types are compatible (Ex: int, float i.e. int can assign directly to float variable)
2. Destination type is larger than source type (Ex: int, long i.e. long is larger datatype than int)

**@. Situation where exceptions may arise in java**

* Accessing an element but that not exist in array.
* Invalid conversion of number to string or string to number (NumberFormatException)
* Invalid casting of class (ClassCastException)
* Trying to create an object for interface or abstract class (InstantiationException)

**N:** Default value of int variable if not initialized in a class is ‘0’

**N:** Default value of String variable if not initialized in a class is ‘null’

**N:** Default value of Boolean variable if not initialized in a class is ‘false’

**@. java.util classes and interfaces support Event Handling processing** ----Event object class and EventListener interface

Automatic type conversion in java

Java automatic type conversion is done if the following conditions are met

1) When 2 types are compatible. for Ex: int, float (int can be assign directly to float variable).

2) Destination type is larger than source type. for Ex: int, long (Automatic type conversion takes place if int is assign to long because long is larger data type than int)

**N:** Widening conversion comes under automatic type conversion.

**Conditional statements** are used to make decisions based on given conditions. The main types of conditional statements are: 1) if Statement, 2) if-else Statement, 3) if-else if-else Statement, 4) Nested if Statement, 5) Switch Statement. Ref: Java\_Examples\ConditionalStatements

**Loop Conditions:** Loops are used to execute a block of code multiple times based on the condition. The main types of loops are: 1) for Loop, 2) while Loop, 3) do-while Loop, 4) Enhanced for Loop/ nested for loop/ for-each Loop. Ref: Java\_Examples\LoopConditions

**Basic programs on Array, ArrayList, List and Set:** Ref: Java\_Examples/C1\_ArrayPractice\ArrayListPractice\ListPractice\SetPractice

**N:** It is possible to call the base method without creating an instance and that method should be static.

**N:** **Recursion** is a programming technique where a method calls itself to solve smaller instances of a problem. It breaks a problem into smaller subproblems until it reaches a base condition. Ref: Java\_Examples\RecursionEx

**@. Methods available in Scanner class:** nextByte(), nextByte(), nextInt(), nextFloat(), nextDouble()….

**@. Different ways to print exception message in console:** e.printStackTrace(), e.getMessage();

**remoneAll()** is used to compare the 2 lists and remove all common values.

**retainAll()** is used to compare both lists and retains only the common values.

**N:** Default package in java is java.lang package

**N: equals()** method is used to compare two objects for equality, while **hashCode()** method returns a unique integer value for each object.

Class extends single class Ref: OOPS\_Abstraction\Why\_Interface\_Intraduced

Class not extends multiple classes

Class implements multiple Interfaces

Interface extends multiple interfaces

Interface not implements multiple Interfaces

Interface not extends multiple classes

**N: Optional Class** helps to prevent NullPointerException by representing optional values (i.e. methods to deal with missing values). Optional Class is a container object introduced in Java 8 (as part of java.util package) that is used to represent a nullable value in a more explicit and safe way. Ref: \Java\_Examples2\OptionalClass

**N: FileHandling** provides classes to read, write, and manipulate files in java.

Types of Java Libraries: It is a collection of reusable java classes and interfaces. Ex: Apache commons, JUnit…

**@. Hidden and Disable WebElements** Ref: WD\_Examples\EX43b\_JsE\_HandlingHiddenElements\_nd\_PasteSytemClipboardCopiedDataIntoEclipseConsole **/** EX43c\_JsE\_HandlingDisabledElements

In case of **Hidden** element, node will present on the DOM but element will not display/visible on web page so we can’t able to interact with that element. Status of **isEnabled()**-true (bcz ‘disabled’ attribute is not set in node), **isDisplayed()**-false.

In case of **Disable** element, node will present on the DOM and element will display/visible in web page but we can’t able to interact with that element. In that node some where we can see ‘disabled’ attribute name for disable elements. Status of **isEnabled()**-false (bcz ‘disabled’ attribute is set in node), **isDisplayed()**-true.

Both the cases we will get ‘ElementNotInteractableException’.

@. To use JavaScript execution to interact with hidden or disabled elements?

In Selenium, sometimes elements are hidden (not visible in DOM) or disabled (read-only/unclickable), and normal WebDriver actions like click() or sendKeys() don’t work. In such cases, we can use JavaScriptExecutor to directly interact with the element.

1. Click on a Hidden/Overlapped Element --> js.executeScript("arguments[0].click();", element);

2. Set Value in a Disabled/Hidden Input Field --> js.executeScript("arguments[0].value='testUser';", element);

3. Remove disabled Attribute and Interact Normally --> js.executeScript("arguments[0].removeAttribute('disabled');", element);

4. Make Hidden Element Visible Temporarily --> js.executeScript("arguments[0].style.display='block';", element);

5. Scroll Into View (if element is off-screen) --> js.executeScript("arguments[0].scrollIntoView(true);", element);

6. Trigger Events (like onchange, mouseover) --> js.executeScript("arguments[0].value='Option2'; arguments[0].dispatchEvent(new Event('change'));", element);

**@. Loop to use**

for loop: Use when the number of iterations is known.

while loop: Use when we want to repeat until a condition changes.

do-while loop: Use when we want to print the statement inside the ‘do’ without checking the condition on first time.

Naming conventions:

Class Names: Use pascal case (Ex: Student, EmployeeDetails)

Variable and Method Names: Use camel case (Ex: studentName, calculateTotal)

**N:** We can use both ‘extends’ and ‘implements’ keywords in single class signature.

**@. Suppose there is one method in interface as private and a class is implementing. Can implemented method be public or vice versa?** ----- In interface only public, abstract, default, static modifiers are permitted

**@. Dynamic array in java**

In Java, a dynamic array is an array that can grow or shrink in size during runtime. Unlike a regular array, which has a fixed size, a dynamic array adjusts its size as elements are added or removed. Common Dynamic Array in Java: ArrayList

**@. C we write any code after throw statement**

After throw statement JVM stop execution and subsequent statements are not executed. If we try to write any statement after throw we do get compile-time error saying unreachable code.

**N: System.out;** Regular output**, System.err;** Error output; **System.in;** input (reading data from the user)

@. Right datatype to represents price in java ---- BigDecimal Ref: Java\_Examples\PriceRepresents\_usingDataType

**N:** readlIne() returns null when it has reached the end of a file.

@. Y there is no Global variables in java

GVs are globally accessible. Java doesn’t support GVs because of its design philosophy around encapsulation, security, and object-orientation and also GVs breaks the referential transference and GVs creates collisions in namespace.

@. Method

It contains executable body that can be applied to specific object of the class. A method includes method name, parameters or arguments and return type. Methods may have multiple arguments separated with commas.

@. Null means in java --- When a reference variable doesn’t point to any value it is assigned null

@. Where variables are created in memory

Variables are created in stack. So when the variable is out of scope those variables get Garbage collected.

@. Can a variable be Local and static at the same time --- No, defining LV as static it gives compilation error.

@. H Objects are stored in java

In java, each object when created gets memory space from heap. When an object is destroyed by a garbage collector, the space allocated to it from the heap is re-allocated to the heap and become available for any new objects.

@. Can we cast any other type (Primitive type) to Boolean type with typecasting? --- No, and wise versa also not.

@. State some situations where exceptions may rise in java

* Accessing an element that does not exist in Array.
* In valid conversion of number to string and string to number (NumberFOrmatException).
* Invalid casting of class (ClassCastException).
* Trying to create an object for Interface and Abstract class (InstantiationException).

@. Y prefer ArrayList in Selenium

Works with findElements(), store dynamic WebElements and fast index based access for loops.

@. To store elements from findElements()

Use of List<WebElement> to loops through elements



@. To map test steps or locators ---- Use Map<String, By> to map locator keys to locators

@. Where Set is used in selenium ---- For window Handles and to validate uniqueness in dropdowns

@. To manage test data in DDT ---- Ref: WD\_Examples\EX78\_DDT\_or\_UseOf\_ListAndMap\_Together

**Map<String, String>** for single row of test data (column name -> value)

**List<Map<String, String>>** for multiple rows of test data

Works well when reading data from excel, JSON, or CSV

@. To remove duplicates from test data ---- Store values in a Set to auto remove duplicates.

@. To remove duplicates from List - List<String> unique = new ArrayList<>(new HashSet<>(originalList)); Ref: Java\_Examples\RemoveDuplicates\_from\_List

@. To maintain insertion order in tests ---- Use LinkedHashSet or LinkedHashMap

@. Java and JavaScript

* Java is OOP language and JavaScript is an OO Scripting language
* Java will used for backend automation (Selenium, TestNG) whereas JS will used for frontend automation (Cypress)
* Java creates applications that runs in VM or browser by JVM while JavaScript code is run on a browser only
* Java code needs to be compiled while JavaScript code are all in text.

@. Ways to create object

Using: new keyword, new instance, clone() method, deserialization and newInstance() method of constructor class.

@. Java Array

An Array is a data structure that can store a fixed-size sequence of elements of the same data type. Array is an object in java, which means it can be assigned to a variable, passed as a parameter to a method, and returned as a value from a method.

@. H objects in a java class be prevented from serialization Ref: Java\_Examples\TransientKeyword3

Serialization converts an object into a byte stream for storage or transmission, to prevent serialization:

* Declare fields as **transient** to exclude them from serialization.
* Implement **writeObject()** and **readObject()** methods to control serialization.
* Extend **NotSerializableException** to explicitly prevent serialization.

@. Object cloning

Object cloning is a way to create an exact copy of an object. Java provides the clone() method from the Clonable interface to perform shallow copies. A shallow copy copies field values but does not duplicate referenced objects, while a deep copy creates new instances of referenced objects.

@. Memory leak in java / H java handle memory leaks

Memory leak occurs when object that are no longer needed are not garbage collected because they are still referenced somewhere. This can cause excessive memory consumption and slow down the operation. Java uses garbage collection to handle memory leaks.

@. new and newInstance()

new keyword is used to create a new object of a known class at compile-time.

newInstance() (from class) creates an object dynamically at runtime, requiring reflection, and is slower because it involves additional security and access checks.

@. Abstraction and Encapsulation

Abstraction hides implementation details and expose only essential functionalities by using Abstract class and Interface.

Encapsulation bundles data and methods within a class and restricts direct access using access modifiers.

@. Purpose of default keyword in interfaces

The default keyword allows to add methods to interfaces without breaking existing implementations.

@. Path and Class path

Path specifies the location of system executables like java and javac. Path is used by OS.

Class path specifies the location of java class files, JARs, and other resources. Class path is used by the JVM.

@. W happens if two keys have the same hash code in a HashMap

* Both keys go to the same bucket
* HM uses equals() to resolve the key collision
* If keys are equal, value is updated
* If keys are different, a linked list or tree stores them.

@. Y Map is not a part of Collection Interface

* Collection represents a group of elements (single values)
* Map represents key-value pairs, which is a different structure

**@. equals() and hashCode()** Ref: Java\_Examples\equals\_nd\_hashCode\_diff

equals() used to compare the actual content (or state) of two objects.

hashCode() returns an integer hash code for an object, used in hash-based collections (like HM, HS, HT) to quickly locate objects.

@. Structure of a maven-based automation project

Typical folders: src/main/java, src/test/java, pom.xml, tetng.xml, and resources.

@. Utility/helper class --- Reusable methods like waits, data functions, or common validations stored in one class.

@. Java Memory Model

JMM defines how threads interact through memory, ensuring visibility and ordering of variable access.

@. Design patterns

Design patterns are reusable solutions to common software design problems. Ex: Sigleton, Factory, Observer.

**N:** **Thread safety** ensures that piece of code or object can be accessed by multiple threads without causing data corruption or unexpected behavior.

@. To read and write to a file in java

Reading and Writing to a file can be done using FileReader, FileWriter, BufferedReader, and BuffredWriter classes.

@. FileReader and BufferedReader

FileReader is used to read data from a file as a stream of bytes.

BufferedReader is used to read the text from a character-based input stream, buffering characters for efficient reading.

@. FileInputStream and FileOutputStream

FileInputStream is used to read data from a file as a stream of bytes.

FileOutputStream is used to write data to a file as a stream of bytes.

**N: File class** is used to represent the files and directories of the local file system. It can be used to create, delete, and query file information.

**N: exists()** method of File class can be used to check whether the file or directory exists.

@. Classloader

Classloader is a part of JRE that loads classes into memory when required during the execution of a program. It is responsible for dynamically loading, linking, and initializing classes and interfaces. Java uses 3 main type of class loaders: Bootstrap Class loader, Extension Class loader, and System/Application Class loader.

**N: Dot operator** is used to access the instance variables and methods of class objects.

**@. W happens if don’t provide constructor in a class**

If we don’t provide constructor in a class, the compiler automatically generates a default constructor with no arguments and no operation which is a default constructor.

@. To manage cookies in Selenium?

We can manage cookies using several methods. Selenium provides methods to manage cookies:

Add a cookie: driver.manage().addCookie(new Cookie("key", "value"));

Delete a cookie: driver.manage().deleteCookieNamed("key");

Get a cookie: Cookie cookie = driver.manage().getCookieNamed("key");

Delete all cookies: driver.manage().deleteAllCookies();

@. Java 8 features for functional programming

Java 8 introduces several features to support functional programming, including lambda expressions, functional interfaces, the Stream API for working with collections, and default methods in Interfaces.

**N: Stream API** will Processes data in a functional style using map, filter, and reduce operations.

@. BufferedReader and Scanner class

BufferedReader in java reads text from character stream with efficient buffering, while a Scanner class can parse different types of data from various sources such as files, String, or standard input.

@. public class and default (package-private) class

public class can be accessible from any other class, regardless of the package they belong to. A default class, also known as package-private class, is only accessible with in the same package and cannot be accessed from outside the package.

@. Character class

Character class provide methods for working with individual characters, such as checking for character types (letters, digits, whitespace), converting case, and performing character-based operations.

@. Integer class

Integer class is a wrapper class that provide methods for working with integer values, such as converting strings to integers, performing arithmetic operations, and converting integer to different representations (binary, hexadecimal).

@. Purpose of import keyword

import keyword in java used to import classes, interfaces, or packages into a source file. It allows using classes from other packages without specifying their fully qualified names.

@. Which keyword is used to define a block of multiple options in java --- switch

@. Which operator is called the ternary operator --- ?:

@. Which is not a valid identifier in java --- myValue, \_count, 123abc, $price

@. Default value of reference variable in java --- null

@. Advantages of using Java for automation testing?

Java is widely used in automation testing because:

* It is platform-independent and runs on various operating systems.
* It supports multiple testing frameworks like Selenium, TestNG, and JUnit.
* It has strong community support and a rich set of libraries for automation.
* Java provides robust exception handling and multithreading capabilities, making test execution faster.

@. Java collections, and which collection types are commonly used in automation testing?

Java Collections is a fw that provides data structures like lists, sets, and maps. Commonly used collections in aut testing:

List (ArrayList, LinkedList) – Storing test data dynamically.

Set (HashSet, TreeSet) – Storing unique test data.

Map (HashMap, TreeMap) – Storing key-value pairs (e.g., configuration settings).

@. Role of synchronization in multithreading? To implement it in Java?

Synchronization prevents multiple threads from accessing the same resource simultaneously, avoiding race conditions. Synchronization is useful in test automation when handling shared resources like test reports.

**Ex:**

class SharedResource {

synchronized void display() {

System.out.println(“Synchronized Method”);

}

}

end#1

//Java int coding qns : #1 Reverse a String

//Java int coding qns : #2 Check for Palindrome

//Java int coding qns : #3 Fobonacci series

//Java int coding qns : #4 Factorial of a Number

//Java int coding qns : #5 Prime Number Check

//Java int coding qns : #6 Count Vowels And Constants

//Java int coding qns : #7 Sort An Array using inbuilt method

//Java int coding qns : #7b Sort An Array Without using inbuilt method

//Java int coding qns : #8 Merge Two Arrays

//Java int coding qns : #9 Largest Element In Array

//Java int coding qns : #10 Largest Element In Array

//Java int coding qns : #10b Remove Duplicates from ArrayList

//Java int coding qns : #11 Check Armstrong Number or Not

//Java int coding qns : #12 Reverse a Number

//Java int coding qns : #13 GCD of two numbers

//Java int coding qns : #14 Check Anegram

//Java int coding qns : #15 Count DIgits in Integer

//Java int coding qns : #15b Some Of DIgits in Integer

//Java int coding qns : #17 Second Largest Element in an Array

//Java int coding qns : #18 Swap two Numbers

//Java int coding qns : #19 Pascal's Triangle

//Java int coding qns : #20 Find the missing Number in An Array

//Java int coding qns : #21 Convert Decimal to Binary

//Java int coding qns : #22 Check for Perfect Number

//Java int coding qns : #23 Remove Spaces From String

//Java int coding qns : #24 Find Common Elements in Arrays

//Java int coding qns : #25 Find First and Last element in Array

//Java int coding qns : #26 Number of occurrence of given char in a string

//Java int coding qns : #26b Each char occurrence from given string

//Java int coding qns : #27 Number of occurrence of given word in a string

//Java int coding qns : #27b Each word occurrence from a given string

//Java int coding qns : #28 Split an Alpha Numeric digit without using split method

//Java int coding qns : #29 Java program to move all zeros to left \*\*\*\*

//Java int coding qns : #30 Java program to Add or Sum Array Elements

//Java int coding qns : #31 Java program to Move Even to Left

//Java int coding qns : #32 java program to find max and min with IF, you can use collection

//Java int coding qns : #33 java program to compress String a2b3c4d2 --> aabbbccccdd and aabbbccccdd --> a2b3c4d2

//Java int coding qns : #34 java program to find duplicate characters from a word without using loop

//Java int coding qns : #35 java program to find and print the most frequently occurring prefix in a given array of strings

**SELENIUM**

**@. Automation? Advantages & Disadvantages**

The process of converting the manual Test Cases into Test scripts and execute these scripts by using any automation tool is known as Automation

|  |  |
| --- | --- |
| **Adv**: ---- frctcabqrr | **Dis-adv:** |
| 1. Fast in execution 2. More reliability 3. More consistence (any time can execute script) 4. It saves time to executing the test cases without manual effort 5. Cost to company (CTC) can save 6. We can maintain accuracy by repeating the same task in same manner 7. Bugs can identify easily and we can report the bugs to the developer 8. We can ensure for quality 9. Automation test scripts are reusable on different versions of applications 10. Automation test scripts are repeatable so we can execute multiple times | 1. Automation tools are expensive 2. Skilled automation TEs are required 3. While automate with some tools some of the environments or technologies are not possible to automate like MF & Desktop, mobile appl’s by using selenium alone, captcha reading, OTP verification, Barcode reading, Bitmap comp, GUI, Flash elements. |

**@. Challenges in Automation Testing**

High initial cost, UI changes dynamically, dynamic WebElement locator identification, Handling captchas, Test maintenance, and selecting the right TCs for Aut, Multiple application integration Test Scenarios, CI/CD integration.

**@. Type of Test Cases should be Automated:** Repetitive tests, Regression tests, smoke tests and high-risk Test Cases.

**@. Type of TCs should not be Automated:** Exploratory tests, UI-based usability tests, and frequently changing TCs

**@. Selenium / and composed of / Diff forms of selenium / Components in Selenium / Structure of selenium**

Primarily selenium was introduced by **Jason Huggins** in **2004**. Selenium is an open source frontend functional testing robust automation tool / framework. It is a suite of tools and it is used for automating web application testing by directly interacting with web browsers across different browsers and platforms. It supports multiple programming languages. Selenium is composed of:

**Selenium IDE:**IDE stands for Integrated Development Environment. It is a browser extension and it is developed to speed up the creation of automation scripts by recording the user actions on the web browser using existing selenium commands and exporting them as a reusable script. It supports Java, Python, C#, Ruby, and JavaScript languages. Now it is a plugin for Firefox, Chrome and MS Edge. Recorded script can run against other browsers by using Selenium IDE environment or selenium WebDriver.

**Selenium RC:**  Selenium IDE have some limitations in terms of browser support and language support to overcome that limitations selenium RC was introduced. RC stands for Remote Control. It is a server that enables users to generate test scripts in their preferred programming language. It has 2 components those are Selenium-Server and Client-Libraries. Here Selenium-Server is responsible for launching and killing the browser and run the selenium commands that are sent from test program and Client-Libraries provides an interface between Selenium-Server and programing language.

**Selenium WD / Advanced Selenium:** The functionality of WebDriver is to developing and executing the Automation scripts. It is used for automating web applications by using browsers native support or browser built-in support.

**Selenium Grid:**It is for parallel execution. Here we can launch all the browsers parallelly and scripts can be executed on it parallelly. With the help of the Grid, we can distribute tests on multiple machines so that the test exe time can save.

**@. Y selenium is one API / Is selenium is a tool or suite**

Selenium is not a tool it is a suite with components like selenium IDE, RC, WD and Grid. And selenium is not having any .exe file to install selenium it is just library or API. To access selenium, we have to add selenium libraries to our project then only we can able to access all WebDriver methods that is the reason we are getting consider as selenium is an API.

**@. Y should we go for Selenium instead of QTP / H Selenium is different from commercial browser automation tools**

1. First of all, Selenium is an opensource web application automation tool
2. It supports multiple languages like Java, Ruby, PHP, python, C#, pascal, Perl (scripting language)
3. It supports multiple browsers like Firefox, IE, Google chrome, edge, Safari and Opera
4. It is a platform independent and it supports multiple OSs like Windows, Linux and MacOS
5. It’s supports Web Applications by using browsers native support or browser built-in support.
6. 3rd Party Integration: Selenium integrates with tools like JUnit & TestNG for better test management and reporting.
7. It is very flexible and extendable

**@. Aut selection criteria for proj / On wt basis we will do Aut / key factors consider before automating TCs -** rre scrsd

1. More releases are expecting
2. More resource requires and time taken by doing manual testing
3. Application environment should support for automation
4. Application should be stable
5. Client accepts in terms of budget
6. Return on investment (ROI)
7. Size of the project (i.e., the application should contain more modules)
8. Test Case Repeatability
9. Data-driven scenarios are present

**@. Latest versions:** Using stable versions maven-4.0.0 selenium-java-3.141.59/4.9.0.jars, java - 21

**@. Locators available in selenium WD / Type of locators** / **various ways used for locating WebElements in selenium**

In Selenium, elements within a web page can be located using various strategies, each suitable for different scenarios. Locators are used to identify and interact with WebElements on a web page. Selenium supports various locator strategies, such as id, name, linkText, partialLinkText, className, tagName (is used to locate elements by their HTML tag), XPath and cssSelector.

**N:** If the WebElement has ‘id’ value usually use ‘id’ else I go with XPath or cssSelector. ‘id’ values are unique and directly mapped so it is the fastest way to locate WebElements.

@. Locators priority Order I Follow in My Project

|  |  |  |
| --- | --- | --- |
| **Pri** | **Locator Type** | **Reason for Priority** |
| 1st | id | Fastest and most reliable (unique, browser-optimized) |
| 2nd | name | If ‘id’ not available, and name is unique and stable |
| 3rd | CSS Selector | Clean, readable, and performance wise better than XPath; ideal when id/name missing |
| 4th | XPath | Used only if other attributes are dynamic or missing; allows complex DOM traversal |
| 5th | LinkText / PartialLinkText | Used for hyperlinks when text is static |
| 6th | ClassName / TagName | Least preferred, often not unique |

**@. To capture screen shot using Selenium WebDriver**

We can use the TakeScreenshot fun to capture the screenshot and getScreenshotAs() command to save the screenshot.

File myFile=((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE); -- To capture screen shot of page (java.io.File)

FileHandler.copy(myFile, new File(“path of file.png”)); ----- To save image in a specific location (selenium.io.FileHandler)

FileUtils.copyFile(myFile, new File("path of file.png")); ----- Ref: WD\_Examples\EX40b\_FullPageScreenshot\_Using\_AShot**,** EX40c\_WebElementScreenshot

**N:** It takes a screenshot of web page visible area only not full page to take full page we have to use **AShot** library (3rd) or use getFullPageScreenshotAs() in Firefox browser only.

**N:** Output type formats available for screenshot: outputType.FILE, outputType.BYTES, and outputType.BASE64

**@. To capture WebElement screenshot** ---- Use getScreenshotAs() on WebElement (Selenium 4 feature)

**@. Challenges / Difficulties have faced with Selenium -----** mhfptdpp

1. Maintaining synchronization is a challenge in Selenium.
2. Handling with MF, Desktop, CR (are designed to prevent automation), OTP verification, GUI, flaky tests, Flash objects is challenge.
3. Frame tags involved in a page is a challenge.
4. Proving user defined Results is a challenge
5. Taking data from the application is a challenge
6. Handling dynamic UI elements is a challenge
7. Handling dependencies is a challenge.
8. Handling with multiple popup windows is a challenge
9. Page load synchronization is a challenge

**@. Technical challenges / Limitations / Dis-Advantages of using Selenium as a testing tool** --wbrvo

1. Selenium supports only web-based applications it doesn’t supports desktop or window-based applications but it can overcome by using the tools SIKULI or AutoIT or Robot class.
2. We can’t test webservices using selenium.
3. Bitmap comparison is not possible with selenium.
4. Lack of Built-In Reporting (For any reporting related capabilities, we have to depends upon third party tools like QC, Bugzilla, Jira) & Test Management.
5. There is no vendor support for this tool compared to commercial tools like UFT.
6. By default, in-built object repository concept is not possible with selenium. However, object repositories can be built using the key-value pair approach wherein key refers to the name of the given object and value refers to the properties used to unique identification of an object within the web page.
7. For creating robust scripts in selenium programming knowledge is required.
8. We have to depends on some external libraries or tools for performing tasks like – logging (Log4j), testing framework (TestNG), reading from external file (Apache POI for excel), etc.
9. Programming knowledge is required.

**N:** Images are stored in series of tiny dots (it’s nothing but pixels) is called **Bitmap** or **raster**. WebDriver does not provide direct any function for **Image comparison** but we can verify images by taking two screenshots of the whole web page, one at **script creation time** and another at **script execution time**. After that these screenshots can compare manually.

**@. SSL (**Secure Sockets Layer**) certificate error in selenium / To handle** Ref: WD\_Examples\EX51\_HandleSSLError

**SSL** is a standard security protocol for establishing the secure connection between server and client. Browser and server use **SSL Certificate** mechanism to able to establish a secure connection.

**@. Synchronization in selenium**

**Synchronization** is a mechanism which involves two or more components are working parallel with each other. Usually, in automation we have two components such as **Application under test** and **test Automation tool** both of them can have specified speeds and the test scripts should be written in a way such that both these components will work with same speed. This will help to avoid “NoSuchElementException” otherwise will consume more time to clear off here the synchronization will come for help.

**N:** In Selenium, there is no official “ElementNotFoundException” the correct exception is “NoSuchElementException” (It occurs when Selenium cannot find a web element on the page using the locator provided).

**@. Purpose of using waits in selenium**

Sometimes it takes longer time for a WebElement to load/appear. At that time our tests might throw NoSuchEleExcep, to handle this case we will use selenium waits.

**@. Types of XPaths are avl:** On the basis of node hierarchy XPath can div into 2 types those are Abs XPath & Rel XPath

**@. Categories of Synchronization in test Automation / Dynamic wait times / Type of waits in selenium**

**Unconditional Synchronization:** Here only the **timeout** value to be specified so that the tool will wait until the specified time before it proceeding. Where it is not possible to write the condition or check for the condition. The major dis-adv in sometimes, the tool will wait unnecessarily even when the WebElement is ready so it reduces script efficiency because of this it is not recommended due to unnecessary delays. **Ex:** Thread.sleep(1000)

**Conditional Synchronization:** Here a **condition** also specified along with **timeout** value because of this the tool will wait to check the condition and will come out if nothing happened. In selenium there are 3 diff types of conditional statements are available to avoid synchronization problems those are **Implicit wait**, **Explicit wait** and **Fluent wait**.

**Implicit Wait** Implicit wait set a default timeout for all elements if element is not readily available. It is applicable to all the WebElements from where it is specified so that the specified amount of time it will try looking for element before throwing the Exception. It throws NoSuchElementExcption. Applicable to simple applications with consistent load times. It slows down tests if overused. Implicit Wait applies globally.

**N:** The default time for Implicit wait is ‘zero’ and polling frequency is 500 milli.sec.

**Syn:** driver.manage().timeOuts().implicitlywait(Duration.ofSeconds(10));

**Explicit Wait / WebDriverWait** allow to specify custom condition along with timeout value for specific WebElement. Condition should be satisfied within the specified timeout period then only code will be executed. It is not applicable to all WebElements, applicable only for specified element. It throws ElementNotVisibleExcp. Applicable to dynamic or complex loading scenarios. Explicit Wait and WebDriverWait waits for specific condition.

**Syn:** WebDriverWait wt = new WebDriverWait(driver, Duration.ofSeconds(10)); ---- selenium.support.ui.WebDriverWait

wt.until(ExpectedConditions.visibility/presence OfElementLocated(By.LN(“LV”))/.alertIsPresent()); ---- \*.EC

**FluentWait** is an implementation of **Wait** interface. FluentWait instance is used to define maximum amount of time to wait for a condition (timeout value) as well as the frequency to check the conditions (polling interval). Using this type, certain types of exceptions (such as “NoSuchElementException, StaleElementReferenceException, etc”) can be ignored. It is not applicable to all WebElements, applicable only for specified element. Applicable to highly dynamic elemnts.

**Syn:** For Ex here the wait time can set to maximum of 30 sec, with a polling interval of 5 sec between each check.

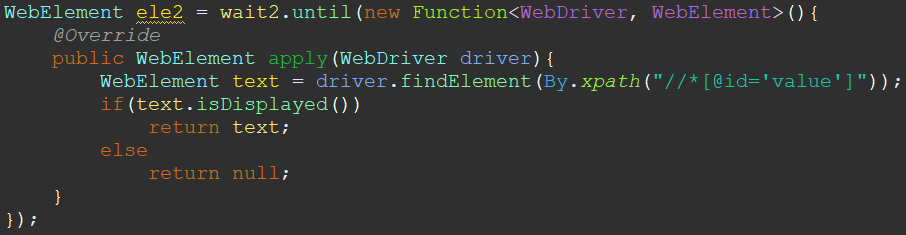
Wait<WebDriver> wait = new FluentWait<WebDriver>(driver)

.withTimeout(Duration.ofSeconds(30))

.pollingEvery(Duration.ofSeconds(5)) (defines how frequently the fluent wait should check the condition)

.ignoring(NoSuchElementExcp.class, StaleElementReferenceExcp.class, …);

wait.until(ExpectedConditions.visibilityOfElementLocated(By.LN("LV")));

(or)

**N:** If ‘elementNotVisible’ exception is still thrown .ignoring(‘NoSuchElementException.class’) will works.

**N:** If we use both imp, exp and sleep() waits at a time it’s not good practice bcz of this our t-script will wait more time so it causes timeout conflicts or unpredictable delays.

**N:** Default polling interval of WebDriverWait is 500milli.sec.

presenceOfElementLocated() → element is in DOM and may or may not be visible

visibilityOfElementLocated() → element is in DOM and visible

**@. WebDriverWait and FluentWait** ---- FluentWait allows custom polling intervals and ignores exceptions.

**@. W happens if the wait times out** ---- It throws a TimeoutException

**@. Single and Double slash in XPath**

|  |  |
| --- | --- |
| **Single slash ( / )** | **Double slash ( // )** |
| Absolute XPath starts with single forward slash ( / ) | Relative XPath starts with double forward slash ( // ) |
| ( / ) starts the path selection from root element <html> to desired WebElement | ( // ) Starts the path from desired WebElement (i.e. any middle node that we've selected) |
| Single slash ( / ) is always navigate between the tags |  |

**@. Handle Alerts / JavaScript pop-ups / JavaScript Alerts / Confirmation Alerts in WD** Ref: WD\_Examples\EX26a\_JS\_AlertHandling

An alert could be triggered in a web application that involved submitting a form, depending on the data entered. An alert message contained, confirming the successful submission or notifying about validation errors. Alert interface is used to handle JavaScript alerts or pop-ups. Along with .accept(), .dismiss(), .getText(), .sendKeys().

//first we have to navigate to Alert window by using the command -------Alert alert=driver.switchTo().alert();

//To click **OK** or **Cancel** on Alert or Confirmation Alert by using the commands

alert.accept(); — To click OK

alert.dismiss(); — To click Cancel

alert.getText(); — To retrieve the alert text

alert.sendKeys(“input text”); — To send input to prompt alert / used to handle prompt dialogue

alertIsPresent() — is used for confirmation of alert.

**N:** If we use both accept and dismiss simultaneously, we get NoAlertPresentException

**N:** We can handle file upload po-pup (file explorer 'Open' window) by using Robot class mtds keyPress(), keyRelease() with Up-Down arrows. Ref: WD\_Examples\Ex27\_Fileuplod\_using\_RobotClass\_KB\_Events\_r\_Operators |||| https://www.youtube.com/watch?v=l0YWFT-aUBQ&ab\_channel=SurendraJaganadam

**@. Best wait**

* Explicit wait is generally best for automation testing due to: targeted approach, reduced delays and flexibile for dynamic applications.
* Fluent wait is deal for scenarios requiring advanced polling and exception handling.

**@. Handle popups in WebDriver / Is it possible to handle multiple popups in selenium** ----- questionpopup

//To navigate from main window to popup -------- driver.switchTo().window(“popup\_Window\_Name”);

//To navigate from Popup to main window -------- driver.switchTo().window(“Main\_Window\_Name”);

//method 2

Yes, First of all, take all the window names into ‘Set<String>’ **variable** with the help of driver.getWindowHandles(); its return type is String and convert it to **ArrayList**.

Use the Array index to navigate to specific window by using driver.switchTo().window(ArrayList\_Object.get(Index));

**@. To launch FF, Safari and Opera with WebDriver / H can we launch different browsers in selenium WD**

To launch any browser first we need the supported Driver executable file. It can download from, selenium dev page. After that set browser property by System.setProperty(“webdriver.chrome.driver”, “pathofchromedriver.exe”); - (k, v)

After that we have to create a driver instance of WebDriver with ChromeDriver object then only we can able to access both WebDriver methods as well as ChromeDriver methods. WebDriver driver = new ChromeDriver(); (--- RP concept ---)

**@. Name an API used for reading and writing data to excel files / Significance of Apache POI**

**Apache POI** API is an open-source java library it is used to create, Read, Write and Update excel files.

**@. To capture and validate page**/**window title**/**name**

**driver.getTitle().equals(“ExpTitle”)** method can be used to capture capture and validate page/window title/name

**@. To select the 2nd item in a List box or drop down** ----selectByIndex(index); index can start from zero

**@. How to maximize the browser in WebDriver** ------- driver.manage().window().maximize();

**N:** **Select** class is used to select an option from the single and multi-select dropdowns. It has methods like getOptions(), **getAllSelectedOptions()**(return type is list of WEs), getFirstSelectedOption() (return type is single WE)…..

**N:** **isMultiple()** is used to check whether a select element (dropdown) allows multiple selections or not. Ref: WD\_Examples\EX11a\_MultiSelectDropdown

**N: deselect** methods are used to deselect options from a multi-select dropdown ------- deselectByIndex(index), deselectByValue(value), deselectByVisibleText(text) and deselectAll()

**@. To Handle AJAX controls / AJAX calls / AJAX content loading / AJAX-based web applications using selenium (Ex. By typing in search engine h to cap the auto suggestions)**

AJAX stands for Asynchronous JavaScript And Xml. It allows the web pages to retrieve small amounts of data from the server without reloading the entire page. Ajax elements load asynchronously. Ajax means for example when we enter some text in the google textbox, it displays the auto suggested values that means that textbox is under AJAX control.The **biggest challenge in handling Ajax call is loading time of the web page.**Since the webpage is loading it will last only in fraction of seconds, it is difficult for the tester to test such application through automation tool. For that, Selenium WebDriver has to use Explicit wait with conditions like visibilityOfElementLocated() to ensure elements are ready before interacting with them to handle Ajax Call. So, by executing this wait command, the script will pause until specified timeout period for the expected value.

We can handle the AJAX calls using explicit waits with ExpectedConditions:

WebDriverWait wait = new WebDriverWait(driver, 10);

wait.until(ExpectedConditions.elementToBeClickable/visibilityOfElementLocated(By.id("ajaxElementIdValue")));

**@. To take specific cell data from excel(xls) file ------ &&&&& ------ #36b. To take or print data from XML file**

//to take/print data from excel first read the excel file up to the sheet. We can use the syntax like,

FileInputStream fi = new FileInputStream(“Path of the excel file”); ---------// to focus on particular file

XSSFWorkbook wb = new XSSFWorkbook(fi); ------- // to open particular workbook //.Xlsx - Excel 2007 OOXML onwards

XSSFSheet ws = wb.getSheet(“sheet\_name”); ---------// to focus on particular sheet

String cell = ws.getCell(columnID, rowID).getContents(); --------- //to take particular cell data from sheet

**@. To handle Flash objects and Desktop in Selenium**

We can use SIKULI to support Desktop, Flash objects and it is an add-on for Selenium. SIKULI is an open-source automation tool and this script can be developed by using JAVA.

|  |  |
| --- | --- |
| **@. To print data from notepad (txt file)** | **#40. To create and write data into txt file** |
| File f = new File(“E:\\data2.txt”); // to focus on parti file  FileReader fr = new FileReader(f);  BufferedReader br = new BufferedReader(fr);  String str;  while((str=br.readLine())! = null) {  system.out.println(str);  }  Ref: Java\_Examples2/Check\_Whether\_Perfect\_String\_Int/IntCheck\_checkIntOrNot\_UsingBufferedReader | File f = new File(“E:\\data2.txt”); // to create file on spec loc  FileWriter fw = new FileWriter(f);  BufferedWriter bw = new BufferedWriter(fw);  bw.write(“Name”);  bw.newLine(); --------- // for next line  bw.write(“City”);  bw.close();  fw.close(); |

**Build:** Build is nothing but an executable code. It is a process of converting the source code into software system. There are number of steps to convert the source code to executable code those are,

* 1. Taking the source code from src repository
  2. Preparing a build area
  3. Compiling the source code
  4. Build the compiled code to executable code

**@. Maven**

* Maven is java-based build automation project management tool. It provides a standard way to manage project dependencies, build processes (to build the code), and project lifecycle tasks.
* Once we build the code we will get (output) .jar (java archive file)/.war (web archive)/.ear (enterprise archive file)
* Maven is used to add the dependencies to our project. Maven will work based on POM.xml

**Adv:**

**Dependency management:** Maven simplifies the management of project dependencies by automatically downloading and including required libraries.

**Consistent build process:** Maven standardizes the build process across different projects and environments, making it easier to maintain and share code.

**Convention over configuration:** Maven follows conventions and defaults, reducing the need for manual configuration and allowing developers to focus on writing code.

**Centralized repository:** Maven Central Repository serves as a central repository for sharing and distributing libraries, making it easy to find and use third-party dependencies.

**@. Ant or Apache Ant / Maven / Ant and Maven** ---- vpfpLr

Ant and Maven is Java based build automation project management tools used to build the code. Ant will manage all the steps through **Build.xml** file. Whereas Maven through **pom.xml** file. (xml - eXtensible Markup Language)

1. Maven is advanced version of Ant. (pom – Project Object Model)
2. Maven is widely preferred than Ant because Ant is an older tool.
3. Maven follows strict conventions, Ant allows to define their own conventions.

**N:** Convention refers predefined, standardized practices and configurations that help guide and simplify the process of building, managing dependencies, and packaging software

1. Ant is procedural it means in Ant, we have to specify the order what should have to be done, whereas as Maven is declarative it means Maven takes care of all the directories that stored in the pom.xml file.
2. Ant doesn’t have a life cycle whereas Maven has a life cycle.
3. The scripts in Ant are not reusable whereas Maven comes with reusable plugins.

**@. Dependencies in Maven**

Dependencies are external libraries or modules required by a project to compile, build, or run. They are declared in the POM file using <dependency> elements, along with coordinates such as group id, artifact id, and version.

**@. Verify and Assert Commands**

"assert," "verify," and "waitFor" are the three kinds of Assertions.

**Verify** (verifyText, verifyTitle, verifyElementPresent, etc.) command is belongs to Selenium IDE and Selenium RC. It will check whether the element is present on the page. If the verified element is not available, verify command will not stop execution of Test Case. It will log an error and proceed with execution of rest of the Test Cases. In verification, all the commands are going to run guaranteed even when the test fails (it Verify the condition true or false)

**Assert** command is belongs to selenium WD. It will check whether the element is present on the page. If the asserted element is not available, assert command will stop execution of Test Case. It will log an error and not proceed with execution of rest of the TCs. In assert, the test will be terminated at the point where check fails. (it checks the cond)

**@. Selenium RC and WebDriver / Advantages of WebDriver over Selenium Server** ----bss ad

* + 1. With the WebDriver we can access all the latest browsers. Selenium WebDriver makes direct calls to browser by using each **browser native / built-in support** for automation, while using Selenium RC, it’s requires selenium-server to inject JavaScript into the browser while executing the program.
    2. If we are using Selenium-WebDriver, we don’t need the Selenium-Server it is using totally different technology
    3. WebDriver by default maintains page load synchronization, for page refresh we need to handle it.
    4. WebDriver having auto scroll down action into the application while executing the script.
    5. With the WebDriver we can effectively take the data from application.

**@. Your reporting method**

Bug Reporting is always a manual process. Getting the test report from automation tool and analyze the report for bugs. If identified bug, report the bugs to the developer by using any Bug reporting tools like QC, Bugzilla or JIRA.

**@. Bug report**

During testing testers record their observations, findings and documents a defect, including details like summery, steps to reproduce, expected vs actual result, severity / priority and Attachments (logs / screenshots) is called Test record or Bug report.

**@. Defect tracker**

Defect tracker is a tool used to log, track and manage software defects, defect tracking tools are QC, Bugzilla or JIRA.

**@. To use selenium for performance testing**

Performance testing comes under non-functional testing. Selenium is afront end functional testing robust automation tool and it’s not for performa­nce testing but we can generate only the load by using Selenium.

**@. To get all the links in** [**http://google.co.invalidate**](http://google.co.invalidate)

Link means **Anchor tag** is common in any appln. 1st Identify all the links by using **FindElements** and **tagName** attribute and get all the links to **List** variable and take the target link by using **for-loop** and **if** condition.

**@. JUnit and TestNG framework**

1. Both JUnit and TestNGare the open-source Unit testing frameworks supported by selenium.
2. We need to implement all the test methods very independently in Junit and not necessary in TestNG.
3. JUnit contains very limited annotations like @Before, @Test, @After, @Ignore and @RunWith whereas TestNG contains multiple annotations.
4. TestNG supports Grouping, parameterization, parallel execution and supports both java & .net.
5. TestNG provides html result file by default where as in JUnit it won’t provide html result file by default but it can generate XML reports that can then be converted to HTML using additional tools.

**@. If the default port of selenium is busy then which port you use**

Selenium by default uses 4444 port no. If the port is already used by any other server, then we can change it to any other 4-digit no. **Ex:** 5555 or 1234 or 5861

**@. Automation Lifecycle / H to start Automation / Approach / Plan**

Define scope: Identify what needs to be automated.

Select tools: Choose appropriate tools and frameworks (Ex: Selenium, TestNG)

Develop the Script: Write and organize the script.

Set CI/CD: Integrate with CI/CD tools for continuous testing.

Execute and maintain: Run tests, analyze results and maintain the automation suite.

1. First understand the Functional specifications of the project
2. Execute or Review all the manual TCs to get the func knowledge on appl and to identify the TCs for automation.
3. Choose Automation tool
4. require POC for the project
5. Design the framework (folder structure and required files (what kind of files we have to maintain), etc.)
6. Develop the framework it’s nothing but: Preparing the Test Scripts for all the identified Test Cases and Integrate these scripts with Framework for execution.
7. Before executing identify the build changes, if available update the script
8. Execute Test Scripts using different test execution approaches without any Issues. ------follow CV------

**@. Write a program to get all the text boxes in ‘mail.in’ register page using WebDriver**

List<WebElement> str = driver.findElements(By.tagName(“textbox\_tagname”));

for (int i=0; i<str.size()-1; i++) {

System.out.println(str.get(i).getAttribute(“id”));-------- //To print all id values

} **N:** textbox must contain type=’text’ AN=’AV’

**@. To mouseHover on one element by using WebDriver**

Actions act = new Actions(driver);

WebElement ele = driver.findElement(By.linkText(“About Us”));

act.moveToElement(ele).build().perform();

**N:** WebElement interface is used to assign or represents webpage WebElements.

**N:** **Actions** class is used to perform advanced user interactions such as, mouse hover, drag and drop, double click and RC operations. Which are not possible with basic click and sendKeys commands.

**Actions (Class) -** Used to build a chain of complex actions

**Action (Interface) -** Represents a single executable action

**@. DesiredCapabilities? H it is useful in terms of Selenium**

The DesiredCapabilities is a series of key-value pairs used to set the browser-specific properties like browserName, browserVersion, platformName, platformVersion, deviceName, automationName, app and Path of the browser driver in the system (good Ex is ‘System.setProperty()’). And also, determine the behaviour of the browser at run time. It is highly useful during cross-browser testing or testing across the different OS. It can useful in terms of Selenium is,

* It is used to configure the driver instance of Selenium WebDriver.
* It is used to handle SSL certification error.
* When we want to run the Test Cases on different browser with different operating systems and versions.

**@. XPath and CSS selector**

Both XPath and CSS selectors are used to navigate XML document to find the WebElements in a web page.

The primary difference between **XPath** and **CSS Selectors** is that, with the **XPath** we can traverse both f**o**rward (d**o**wnwards) and backward (upwards) in DOM, locate a parent element using child element and **select elements based on attributes and text.**

Whereas in **CSS selector** we can traverse only in f**o**rward (d**o**wnwards) in DOM. It supports attributes based on section but has limited capabilities compare to XPath. Although **CSS selector** performance is faster than **XPath**. It has following **Advantages and Dis-Adv over XPath**:

* **XPath is slower than CSS Selector**
* **XPath supports text, while CSS Selector does not**
* **XPath supports index**
* **XPath can move in both forward and backward directions whereas CSS Selector can only move forward**
* **XPath engines are different in each browser but CSS is same for all browsers.**
* **XPath tends to become complex and hence make hard to read.**
* XPath not allow the Ends-with method to find the WebElements whereas CSS will allow this.

However, there are some situations, where we need to use XPath like **while searching for a Parent Element** and it is also useful **to identifying the Dynamic Elements** using XPath Axis. So, CSS Selector is faster but XPath is more flexible.

**@. H much time we can save with automation testing**

It’s always depending on the **Application** and **complexity of TCs** but on average we can save around 50-60% of the time.

**@. To Scroll down and Scroll up in the browser**

We can use javascritExecutor or Actions class to perform scroll down and scroll up activities

// Scrolling down the page till the element is found

((javascriptExecutor)driver.exdecuteScript(“arguments[0].scrollintoview(true);”, ele); **(or)**

Actions act = new Actions(driver);

act.moveToElement(ele);

act.perform();

Actions act = new Actions(driver); (**N:** Actions class is used to simulate input actions from mouse and KB on specific WEs)

act.keyDown(Keys.CONTROL).sendKeys(Keys.END).build().perform(); //To scroll down

act.keyDown(Keys.CONTROL).sendKeys(Keys.HOME).build().perform(); //To scroll up

Ref: act.keyDown(Keys.ALT).keyDown(Keys.SHIFT) .keyDown(Keys.CONTROL).sendKeys('test').perform();

**@. Test types that are supported by Selenium**

Selenium can be used to automate web applications and the test types can be supported into: front-end Functional robust [automation testing](https://www.guru99.com/functional-testing.html), Re-testing, Regression Testing, etc.

**N:** For **post release validation with CI** automation tools could be used: Jenkins, Hudson, Quick Build and Cruise control.

[](https://www.guru99.com/images/SeleniumSuite.png)**@. Selenium 2.0, 3.0 and 4.0**

Selenium RC and WebDriver are consolidated as a single tool in Selenium 2.0. Selenium 3.0 is the latest version from Selenium family. Selenium 3 used JSON wire protocol to interact with real browsers. It was released in 2 Beta versions with some of the added new features.

Selenium 4 introduces a new W3C (World Wide Web Consortium) WebDriver protocal for better compatibility, enhanced support for modern browser features, improved debugging capabilities, selenium 4 also includes a new grid for easier management of test infrastructure.

**@. To find an element using Selenium**

#### In Selenium every object or control in a web page is treated as element, there are different ways to find an element in a web page by using 8 locator identifying mechanisms.

**@. Advantages or capabilities of Selenium** &&&& **Y testers should option for Selenium and not QTP ------** gLd

* First of all, selenium is open-source tool whereas QTP (Quick Test Personal) is a commercial tool.
* It sups mul lang’s like Java, Ruby, PHP, Python, JavaScript, C#, Perl, Groovy, Scala and QTP supports only VB script.
* It supports multiple browsers like FF, CH, IE, edge, safari and opera and QTP supports IE, CH, FF, MS Edge browsers
* It supports different OS like Windows,[Linux (Opera Br)](https://www.guru99.com/unix-linux-tutorial.html)& MacOS (Safari Br) and QTP supports windows only.
* Handling with multiple frames, popups, alerts and multiple browser windows.
* Page navigation and drag & drop.
* Handling with Ajax based UI elements.
* By using selenium grid component, **Distributed testing** can be carried out on remote machines.
* It has powerful methods to locate elements (methods like XPath, CSS selector).
* Selenium Test scripts can able to develop in various IDEs like Eclipse, Visual studio, NetBeans and IntelliJ………

A **Client**-**server Application** is a piece of software that runs on client computer and makes requests to remote server. Many such applications are written in high-level **visual** programming lang. Most business logic reside in thisapplication.

**@. Same origin policy? to avoid same origin policy**

The **“Same Origin Policy”** was introduced for security reason. It ensures the content of any site is never be accessible by a script from another site.  As per this policy, any code loaded within the browser can only operate within that website’s domain. To avoid “Same Origin Policy” **Proxy injection** method is used, in proxy injection mode the Selenium Server acts as a **client configured HTTP proxy**, which sits between the browser and application under test.

**@. Proxy in selenium**

A proxy server can be used to route traffic through a middle layer for testing purpose. We can configure proxies using DesiredCapabilities.

**@. sleep() and** **setSpeed() methods**

Both will delay the speed of execution.

**sleep()** is used to sleep the currently executing **java thread** for the given amount of time. It takes a single argument in integer format. It waits only once at the command given at sleep.

**Ex:** thread.sleep(2000) ---- It will wait for 2sec (-----It is also called static wait-----)

**setSpeed()** is selenium RC Command used to stop the execution of every **selenium command** for a given amount of time. It takes a single argument in integer format and runs each command after setSpeed() delay. This command is useful for **demonstration purpose** or if we are using a **slow web application**.

**Ex:** selenium.setSpeed(“2000”) ---- It will wait for 2 sec – sel RC cmd

**@. Four parameter we have to pass in Selenium** --------Host, Port Number, Browser, URL

**@. To submit a form using Selenium** Ref: WD\_Examples\EX82\_Submit\_A\_Form

**submit()** method will use on any element to perform enter activity - ele.submit();Instead of this we can use click() also.

**@. Which attribute should consider throughout the script in frame for “if no frame Id as well as name”**

driver.findElements(By.xpath(“//iframe”)) will return the list of frames and switch to each and every frame and search for locator which we want.

**@. @FindBy, findElement() & findElements() / Return types of both**

**@FindBy** annotation used in POM design pattern to store element

**findElement()** finds the **first** matching element within the current page using the given “locating mechanism”. It returns a single WebElement. If element is not found it throws NoSuchElementException.

**findElements()** find list of **all** matching elements within the current page using the given “locating mechanism”. It returns a list of WebElements. If elements are not found it will give empty list with size ‘zero’.

**@. Type of Automation Frameworks and H to create those**

**Data Driven Testing / Framework / To perform DDT in selenium:** DDT is a software testing method, it can be performed by reading test data from external sources and stored into variables inside test scripts. DDT allow to execute the TCs on multiple times with different set of test data and we can expect the test output in same table. It is also called Table-driven / Parameterized testing. Input data is stored in single or multiple data sources like .xls, .xml, csv, JSON, or DB.

**Keyword driven framework** uses keywords to represent actions to be performed and allowing non-technical users to create TCss. Basically, **KDF** requires the development of data tables and keywords independent of the test automation these keywords representing an action.  In keyword driven test, the functionality of the application under test is documented in a table as well as step by step instructions for each test.

**Hybrid Framework** is a combination of one or more frameworks. Normally it is associated with POM Design Pattern, DD and Keyword driven frameworks. Using external test data along with predefined keywords.

**Behaviour-Driven Development (BDD)** Focuses on behaviour of the application using plane English language. It is a software development process that encourages collaboration between developers, QA testers and non-technical stakeholders to create a shared under’ng of how the appln should behave. Common BDD tools: Cucumber, SpecFlow….

**@. Cucumber**

Cucumber is an open-source testing framework used for Behavior Driven Development (BDD). It allows writing test cases in plain English using the Gherkin language, which makes it easier for non-technical stakeholders (like business analysts or product owners) to understand test cases.

**BDD:** Behavior-Driven Development, focuses on how the application should behave.

**Gherkin:** Domain-Specific Language (DSL) used in Cucumber to write test cases.

**Feature File:** Contains scenarios written in Gherkin syntax.

**Step Definitions:** Java (or other language) methods that implement the steps in the feature file.

**Runner Class:** Executes the feature files using TestNG or JUnit.

**@. C login into any site if it’s showing any Authentication popup for username and pass / C WD handle Aut popups**

Pass the username and password with URL i.e., driver.get("https://username:password@yourwebsite.com"); (HTTP Basic Authentication popups)

**isEmpty()** is commonly used with String, List, Set, Map, and other collection classes to check whether these String, List, Set, Map, and other collection classes contain no elements or no characters.

**@. To assert ‘text’ of webpage using selenium 2.0**

WebElement element = driver.findElement(By.LN(“LV”)) ------ //first we have to create reference to WebElement

String actText = element.getText(); ------ //To get test from element and stored in String variable

String expText = “Selenium WebDriver”;

Assert.assertEquals(**a**ctText, **e**xpText, “if test fails user message will print”); //To assert text from expected

#### **@. Selenium Grid working**

#### [Selenium Grid](https://www.guru99.com/introduction-to-selenium-grid.html) first send tests to the hub and these tests are redirected to Selenium WebDriver, which launch the browser and run the test with entire test suite. **Adv:**

* It allows running the Test Cases in parallel so that the test execution time can save.
* It allows cross-browser testing.
* It allows cross-platform testing.

**@. H can find Broken Links / images in a page using Selenium WebDriver** (EX49,50) ------- vibuL

Due to existence of broken links, the website reputation gets damaged and there will be a negative impact on business. So, it’s mandatory to find and fix all the broken links before release the application. To find the broken links in a page by using Selenium WebDriver is,

* First Get all the links/image links in the page by using **findElements()** and **tagName(**img/a**)** and assign to **List** var.
* Readall the Attribute values of image/link with the help of loop condition and **getAttribute(**“src/href”**)**
* Create object to **URL** class.
* openConnection to **HttpURLConnection** abstract class with src/href link.
* Next connect to each and every link.
* get HTTP status code in the target page by using httpurlconnection**.getResponseCode()** method
* And get response message using httpurlconnection**.getResoponseMessage()** method

Let’s see some of the HTTP status codes

200 – Valid Link

500 – Internal Server Error

400 – Bad request

401 – Unauthorized

404 – Link not found/Page not found

**N:** **href** (Hypertext REFerence) is an attribute of anchor tag which is used to identify the sections within the document.

**N:** **manage()** provide access to various options and settings of the WebDriver instance.

**N:** In case of manual testing we can use ‘Check My Links’ browser extension (it generates report G: Valid links count R: Invalid links count Y: Skipped links count)

**@. Object Repository**

An Object repository is a centralized storage location for all the objects (WebElements) that are used in test scripts. It provides a way to maintain these objects separately from test scripts. We can create Object Repository using **POM** design patterns with **@FindBy** annotation or will use properties file…… This makes it easier to manage and use these elements in the testing framework, allowing for better organization and reusability in test scripts.

Benefits: Easy updates, Reuse the map, Clear instructions and Be flexible.

**@. While injecting capabilities in WebDriver to perform tests on a browser which is not supported by a WebDriver? Limitation that one can come across** ---Major limitation is that “findElement” command may not work as expected.

**@. Can handle colors, front, location in WebDriver** Ref: WD\_Examples\EX60\_getCssValue, EX54\_HandleColorsWithWD / EX83\_color\_front\_position

Use getCssvalue("background-color") (by sending ‘color ID’ as arg) to get the color of the text/button, background color etc… getCssvalue returns value in ‘rgba’ format like – (255, 255, 255, 1) and we can convert rgba value to hex using Color class (i.e. Color.fromString().asHex();) -> ffffff.

**@. SVG elements** Ref: TestNG\TestNG\_with\_WD\InvocationCount, WD\_Examples\EX62\_SVG\_Elements\_Handling, EX39b\_Read\_Tooltip\_SVG\_eles\_Headless

SVG elements can easily create and can easily edit

**Syntax:** //\*[name()=’tagName’] (or) //\*[local-name()=’tagName’]

**Ex:** Tag to tag navigation – //\*[name()='svg']//\*[name()='rect'] svg, rect - tags

Tag level navigation – //\*[name()='svg' and A\_N='A\_V']

**N:** getAttribute() is used to get the value of tooltip and automate it.

**@. Using WebDriver how you can store** / **input** / **enter a value to text box / without using sendKeys()**

**sendKeys(“**our\_value**”)** method will used to store a value which is in text box. without using sendKeys() we use **JsE**.

Js.executeScript(“document.getElementById(‘id\_value’).value=’Alphabin’;”); --- to send text

**@. Frame / To switch between frames / To move to a particular frame in selenium / To work with frames**

Frames are html elements that embed to another htm page within the main webpage. Must switch to the frame before interacting with elements inside it.

**1. Finding the Frame:**

Right-click check: Right-click on the part of the page you think is a frame. If you see options like "This Frame," "View Frame Source," or "Reload Frame," it's a frame!

Code search: Look for <iframe> tags in the website's code to find all the frames.

**2. Stepping Inside the Frame:** **driver.switchTo().frame()**method is used to switch inside the frames in WebDriver it can takes one of the three possible arguments.

* Select the frame by its **index**.
* Select a frame by its **name or Id** attributes (name = String frame name & id = attribute value of id).
* Switch to frame by **WebElement**: Use any locator method to locate the frame element - Ref: WD\_Examples\EX59 \_iFrame\_Maps\_EReport

**Ex:** WebElement frameoriframeelement = driver.findEelement (By.id("frame-id\_Value"));

driver.switchTo().frame(frameoriframeelement);

**N:** Nested Frames (child frame to parent frame): To handle Nested Frames switch to parent frame first, then child. i.e. driver.switchTo().parentFrame(); Ref: Java\_WD\_Ex\src\WD\_Examples

**@. To Determine Whether an Element is Inside an iframe in Selenium Java**

**1.** To determine if an element is inside an iframe using Selenium Java, right-click on the suspected element in your browser and check if options like "This Frame", "View Frame Source", or "Reload Frame" appear in the context menu; if these options are present, it indicates the element is located within an iframe.

**2.** Using try-catch to Detect if Element is Inside an iframe. Ref: WD\_Examples\EX32c\_Check\_If\_ele\_Is\_Inside\_iframe\_r\_not\_Using\_tryCatch

**@. 5 different exceptions had faced in Selenium WebDriver**

#### Some of the exceptions will occur while using Selenium WebDriver those are:

* WebDriverException: WD related issues like browser connection problem **or** If we try to perform an action when the WebDriver connection is in closed state. (Ex: driver.get(“url”); with incorrect driver setup). Fix: Verify driver compatibility with browser version, or check if browser closed unexpectedly or driver crashed.
* BrowserNotConnectedExcp: Browser unexpectedly disconnects or shuts down before the execution finished.
* NoAlertPresentException
* UnExpectedAlertPresentException
* NoSuchWindow(When we try to switch a win but the targeted win is not found)/Element/Frame/AttributeExcp( When we trying to fetch an ‘attribute value’ but the ‘attribute name’ is not correct)
* SessionNotCreatedException
* TimeoutException
* **InvalidElementStateException:** If element is disabled or wrong action performed, then we will get this exception.
* **IlegalStateException:** is a common Java runtime error that occurs when an operation is invoked at an improper or time in the program's execution or improper initialization. For ex: If we don’t write system.setProperty and try to launching the browser then will get this Exception.
* StaleElementReferenceException

**@. Perform double click LOC – Line of code**

**Actions act = new Actions (driver);**

**WebEelement ele = diver.findElement(By.LN(“LV”));**

**act.doubleClick(ele).build().perform(); // build() - for compile & perform() - to exe the seq of steps/mtds in a LOC**

**@. HTMLUnitDriver ---Or--- Which WebDriver implementation is fastest** Ref: AUTOMATION\ExtentReports\_v5\_2\src\test\java\HTMLUnitDriver\_Practice

HTMLUnitDriver implementation is fastest for HtmlUnit (Headless browser) unlike other divers (geko, ch...). It is non-GUI and does not execute tests on browser but **plain http request**, which is far quick than launching a browser and executing tests.

**@. To upload a file using Selenium**

* If the input is accessible we can use .sendKeys() to send file path to file input field on the element.

**Syn:** driver.findElement(By.id(“uploadEleID”)).sendKeys(“filepath”); //filepath– “c:\\path\\to\\file.txt”

* If the input is non-accessible we have to use tools like Robot class in JAVA, AutoIT, or JavaScript based solutions

**@. To Downloading files using selenium**

Downloading files using Selenium requires some configuration because Selenium doesn't handle downloads directly. However, we can set browser preferences to automate file downloads. By using ChromeOptions or FirefoxProfile.

ChromeOptions ops = new ChromeOptions(); ops.addArguments(“download.default\_directory=D:\\Downloads”);

**@. To switch back from a frame** ------ driver.switchTo().defaultContent();

**@. getWindowhandles() and getwindowhandle()**

getwindowhandles()is used to return the id’s of the all opened windows and its return type is String.

getwindowhandle()is used to return the id of the current window where the control is there and its return type is String

**N:** In order to display the **value of a variable** / **string** **content** in console we have to use the **system.out.println()** command. Here ‘System’ is a class present in java.lang.package. ‘out’ is the static variable of type PrintStream class present in ‘System’ class. ’println()’ is a method present in PrintStream class.

**@. To use ‘recovery scenario’ with Selenium**

Recovery scenarios depends upon the programming language which we use.  If we are using Java then we can use **Exception Handling** technique to overcome same by using “**try-catch Block**” within our Selenium WebDriver tests.

**@. Mention in what ways you can customize TestNG report**

There are two ways to customize the TestNG report those are: Using **ITestListener** **Interface** &&&& **IReporter** **Interface**

**@. To insert a break point in Eclipse**

While working with Eclipse IDE to insert and remove Break points we have to follow the same procedures as below:

* By right click on the line where we want to insert break point and click on “Toggle break point” **Or** Focus on particular command and press “ctrl+shift+B” on the keyboard to select the command in Selenium.
* Multiple break points can be set in IDE.

**@. To debug tests in Eclipse / Steps to dbug**

* First insert the break point at required locations from where we want to execute the script step by step.
* Execute the program by perform context click and select ‘Debug As’ **-->** ‘java application’
* Give confirmation on ‘Confirm Perspective Switch’ window by click on ‘switch’
* At given break point the execution will be paused.
* To continue with the next statement, click on the ‘**Step Over (F6)**’ button for moving each step.
* Step over (f6) – Step by step Execution
* Step into (f5) – Navigate to inside function
* Step return (f7) – Return from inner function
* Resume (f8) – Navigate from breakpoint to breakpoint
* While debugging the script step by step we have to identify the variable values on debug console.
* Click on the “Run” button to continue the execution of remaining program at a time.

**Use:** To find errors in program code and step by step verification.

**Ways to Debug:** Eclipse Debugger, Dynamic debugging technique and online debugging tool.

**@. APIs available in selenium to support Browsers / Drivers** ----- HUD-HtmlUnitDriver HU-HtmlUnit

CD for Ch, FFD for FF, SafariD for Safari, HUD for HU headless Browser, Android - Selelindroid/AppiumD, ios – XCUITestD/AppiumD (ios, Android- mobile OS platforms, Appium-automation tool)

**Headless approach:** Headless browserperforms same functionality as a standard browser but it Runs UI tests without opening browser GUI. Headless drivers are used for faster test execution compare to normal UI Browser test execution. Ex: Headless Chrome and PhantomJS.

**@. To handle network latency in selenium** ----- driver.manage().timeouts().pageLoadTimeout(Duration.ofSeconds(10));

**@. To retrieve the message in an alert box** -----driver.switchTo().alert().getText()

**@. To generate pdf reports what Java API is required** ----- To generate pdf reports in java, we need **iText** jar files

**@. To verify the specific position of a WebElement** ----- EX55

**@. To find XPath for any WebElement which contains no attribute** -----By using XPath functions text() and contains()

**N: IntelliJ** is an IDE that helps to write better and faster code it can be used as option like NetBeans and Eclipse.

**@. To confirms an identified object using selenium** Ref: Java\_WD\_Ex\src\WD\_Examples\EX66\_isElementPresent.java

In WD **isElementPresent(ele)** command is not a built-in method. This user defined method is used to check whether the specified element is present or visible on webpage. it takes element locator as argument. If found that element, it will return a Boolean value.

**@. Break points and Start points in selenium**

**Break Points: It** is for debugging purpose. When we insert a Break Point in our code, the execution will stop here. This helps to verify that, is our code is working as expected or not.

**Start Points:** Start Point indicates that the point from where the execution should begin. Start Point can be used when we want to run the script from the middle of the code or a breakpoint.

**@. To choose Python over Java in Selenium ----** russ

* Java programs tends to run slower compared to Python programs.
* Java uses traditional braces to start and ends blocks, while Python uses indentations.
* **J**ava employs **s**tatic type checking while Python is dynamically typed.
* Python is simpler and more compact compared to Java.

**N:** Java uses **static type** **checking** to analyse the program during compile time to prove the absence of **typing errors**. The basic idea behind this is to never let bad things happen at runtime.

**N: Indentation** in **Python** refers the **spaces** and **tabs** are used at the beginning of a statement while writing the program

**@.** [**TestNG**](https://www.guru99.com/listeners-selenium-webdriver.html) **Listeners**

* **IAnnotationTransformer** is used to transform/modify the content of all the annotations at runtime. (used in retry logic).Ref: TestNG\ReRun\_Failed\_TCs\_usingIAnnotationTransformer
* **IConfigurationListener** is invoked before and after the test configuration methods. It triggers only when the configuration methods pass, fail, or skip.
* **IExecutionListener** is invoked at the beginning and ending of a test run.
* **IHookableListner** is used to execute any code before and after a test method, test configuration method, or suite.
* **IInvokedMethodListener** is used to performs some tasks before and after the method invocation (Overridden methods are beforeInvocation(), afterInvocation()). Ref: TestNG\Listener\_Use\_IExecutionListener\_to\_Mark\_Executed\_TCs\E1\_SkipTestByNameUsingListener
* **IMethodInterceptor** is used to add an interceptor to test method invocation.
* **ISuiteListener** is invoked before and after a suite is run.
* **ITestListener** is invoked before and after a test is run and on test success or failure.

**@.** [**WebDriver**](https://www.guru99.com/listeners-selenium-webdriver.html) **Listeners**

* **WebDriverEventListner** is an interface that contains methods for listening WebDriver events.
* **AbstractWebDriverEventListener** is an abstract class and it implement the **WebDriverEventListener** interface.
* **EventFiringWebDriverListner** class wraps up a WebDriver instance and enables the firing of WebDriver events
* **WebDriverListnerManager** is a class that manages the registered listeners and notifies them of the events.

**@. Use cases of WebDriver Listeners and TestNG Listeners**

* Logging test results
* Taking screenshots of test failure
* Analysing browser logs
* Capturing performance metrics

**@. For Database Testing in Selenium WebDriver what API is required**

For [Database Testing](https://www.guru99.com/data-testing.html) in Selenium WD, we need JDBC (Java Database Connectivity (JDBC enables java programs to connect and interact with databases)) API. It allows to execute [SQL](https://www.guru99.com/sql.html)statements.

**@. Mention when to use AutoIT**

Selenium is designed to automate web-based applications on different browsers but to handle window based and non-HTML popups in the application we need to use AutoIT.

**@. Do we need Session Handling while working with Selenium** Ref: Java\_WD\_Ex\src\WD\_Examples\EX67\_SessionHandling

Session handling in Selenium WebDriver involves managing the connection between the WebDriver and the browser instance, ensuring that commands are sent to the correct browser session, and maintaining that session across different parts of our code. **Adv:** Reusing Browser State and Data (Ref prog), Improving Test Performance, etc.

**@. Advantages of Using GitHub for Selenium**

* When mul people have to work on the same proj, they can update the proj details to other mem’s in team simul.
* Jenkins can help to build the project from remote repository regularly and this helps to keep track of failed builds.

**@. XPath / Main features of XPath / XPath syntax** **/ XPath expressions ---** ps

XPath stands for **X**ml **Path**. XPath uses the path expressions to traverse the elements or attributes in an XML or HTML document. XPath expressions are care sensitive. XPath specifies seven types of nodes those are: Root node, Element node, Text node, Attribute node, Comment node, Processing-Instruction node, and Namespace node. When id, name, className is not present we go for XPath.

**@. Types of location paths in XPath? Use of this / Types of XPath expressions and their usage**

There are two types of location paths are used to specify the location of node in XML documents. Those are:

**Absolute path** is a way of locating an element using XML expression and it starts from the root node **or** document node. It begins with the single forward slash (/). The main dis-advantage of absolute XPath is even a slight change in the UI or any element the hole absolute XPath will get fail (because application html structure might change) and also it is complex to read.

**Relative XPath** is a way of locating an element using XML expression and it starts from any middle node that we've selected. It begins with the double forward slash (//). In Relative XPath there are different ways to creating robust XPaths (it is unaffected by changes in other UI elements).

**Com’n XPath Patterns:** Using single Attribute, Mul Attributes, contains(), starts-with(), text(), or, and, index, XPath Axis.

**@. Functions available in XPath:**

text(): Used to identify elements by their visible text.

contains(): Used to identify elements where an element attribute value is dynamic or partially known.

starts-with(): Used to identify elements where an attribute value starts-with a specific string.

**@. XPath Axes** – Ref: WD\_Examples\EX57\_XPathAxes <https://www.way2tutorial.com/xml/xpath-axes.php> &&& https://www.softwaretestinghelp.com/xpath-axes-tutorial/

XPath Axes are used to identify elements by their relationship like parent, child, sibling, etc. Those are:

**‘Self’ -** is used to get the current node (context node)

**‘child’ -** is used to get the child of current node.

**‘descendant’ -** is used to select all descendant nodes (i.e., child, grandchild’s (c-o-c’s), etc) of **current node within the specified element region only. ---** //\*[@AN=’AV’]//descendant::a Ref: WD\_Examples\EX34A\_Print\_AutoSuggetionsList

**‘descendant-or-self’ -** is used to get current node (self), child, grandchild’s (c-o-c’s), etc. **Within the specified element region only.**

**‘parent’ -** is used to get the parent of current node.

**‘ancestor’ -** is used to select all ancestor nodes (i.e., parent, grandparent’s (p-o-p’s), etc) of current node till root node.

**‘ancestor-or-self’ -** is used to get the current node (self), parent, grandparent’s (p-o-p’s), etc till root node.

**‘preceding’ (**Before**) -** //\*[@A\_N=’A\_V’]//preceding::tag – (here sibling is not mentioned so it will navigate through tags and Ignore its parent tag). It takes all the elements that come **before** the context node to till root node and Ignore its parent and grandparent’s tag.

**‘following’ (**After**) -** //\*[@A\_N=’A\_V’]//following::tag[1] – (here sibling is not mentioned so it will navigate through tags only). It takes all the elements that come **after** the context node till end node (It may be child or grandchild node).

**‘preceding-sibling’ -** Here sibling is mentioned so it will get **before** the context node sibling.

**‘following-sibling’ -** Here sibling is mentioned so it will get **after** the context node sibling.

**@. Categories of XPath operators**

XPath operators are used to perform some special operations in XML document.

* Athematic Operator (+(//input[@id='price' + 1]), \_, \*(//div[@count \* 2]), div(//div[@price div 2]), mod(//input[@id mod 2 = 0]) )
* Comparison Operators (=(//input[@id='username']), !=(//input[@id!='username']), <(//input[@price<50]), <=(//input[@price<=50]), >(//input[@price>50]), >=(//input[@price>=50]) )
* Filter Operators ([](//div[@class='container']), @(//input[@type='text']), `(Union operator, combines node sets) )
* Logical Operators (and(//input[@type='text' and @name='username']), or(//input[@type='text' or @name='password']), not()(//input[not(@disabled)]))
* Node Operators (//(//input (all input elements in the document)), /(/html/body/div), .(.//span), ..(//span/..), \*(//\*[@id='main']))
* Position Operators ([n] (//ul/li[1] (first list item)), last()(//ul/li[last()] (last list item)), position()(//ul/li[position()<=3]))
* String Operators (contains(), starts-with(), text(), normalize-space())

**@. List of standard functions in XPath / XPath Functions** https://medium.com/@nambiarjishnu1210/xpath-functions-a078e2fccbb6

text(), contains(), starts-with(), normalize-space() (//button[normalize-space(text())='Submit']), not() (//input[not(@disabled)]), position() ((//div[@class='example'])[position()=1]), last() ((//li[@class='item'])[last()]), substring() (//div[substring(@id, 1, 4)='test']), substring-before() ( //div[substring-before(@id, '-')='menu']), substring-after()(//div[substring-after(@id, '-')='item']), and/or, string-length()(//div[string-length(text())>10]), name() (//div[name()='span']), local-name() (//\*[local-name()='svg']), round() (Rounds a number to the nearest integer), floor() (Rounds down to the nearest integer), ceiling() (Rounds up to the nearest integer).

**N:** normalize-space() function in XPath is used to trim leading and trailing spaces and replacing multiple with one.

**Ex:** //label[normalize-space(“Search Amazon.in”)]

**@. XPath Comparison operators** are used to compare one value to another value. Those are: =, !=, <, >, <=, >=

**@. XPath Boolean operators** are used to combine or exclude keywords. Those are: AND, OR, NOT etc.

**@. XPath Number operators** are used to perform mathematical operations on diff. kw’s. Those are: +, -, \*, div, mod(%)

**@. XPath String functions** are used to deal with string content according to our preferences.

**@. XPath nodes / To define relationship among nodes**

There are different types of nodes in XML document. All these nodes are related to each other in the form of **tree structure** based on this structure we can define the relationship among the nodes. These nodes are: Parent, Child, Sibling, etc. The top most node of the tree is called **Root Node**.

**@. To check a checkbox in selenium**

click() is used for clicking buttons or radio buttons. isSelected() is used to check whether CB or RB is checked or not.

**N: Instead of click()** we can use Keyboard events i.e. **sendKeys(Keys.ENTER)** /**submit()** methods to perform click activity.

**N:** To copy text from user name edit box and paste in password field with the help of **KB Events**.

**@. get() and navigate().to() method**

Both commands are used to navigate to a URL passed as parameter. When we use **get()** method to navigate particular URL, the WebDriver will wait until the page is loaded (i.e. after refreshing the page) and It doesn’t maintains browser history or cookies to navigate back or forward in browser hist. When we use **navigate().to()** method to navigate particular URL, It doesn’t wait for the full page loading, the WebDriver will go to next step in the script to execute the script irrespective of page is loaded or not (i.e. without refreshing the page) so it may have chance to get exception. It maintains browser history or cookies to navigate back or forward in browser history by using driver.navigate().forward() & driver.navigate().back() methods.

**@. Atomic value in XPath ------** Nodes that has no parent node or child node are called as **Atomic values** in XPath.

**@. pom.xml**

**pom**.**xml** is an XML file which contains project configuration details (such as dependencies, build settings/configurations like (build directory, source directory, test source directory, plugin, goals, etc.)), plugin configurations, and project metadata used by Maven to build the project. In normal project development, we have to add libraries manually as required. In **pom** context we call those libraries as **dependency’s**. In **Maven based** development, require **dependency’s** can added to the project using **pom.xml inside the dependency tag** and then save the file then libraries will automatically download and add to our project. Just by changing the version number in dependency the libraries will automatically update. Maven reads the pom.xml file and executes the goal.

It is an XML file that contains project configuration information such as project dependencies, build settings, The POM file serves as the centrepiece of a Maven project.

So, the pom.xml file in a Maven project is used to manage project dependencies and build settings, such as specifying external libraries (dependencies) needed by your application, configuring plugins, defining project properties, and specifying the version of Java to use for compilation.

**@. Software Metrics can used in project ----** pqr

A **Software** **Metrics** are used to measuring the software performance, productivity, planning work items, quality, risk etc. It can be classified into 3 types those are:

* **Product metrics** describe the product char’s such as size, complexity and design features.
* **Process metrics** can be used to improve software development and testing process such as effectiveness of defect removal during development and pattern of testing when defect is arrival.
* **Project metrics** describe the project characteristics such as staffing pattern over software life cycle (number of software developers, Testers are required, etc), cost, schedule and productivity.

**@. To handle Dynamic Elements since the webpage is designed in AngularJS**

Different ways to handle dynamic elements those are by constructing a **Generic XPath** (dynamic XPath), waiting strategies also helpful, XPath with indexes, custom locators, or JavaScript Executor. In few scenarios, the element attribute values change dynamically. **For Ex** let us consider the ‘id’ of a username field is ‘username\_123’ and the XPath will be *//\*[@id='username\_123′]* but when we open the page again the ‘id’ of ‘username’ field might have changed and the attribute value may be changed to ‘username\_234’. In this case, the test will fail because the WebDriver could not find the XPath we have passed earlier as the ‘id’ of the field has changed to some other value. There are many approaches that depending upon the type of problem like If some part of the attribute value changesor If entire value of the attribute changes dynamically:

Strategies:

* Use Relative XPath with stable attributes (contains(), starts-with()).
* Use parent-child relationships.
* Use custom waits instead of fixed locators.

**1)** XPath with contains --- //\*[contains(@id,'username')] (If some part of the attribute value changes it will work)

**2)** XPath with starts-with --- //\*[starts-with(@id,'username')] (If some part of the attribute value changes it will work)

**N:** ‘contains()’ will checks whether the ‘id’ of the attribute ‘value’ contains the specified substring and starts-with() will checks whether the ‘id’ of the attribute ‘value’ must starts with specified substring.

**3)** XPath with OR/AND --- //\*[@A\_N1=’A\_V1’ OR/AND @A\_N2=’A\_V2’]

**4)** We can create generic XPath with the help of **XPath Axis** also.

**@**: If id of ‘login’ field changes dynamically and there is no constant value to use **contains()** & **starts-with()** methods.

**Sol:** Selenium provides different methods to use **sendKeys()** with **fun keys** like TAB key, ENTER key and KB Events……

**4)** Waiting strategies: Using WebDriverWait with appropriate ExpectedConditions like elementToBeClickable or presenceOfElementLocated to ensure the element is in the expected state before performing actions **or** to wait for elements to appear.

**@. To handle dynamic IDs in selenium** ---- use XPath with contains() or CSS with partial attr match [attribute=’value’]

**@. XPath for link which contains text ‘Im in a meeting’** --- //tagName[contains(text(),’ im in a meeting’)]

**@. Which scenario can’t be automated**

Dynamic content: Highly dynamic or interactive content that changes frequently.

Visual or subjective validation: Some scenarios requiring human judgement or visual inspection.

Complex workflow: Complex end-to-end workflows requiring real-time human interaction.

**@. Elements in pom.xml (Elements inside Dependency’s tag)**

For creating a simple pom.xml file, we need some elements those are:

|  |  |
| --- | --- |
| **Element** | **Description** |
| **Project** | It is the root element of pom.xml file |
| **modelVersion** | It specifies the version of the Maven POM model used by the project. By default, it should be 4.0.0 |
| **groupId** | It specifies the id for the project group. |
| **artifactId** | It specifies the id for the project artifact (i.e., proj name). An artifact is something either produced or used by a project. Ex’s of Maven produced artifacts are: JARs, source, binary distributions and WARs. |
| **Version** | It specifies the version of the artifact under given group. By default, it should be 0.0.1-SNAPSHOT |
| **Packaging** | defines pack’ng type such as .Jar (java archive file), .war(web archive file), .ear(enterprise archive file) |
| **Name** | defines name of the maven project |
| **url** | defines URL of the maven official website |
| **Description** | define a simple description regarding project |
| **Dependencies** | define dependencies for this project |
| **Dependency** | define dependency for this project and it can keep inside dependencies tag |
| **Scope** | In a Maven project, the <scope> element within a <dependency> tag defines how and when a dependency is used during the build lifecycle. It controls the visibility and availability of the dependency in different phases like 'compile`, `provided`, `runtime`, `test`, and `system`. |

**@. Can use close() and quit() together**

Yes, we can use close() and quit() together in Selenium. These both are used to end a session in Selenium. close() will close the current window **and** quit() will close all open windows and end the session related to WebDriver.

**@. To build the maven project / To add dependency’s in pom.xml**

Maven is java-based build automation management tool that allows to import dependencies into our projects. If we create a maven project by default pom.xml file will create. Using this pom.xml file as a template, we need to add **tags** inside pom.xml file. In order to use Maven, it is necessary to explicitly add required dependencies to **pom.xml** file and save the file then that dependencies will automatically download, add and update.

1. Download **Apache Maven** ([apache-maven-3.8.1-bin.zip](https://mirrors.estointernet.in/apache/maven/maven-3/3.8.1/binaries/apache-maven-3.8.1-bin.zip))from internet and un-zip it.
2. Set Environment variables in local system.

* M2\_HOME --- RC on This PC 🡪 Properties 🡪 Adv Sys Settings 🡪 Env var’s 🡪 click on ‘New’ under ’Sys Vars’ 🡪 set “**variable name**” as “**M2\_HOME**” and “**variable value**” as “Location of folder where the maven is”
* MAVEN\_HOME --- again click on ‘New’ under ’Sys Vars’ 🡪 same as above

**N:** some programs look for try to find M2\_HOME or MAVEN\_HOME that’s why I added both the variables

* Path --- select ‘path’ under Sys Vars 🡪 Edit 🡪 New 🡪 add **.bin** path 🡪 OK

1. To check installation was done or not 🡪 go to CMD and check maven version by using “mvn -version”
2. Install maven plugin (i.e., Maven Integration for Eclipse (Luna/Mars) 1.2.0) in Eclipse.

## Create Maven project: File 🡪 New 🡪 Project 🡪 Maven 🡪 Maven Project 🡪 Next 🡪 Give workspace location 🡪 Next 🡪 Catalog (Internal) and Select an Archetype i.e., ‘maven-archetype-quickstart 1.1’ 🡪 [Next]

## Specify the archetype parameters those are GroupID and ArtifactID (Proj name) by giving these by default version (0.0.1-SNAPSHOT) and ‘package (GroupID.ArtifactID) under src/test/java folder’ will show and click on [Finish]

1. There are two folders will create by default **src/main/java** (maintain **1) Utilities folder** like DataProviders (tdcf), ExtentReportManager, XLUtility (tdcf) and also Test Data files like excel file, properties files...etc) **2) Resources folder** Log4j.xml file (for Log4j configuration) and routes.properties (maintain URL in k,v pair format) --- tdcf - Test data configuration file and **src/test/java** (this folder is used to create automation test scripts
2. Add necessary dependencies by copying it from ‘mvnrepository.com’ and paste inside the dependencies tag and save pom.xml file then only the jars will download and added to project it can see in “Maven Dependencies” folder
3. Prepare the script.
4. Before executing the script, we have to import “maven-surefire-plugin” and “maven-compiler-plugin” from ‘https://maven.apache.org/’. These are helpful to run our tests through pom.xml file and these can keep inside <build> tag. Without ‘maven-surefire-plugin’ we can’t able to execute our script through pom.xml file.
5. testng.xml will specify along with maven-surefire-plugin tag. ‘maven-compiler-plugin’ is for compilation purpose and ‘maven-surefire-plugin’ is responsible to execute our script through pom.xml (i.e. run TestNG TCs when we run the mvn test command) and also generate reports in the target/surefire-reports folder. pom.xml will control our test execution by testng.xml file by specifying testng.xml file inside <build>.

* Browser 🡪 search for ‘Maven Surefire Plugin’ 🡪 click on ‘usage’ under OVERVIEW 🡪 copy <build>..</build> tag only 🡪 paste above the <dependencies> below the ‘</properties>’
* While running through pom.xml our Test Execution is completely depending upon the testng.xml because we specified testng.xml file is kept inside <suiteXmlFiles>

1. To execute the script RC on pom.xml (Run As 🡪 8 Maven test)
2. If we get error regarding “Source option 5 is no longer supported. use 7 or later” or “Target option 5 is no longer supported. use 7 or later” to overcome these errors add two tags inside <properties> those are <maven.compiler.source>7</maven.compiler.source> and <maven.compiler.target>7</maven.compiler.target> and then save pom.xml
3. Execute the script

**Adv:** Maven is used to build the project and to download, add, update libraries to project. Compile and execute the project without using eclipse. We can able to run the script through CMD, batch file, pom.xml file and Jenkins, etc.

**Maven commands:** Toexecute maven proj through cmd / powerShell win must need to install maven on local machine.

**N:** To Run Test Cases from powerShell window

Goto proj directory 🡪 Shift+RC on mouse 🡪 select "open PowerShell window" 🡪 "mvn clean install / test" 🡪 Enter

**==>** **mvn clean** will clean maven artifacts that was created by previous maven builds

## N: To compile the project

Goto proj dir 🡪 open dir (copy path) 🡪 open CMD 🡪 f: 🡪 **E** 🡪 cd paste\_copied\_path 🡪 **E** 🡪 mvn clean compile 🡪 **E**.

**N:** To Run Test Cases from command prompt

Goto proj dir 🡪 open dir (copy path) 🡪 open CMD 🡪 f: 🡪 **E** 🡪 cd paste\_copied\_path 🡪 **E** 🡪 mvn test 🡪 **E**.

**N:** To execute maven goals from the command line using the `mvn` command followed by the desired goal. **Ex:** mvn clean install. This command executes the 'clean' and 'install' goals, which clean the project and install artifacts into the local Maven repository, respectively.

**N:** To Run Test Cases from .bat file

🡪 Open notepad 🡪 f: 🡪 Save As with extension .bat 🡪 double click on .bat file

cd paste\_copied\_path

mvn test

**Repositories in maven:** Maven Repository is a directory or server it holds the metadata files, dependencies and build artifacts (jars, war, plugins…...). There are two types of repositories in maven those are: **Local** and **Remote**:

**Local Repository** will get generate in our local machine (It is typically located **.m2** directory) by executing maven command. Where Maven stores project dependencies and artifacts downloaded from remote repositories.

**Remote Repository** is a custom repo it will create by organisations for those need. It can be accessed over the network

**N: Type of class in utility package:** ExtentReport, ScreenShotUtility, JSONReader, RetryListner and AssertionService.

**N:** To connect Maven and TestNGadd TestNG dependency in pom.xml and use maven-surefire-plugin to run TestNG tests via mvn test command.

**Maven Plugins:** Maven plugins are extensions those provide additional functionality to Maven's build process. These are used to perform various tasks such as compiling code, running tests, generating documentation, and deploying artifacts. Understanding how to configure and use Maven plugins is important for customizing and extending the build process in Maven projects. There are two types of maven plugins are available. Those are:

**Build plugins** are executed during the build. These are declared inside <build> in pom.xml file.

**Reporting plugins** are executed during the site generation. These are declared inside <reporting> in pom.xml file

**Other plugins supported by maven project: Core Plugins:** A list of maven core plugins are --- ccd fir ssv

|  |  |
| --- | --- |
| Clean | Clean up after build (clean the build artifacts that was generated by the previous builds and temporary files from project). |
| Compiler | Compiles java source code. |
| Deploy | Deploy the build artifact into remote repository. |
| Failsafe | Runs the JUnit **integration tests** in an isolated classloader. |
| Install | Install the build artifacts (JAR, WAR, etc.) into the local Maven repository for use by other projects. |
| Resources | Copies the resources to output directory including JAR. |
| Site | Generate a site for the current project |
| Surefire | Run the JUnit **unit tests** in an isolated classloader. It used to run our tests through pom.xml file |
| Verifier | Verifies the existence of certain conditions. It is useful for integration tests. |

1. **Maven Build Phases / Goals / Lifecycle / Lifecycle phases**: Maven organizes the build process into lifecycles and maven build lifecycle goes through a set of stages, they are called build phases. There are 3 main types of lifecycles in maven.
2. **1) Default Lifecycle** is the primary lifecycle used to build, test, & deploy a project. Below is a typical Maven build workflow for the default lifecycle.

|  |  |
| --- | --- |
| **Lifecycle Phases** | **Description** |
| Validate | Check the project configuration and validate if all information required is present (mvn validate) |
| Compile | Compile the source code of the project. (mvn compile) |
| Test | Execute unit tests using a testing framework. (mvn test) |
| Package | Bundle/packages the compiled code into a JAR/WAR file (mvn package) |
| Verify | Perform checks on the packaged code. (mvn verify) |
| Install | Install the package to the local repo for use as a dependency in other proj’s locally. (mvn install) |
| Deploy | Deploy the package to a remote repository for sharing with other developers. (mvn deploy) |

**N:** To do all the steps in single goal: mvn clean package

**N:** Maven can build any number of projects into desired output such as .jar, .war, .ear…

**2) Clean lifecycle:** This lifecycle handles project cleaning, ensuring the build environment is fresh. Its phases include:

|  |  |
| --- | --- |
| **Phase** | **Description** |
| pre-clean | Executes actions needed before cleaning. (mvn pre-clean) |
| clean | Deletes the output of the previous build. (mvn clean) |
| * post-clean | Executes actions needed after cleaning. (mvn post-clean) |

1. **3) Site lifecycle:** This lifecycle is for generating project documentation and reports. Its phases include:

|  |  |
| --- | --- |
| **Phase** | **Description** |
| pre-site | Executes actions before generating the site. (mvn pre-site) |
| site | Generates the project site documentation. (mvn site) |
| * post-site | Executes actions after site generation. (mvn post-site) |
| * Site-deploy | Deploy the generated site to a webserver or repository (mvn site-deploy) |

**@. Maven Archetype**

A Maven archetype is a project template or blueprint used to generate new Maven projects with predefined directory structures, configurations, and build settings. Archetypes provide a starting point for creating new projects and help maintain consistency across different projects within an organization. Maven provides several built-in archetypes for common project types, such as web applications, Java libraries, and more.

**@. To exclude a specific dependency in pom.xml**

To exclude a specific transitive dependency in our pom.xml (used in Maven projects), we can use the <exclusions> tag inside the <dependency> block where the unwanted dependency is being pulled in.

- We can exclude multiple dependencies by adding more <exclusion> blocks.

- This only works for transitive dependencies — those pulled in by other dependencies.

- If we want to exclude a direct dependency, simply remove its <dependency> block.

**@. Test Cases can be automated per day**

The efficiency of our tests is always Depends upon the **Application** and **Complexity of Test Cases**. It includes 1. Analyzing Test Cases 2. Developing the script 3. Executing and Debug the script 4. Stabilizing the script.

**@. To estimate the time to automate a scenario**

Various factors should consider to estimate the time for Selenium specific project those are:

1. **Scope of the project**
2. **Complexity of application**
3. **Use of supporting tools/technologies**
4. **Environment setup**
5. **Implementing the Framework**
6. **Scripting and Reviewing**
7. Capability and experience of the testing team (i.e., Experience with testing tool, framework and environment).

**@. If the element is not present, how will print the Element is not present message**

WebElement element = driver.findElement(By.LN(LV));

Assert.assertFalse(element.isDisplayed()); **(or)** Assert.assertNull(element);

**isElementPresent()** is used to check whether the element is present on DOM. It returns Boolean value. It is not a built-in method.

**isDisplayed()** is used to check whether the element is visible or not on the web page.

**@. To check the Actual vs Expected Result**

Comparison Testing is helpful to determine whether our Test Cases result is pass/fail. After performing the test, tester will get Actual Result (Actual Outcome). It is always documented along with the Test Case during the test execution phase. This Actual Result is compared with the Expected Result and the deviations are noted. If any deviation then we log as **defect** and that defect goes through the defect life cycle. After getting the Actual Result only we can mark whether the scenario is pass or fail. Comparison test tools enables to mark the date and time stamp in result.

**Comparing Automated Testing Tools:**Selenium WebDriver, Katalon Studio and Unified Functional Testing (UFT).

**@. URL and a URI**

**URI** stands for Uniform Resource Identifier and **URL** stands for Uniform Resource Locator. **URI** is name and address (Locator) for online resource whereas **URL** is just address (**L**ocator) for online resource. **URLs** is a subset of **URIs**.

**@. Defect lifecycle**

Defect lifecycle is the journey a defect goes through from the moment it is found until it is fixed including status like New -> Assigned -> Open -> Fixed -> Retest -> verified -> Closed. The cycle begins when a fault is discovered and concludes when the defect is closed after it has been verified that it will not be recreated. If not fixed Reopen -> Deferred -> Rejected -> Duplicate.

**@. 'Defect Triage' / Need of 'Defect Triage'**

Defect triage is a process here each bug is prioritize based on its severity, risk, amount of time it will take to fix the fault, etc. It is used to define the severity and priority of defects, make changes as per need and assign resources. Mainly it is useful in Agile project management.

**@. Mandatory and other participants of 'Defect Triage'**

**Mandatory Participants:** Project Manager, Technical Lead/Development Lead, Test Lead

**Optional Participants:** Developers, Testers, Business Analyst

**@. Roles and Responsibilities of participants during 'Defect Triage' / What happens during DTM / outcome of DTM**

* During the defect triage meeting, each defect is analysed to see whether right **Priority** & **Severity** was assigned to it.
* If needed re-assign the priorities and discuss the reason with each attendee so that other members can understand the **Root Cause** of defect.
* Determine the order in which defect should be fixed.
* Updates will capture in bug tracking system.

**@. TestNG Annotation**

TestNG Annotations are used to control the flow of execution (like sequence and priority) of test methods. These are easily inserted inside the program. Which allows to execute our code before and after certain points.

**Hierarchy of the TestNG Annotations:**

* @BeforeSuite: under this annotation method will execute before the execution of <suite> (this is at suite level)
* @BeforeTest: under this annotation method will execute before the execution of <test> (this is at test level)
* @BeforeClass: under this annotation method will execute before execution of class
* @BeforeMethod: under this annotation method will execute before the execution of each and every @Test ann
* @Test: @Test annotation indicates that this particular method is in test environment
* @AfterMethod: under this annotation method will execute after the execution of each and every @Test ann
* @AfterClass: under this annotation method will execute after the execution of class
* @AfterTest: under this annotation method will execute after the execution of <test> in xml
* @AfterSuite: under this annotation method will execute after the execution of <suite> in xml

**@. Annotation with real time use**

@BeforeSuite //configure Extent report tool to customize reports

@AfterSuite //close connection of extent report and get the execution report backup

@BeforeClass //gobal configuration like launch browser

@AfterClass //close browser

@BeforeMethod//login

@AfterMethod //logout

@Test //actual test scripts

**@. H many Sprints have worked and how often will get Sprints**

My current organisation sprint structure is around 3 weeks of development and 4th week allows us to show the progress to the client, deploy it for testers, eliminate bugs and discuss about the next sprint.

**@. POM design pattern / Adv’s / important parts in POM / Lazy Initialization Technique / Diff bn POM and PgeFactory**

POM is a design pattern used to separate test logic and to create an object repository for storing WebElements while implementing the framework. Here element identification code (WebElements identification methods belongs to the page) and validation codes are kept separated. For each page of the application a class is created. Here each WebElement returns the identification value. Test scripts are maintained in separate class and page object methods are call from the test scripts file to perform operations. In selenium-java POM is implemented, with the help of **@FindBy** and **PageFactory** annotations. There are 3 steps in POM those are Identification (Identifying the elements with @FindBy annotation), Initialization (initialize the elements inside the constructor by using PageFactory annotation), Declaration (creating action methods to WebElements and utilize these methods). If element is not found in POM we will get **StaleElementRefferenceExcp** and if PageFactory is not used to initialize WebElements we will get **NullPointerExcp**.

**Adv:**

* We can create an Object Repository using POM design pattern.
* For any change in UI (r WebElements) only page object files are need to be updated leaving test files are unchanged.
* It makes code readable, reusable and maintainable (by separating test scripts from page objects).
* Avoid code duplication.

**Dis-adv:**

* Initial Setup Overhead: Setting up page classes, methods, and structure takes time, especially for small projects.
* Extra Maintenance: For dynamic appln’s with frequently changing UI, we may need to update many Page Obj classes
* Over-Engineering Risk: For very simple test suites, POM may introduce unnecessary complexity.
* Duplicate Code Without Proper Design: If not structured properly (e.g., using BasePage or inheritance), it may lead to duplicate code in page classes.

**@. Test Automation Framework / Adv’s / Folder structure of a real world frameork**

A FW is a structure with set of guidelines and rules must need to follow while creating and designing the Test Scripts.

**Adv: -- crem**

* Provide consistency in testing
* Readability and Reusability of script
* Improve efficiency and effectiveness
* Maintainability (Less maintenance)

**Folder structure includes:** TCs, pages, utils, configs, resources, and logs/reports folders.

@. Can u walk through the current automation framework design

My current framework is modular hybrid framework - combining UI automation using selenium + TestNG, and API automation using REST Assured. It follows POM design pattern for maintainability. The framework is fully integrated into the CI/CD pipeline (Jenkins), supports parallel execution via Thread Local WebDriver, and generate detailed Extend reports. We also follow SOLID principles to ensure the framework is scalable and easy to extend.

**@. Key components include while designing a test automation framework from scratch**

1. Define Objectives & Scope

2. Choose the Right Tools & Tech Stack

3. Framework Architecture / Example Folder Structure (includes Test Runner, POM, Test Data Management, Logging & Reporting, Configuration Management, Parallel & Cross-Browser Testing)

**@. Cookies / delete cookies in selenium ----** Ref: WD\_Examples\EX01e\_CookieValidation

Cookies are the small text files that are stored by a website in our computer to keep track of information about our browsing on that site.

* With the help of **driver.get(“**url**”)** command to launch URL and to delete cookies.
* Click on Logout link is mandatory to our application sometimes this will delete session cookies.
* By using **driver.manage().deleteCookieNamed(**arg0**)** / **.deleteCookie(spec\_cookie)** / **.deleteAllCookies()** methods

**@. To get cookies in selenium**

* With the help of **driver.navigate().to(“**url**”)** command to launch URL and to get cookies.
* Read the cookie information by using **driver.manage().getCookieNamed(**arg0**)** / **getCookies()** methods
* Store the cookie information (data) by using **FileWriter** Class (to write streams of characters) and **BufferedWriter** class (to write the text into a file and to create as .txt file).

**Return Type:** Methods can return a value but it can possible without **void** in method signature. If the **void** is mentioned in method signature, that means method doesn’t return any value. If method is returning a value, then must specify the type of the value it is returning. return keyword is used to stop a method and send a result (if needed) back to the place where method was called. Ref: Java\_Examples2\Return\_Keyword

**@. To capture browser logs / To monitor console errors** Ref: WD\_Examples\EX01g\_CaptureBrowserLogs

Use LogEntry from driver.manage().logs().get(“browser”);

**@. Various commands / functions provided by Selenium WebDriver**

Selenium WebDriver commands can broadly classify into 3 categories those are:

1. Browser interaction commands
2. [Navigation Commands](https://www.javatpoint.com/selenium-webdriver-navigation-commands)
3. [WebElement Commands](https://www.javatpoint.com/selenium-webdriver-webelement-commands)

**1. Fetching a webpage:** **get()** and **navigate().to()** methods are used to fetch particular web page.

**2. Locating forms** and **sending user inputs:** driver.findElement(By.LN("LV")) method is used to locate forms and sendKeys("javatpoint tutorials") method for sending user inputs.

**3. Clearing User inputs:** clear() method is used to clear the user inputs from text box or edit box.

**4. Fetching data over any WE: getText()** is used to fetch the visible text over a WebElement to performing some Assertion operations and Its return type is string. Without using getText() it is not possible to get the text from webpage.

**6. getAttribute()** is used to retrieve the value of a specified attributes (like href, id, name, class..) of a web element and Its return type is string.

**N:** **getText()** is used to get visible inner text of a WebElement and **getAttribute(“AttributeName”)** is used to get the attribute value of the given WebElement where as getAttribute(“A\_N”) also used to get the input what we have passed to a WebElement. Both this method return type is string. And these methods are present with WebElement interface.

**7. Performing click() & submit() events:** click() method is used to perform click activity on any web element whereas submit() method is used to perform enter activity.

**8. Navigating backward & forward in browser history:** driver.navigate().back(); **&&&** driver.navigate().forward();

**10. Refresh / Reload a web page:** driver.navigate().refresh();

Dis-adv: The moment we refresh the web page sometimes it throws the **StaleElementReferenceExcep** because the DOM will get loading again or when the referenced element is no longer valid or present in the DOM. It can be handled by using WebDriverWait or wrap in a Re-finding mechanism (by try-catch block, for, or while loop) (bcz of this mechanism we can relocate the element) (Ref: WD\_Examples\EX69\_StaleElementReferenceException\_Handling) or Refresh the page.

**11. Closing Browser:** We can use two methods for this

* driver**.close()**; ---- (To close current focusing window in browser associated with WebDriver)
* driver**.quit();** ---- (To close all other opened windows in browser associated with WebDriver and close WD section)

**12. Moving between Windows:** driver.switchTo().window("windowName");

**13. Moving between Frames:** driver.switchTo().frame("frameName");

**14. Drag and Drop:** Drag and Drop operation is performed using the Action class.

WebElement elementSource = driver.findElement(By.name("source"));

WebElement elementTarget = driver.findElement(By.name("target"));

Actions act = new Actions(driver));

Act.dragAndDrop(elementSource, elementTarget ).build().perform();

(or)

To perform drag-and-drop call **clickAndHold()** on the source element, then **moveToElement()** to the target element, and finally release the mouse using **release()**. This simulates the drag-and-drop action effectively.

Some other commands like: getPageSource(), getCurrentUrl(), getLocation(), getFistSelectedOption() and getOptions()

**@. Can we test APIs or web services using selenium WebDriver**

No, selenium WebDriver always interact with browser by using the **browser’s native support** to automate the web applications so API testing is not possible using selenium WebDriver because APIs do not run-on browser. For API and Web services testing we can use Postman, SoapUI, or RestAssured.

**@. To inspect the WebElement attributes in order to use them in different locators**

To inspect web element attributes, we can use various browser developer tools like Chrome DevTools or Firefox Developer Tools. Chropath is a plugin that provides XPaths and CSS Selector. From automation perspective:

- RC on the element we want to inspect and select "Inspect" from the context menu. This will open the developer tools.

- In the developer tools, ensure we are in the "Elements" or "Inspector" tab.

- Locate the HTML code corresponding to the element we want to inspect. It will be highlighted in the developer tools.

- In the HTML code, we can find different attributes of the element such as "id", "class", "name", etc.

- We can use these attributes in diff locators like CSS selectors, XPath, or others to locate the element in test automation

**N:** Capture the Element Attributes Using Selenium (by getAttribute()). Use JavaScript for Dynamic Attributes.

**@. To move the parent of an element using XPath https://www.quorcom/How-can-we-move-to-parent-of-an-element-using-XPath**

**//div[@id=”childid”]/..** or **//div[@id=”childid”]/parent::parent\_tagName** cmd will locate the **Pr** ele with **ch** attribute.

**@. To locate an element by only partially matching the value of its attribute in XPath**

We can use **contains()** method to locate an element while partially matching its attributes value. This is particularly helpful in scenarios where the attributes have dynamic values with certain constant parts. **Ex:** //\*[contains(text(),'text')] Whereas in Links, we can use locator as **partialLinkText(“LV”)** method.

**@. To move nth-child element using XPath** -----https://www.tutorialspoint.com/how-to-identify-the-nth-sub-element-using-xpath

There are two ways to navigating the nth element using XPath

* Using **square brackets** with **index** EX: div[3] 🡪 It will find the third div element 🡪 index will start from ‘zero’
* Using **position()** mtd EX: div[position()=3] / div[‘position()=3’] / div[‘position=3’] 🡪 It will find the 3rd div element

**@. To move nth-child element using CSS Selector** -----https://www.swtestacademy.com/css-selenium/

**tag:nth-child(position\_num)** will locate the nth-child element EX: div:nth-child(2) it will locate 2nd child element (i.e. div).

|  |  |  |
| --- | --- | --- |
| **Description** | **XPath** | **CSS Path** |
| Direct Child | //div/a | div **>** a |
| Child or **child-of-child** | //div**//**a Ref: EX52c\_DDDDDDynamic\_Capt\_DynValues | diva |
| Id | //div[@id=’idValue’] | div**#**idValue |
| name | //div[@name=’nameValue’] | Syn: [Attr=’val’] Ex: [name=’nameVal’] |
| Class | //div[@class=’classValue’] | div**.**classValue (remove the spaces) |
| tagName | tagName(“//tagName”) | cssSelector(“tagName”) |
| following-sibling | //ul/li[@class=’first’]/following-sibling::div[1] | ul>li.first **+** div (ul>li.first – pr, div – ch) |
| preceding-sibling | //ul/li[@class=’first’]/preceding-sibling::div[1] | is not possible with CSS bcz with CSS we can traverse only in forward dir: |
| Attribute | //form/input[@name=’username’] | form>orspaceinput[name=’username’] |
| Multiple Attributes | //input[@name=email and @type=’button’] | input[name=’continue’][type=’button’] |
| OR operator | //input[@name=email or @type=’button’] | input[name=’continue’]**,**[type=’button’] |
| nth Child | //ul[@id=’list’]/li[4] | ul#list li:nth-child(4(position\_num)) |
| First Child | //ul[@id=’list’]/li[1] | ul#list li:first-child (ul#list - pr, li - ch) |
| Last Child | //ul[@id=’list’]/li[last()] | ul#list li:last-child (ul#list - pr, li - ch) |
| Attribute Contains | //div[contains(@title,’Title’)] | div[title\*=”Title”] |
| Attribute Starts-With | //input[starts-with(@name,’user’)] | input[name^=”user”] |
| Attribute Ends-With | //input[ends-with(@name,’user’)] | input[name$=”user”] |
| Element with Attribute | //div[@title] | div[title] |

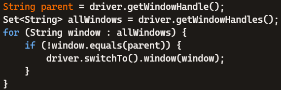
**@. To locate elements using their text in XPath** ----- //\*[text()=’text content’]

**N:** CSS syntax in general / select ele’s by their attribute value using the CSS Selector - “tag[A\_N=’A\_V’]” **Ex:** li[class=’A\_V’]. we can also use other comparison operators like **\*=** (contains), **$=** (starts-with), and **$=** (ends-with) to match attribute values with specific patterns.

**N:** To find matching elements through DOM console**:** $x(“XPath”) | $$(“cssSelector”). To clear console: ctrl+L

**@. Purpose of window handling / Switch b/n mul browser Windows/Tabs in selenium / Diff arg’s pass in window()**

Whenever we have multiple browser windows/tabs to switch the program control between these we can go with window handling concept in selenium. Different arguments pass in windows() or we can perform switch control between windows in 3 ways: using window title, using window url, using window id.

**Set<String>**, **Iterator<String>**, **driver.getWindowHandles()** and **driver.switchTo().window(“**whn**”)** commands allow to work with multiple windows. Here **getWindowHandles()** command returns the list of id’s corresponding to each window and by passing a particular window handle to **driver.switchTo().window(“**whn**”)** will switch to specific window.

**@. Is there a way to refresh the browser using selenium**

* Using **driver.navigate().refresh()** command to refresh the current page
* Using **driver.get(“URL”)** to navigate into URL
* Using **sendKeys(Keys.F5)** method to refresh the textbox on the webpage

**@. To find the value of different Attributes (Attribute\_Values) for name, class, id of an element**

**Ex:** String attributeValue = driver.findElement(By.LN(“LV”).getAttribute(“name/id”); // it will return A\_V of ‘name’

**@. Some expected conditions that can be used in Explicit Waits**

.elementToBeClickable(WebElement element)

.stalenessOf(WebElement element)

.visibilityOf(WebElement element)

.visibilityOfElementLocated(By.LN(LV))

.invisibilityOfTheElementLocated(By.LN(LV))

.presenceOfElementLocated(By.LN(LV))

.attributeContains(WebElement element, String attribute, String value)

.titleContains(String title)

.titles(String title)

.alertIsPresent()

.textToBePresentInElementLocated(By String)

**@. Keyboard Operators / Operations**

actns.keyDown(Keys.***CONTROL***).sendKeys("a").keyUp(Keys.***CONTROL***).build().perform();

sendKeys(“sequence of character”) – Used for passing character sequence to an input or textbox element

keyPress(KeyEvent.***VK\_CONTROL***) – Used for press keys like control keys, function keys, etc. those are non-text

keyRelease(KeyEvent.***VK\_CONTROL***) – Used in conjunction with **keyPress** event to simulate releasing a key from KB event.

**@. Mouse / Pen Actions**

Click(ele)

clickAndHold()

doubleClick(ele)

ContextClick(ele)

mouseUp(ele)

mouseDown(ele)

mouseMove(ele)

moveToElement(ele): method is used to move the cursor on the particular element.

moveByOffSet(xOffset, yOffset)

setActivePointer(PointerInput.Kind.PEN, "default pen")

**@. To fetch the current URL in selenium** ---- driver.getCurrentUrl()

**@. To fetch the title in selenium** ---- driver.getTitle()

**@. To fetch the page source in selenium** - driver.getPageSource(); This method returns a string containing page source.

**@. To check if an element is enabled for interaction on a web page** – isEnabled()

**@. To locate a link using its text in selenium / To click on Hyper Links using selenium**

**linkText()** and **partialLinkText()** methods are used to locate links. Difference between this two is **linkText()** matches complete string passed as a parameter to **linkText**. Whereas **partialLinkText()** matches some portion of string passed as a parameter to **partialLinkText**.

**@. To check which option in the dropdown is selected**

Select countriesDD = new Select(driver.findElement(By.id(“countries”))); ---- arg – dd parent element

countriesDD.selctByVisibleText(“India”); ------- // to select DD option ‘india’

boolean elePresent = driver.findElement(By.xpath("xpath")).isSelected(); **(or)**

System.out.println(driver.findElemnt(By.id(“India”)).isSelected()); -------- // It returns the Boolean value in console

**@. To handle window UI elements and window Popups using selenium**

Selenium is used for automating web-based applications only. For handling window GUI elements, we can use AutoIT or Sikuli. AutoIT is a freeware used for automating window UI. AutoIT scripts follow simple BASIC language (syntax) and these can easily integrate with selenium tests.

**@. Robot class API** Ref: WD\_Examples\Ex27b\_RobotClass

Robot class is a java-based utility class used to simulating **KB** (key pressing and key releasing for copying, pasting, and entering)& **Mouse interactions** on screen for test automation and **to upload files** in the server while executing selenium scripts. Robot class throws AWT Exception because Robot class comes under java.awt class. Robot class enter the values where ever cursor is pointing. It lets the tester automate tasks that can’t be done using selenium’s built-in methods.

**@. To handle hidden elements in selenium WebDriver**

Where elements are not interactable due to hidden prob in those situations we use JsE and check with isDisplayed(). i.e. JavasrcriptExecutor js = (JavasrcriptExecutor)driver;

js.executeScript(“window.document.getElementByClassName(LV)”).click();

**N:** Hub is a server or central point in the selenium grid that controls the entire test execution on different machines.

**@. Node on Selenium Grid**

Nodes are called different remote machines (like diff OS with diff environment (different browsers, browser versions, java versions, etc)) these are attached to selenium grid hub and hub controls all the nodes. There are multiple nodes in Selenium Grid.

**Dis-adv‘s of Grid:** We need to maintain multiple nodes in multiple VMs because I node = to 1VM (nothing but physical machine) we need to add these VMs to central Hub server. (for ex VM1 contain windows OS and this having all browsers, VM2 contain mac OS and this having all browsers, VM3 contain Linux OS and this having all browser. Suppose if we want to run our script on Linux OS & chrome browser grid will choose VM3 or if we want to run our script on mac OS & chrome browser grid will choose VM2 or if we want to run our script on windows OS & ff browser grid will choose VM1, etc). In Selenium Grid all the entire thing (nodes (VMs), Hubs) is present in different VMs machines.

In case of docker every node is considered as a container instead of maintaining physical machines (VMs) we can just pull different containers based upon different configuration (like OSs, Browsers, etc). These containers will associate with one single **SeleniumHub** container and based upon the desired capabilities Hub will choose the proper container (that means proper OS, Browser, etc). In docker all the entire thing (containers, Hubs) is present in single machine.

**@. Explain the line of code WebDriver driver = new Firefoxdriver(); and purpose of ‘driver’ instance**

In the line of code **‘**WebDriver driver = new Firefoxdriver();**’** **WebDriver** is an interface so we can’t able create object to that interface directly so we are creating the WebDriver instance to the object of FirefoxDriver class (i.e., **new Firefoxdriver()**) then only we can able to access all the methods of both WebDriver and FirefoxDriver and also able to execute script on different machines. This entire concept is known as Runtime polymorphism. Mainly WebDriver is used to start a browser session.

**Use of WebDriver:** WD allow interaction with web browsers to automate web application testing, with the WD we can execute our tests against different browsers **and** used for providing connection between browser and local system.

**N:** WebDriver interface is used to provides a common APIs for controlling web browsers. It allows to write automated tests that interact with web applications.

**@. TestNG / Features / Significance of TestNG / list out some of the func’ty in TestNG which makes it more effective**

TestNG standards for Test Next Generation. It is a Built-in unit testing framework. It helps to organizing test execution. TestNG makes the automated test scripts are more structured, readable, maintainable and user-friendly. It can easily integrate with our automation scripts to perform multiple operations. It is introduced to overcome the dis-advantages of JUnit. By using TestNG we can perform End-to-End testing easily. ------adff hxpp adgp

**Features / Functionalities / Adv:** The following functionalities which makes it efficient testing framework those are

* Supports multiple Annotations.
* Allows **Data Driven** **Testing** (basically it is a combination of XLUtility along with @DataProvider).

**@DataProvider** is get the data from the Excel sheets or data bases and store the entire data into 2-D Array (i.e. the method annotated with @DataProvider annotation return a 2D array of object) and proving this data to all test methods. Ref: TestNG\TestNG\_with\_WD\EX6\_TestNG\_dataProvider

* Flexible test configuration
* Ability to Re-execute failed Test Cases (If we are using TestNG library after the batch execution by refresh on the project folder it generates a file called testng-failed.xml under test-output folder. We can get failed Test Cases logs from this file)
* Provide html reports
* With testing.xml we can control the execution of our test scripts.
* **Parameterizing** Test Cases from TestNG XML file by using @parameters annotation.
* It provides **Parallel** execution of test methods. ---- adff hxpp agdp
* TestNG provides different **Assertions** that helps to comparing the actual results with expected results.
* We can define the **Dependency** of one test method over other methods in TestNG.
* It allows **Grouping** the test methods into test groups its nothing but a managing diff automation test suites like Sanity, Regression. It enables selective execution. @Test(groups={“SmokeTest”, “SanityTest”,”RegressionTest”})
* We can assign **Priority** to test methods in selenium.
* Cross browser testing can be achieved by using @parameter annotation

**N: @Test** annotation is used to group Test Cases in TestNG.

@. Attributes used in TestNG

Attributes are mainly used inside the @Test ann and other TestNG ann’s to control how tests behave and execute. TestNG Attributes are: priority, dependsOnMethods, dependsOnGroups, enabled, groups, description, timeOut, invocationCount, invocationTimeOut, dataProvider, dataProviderClass, alwaysRun, expectedExceptions, retryAnalyzer, successPercentage, suiteName

**@. Use of testng.xml file**

testing.xml file is used to configure and triggering the test suite. In testing.xml file, we can create a test **s**uite, **g**rouping or tagging Test Cases, mark tests for **p**arallel execution, execution methods can **c**ontrol, allows to create cross **b**rowser testing, add **L**isteners and pass **p**arameters to test scripts (using @Parameter annotation & ‘<parameter>’ in testng.xml)

**@. Listeners in Selenium / Use of Listeners in TestNG / To customize html report using TestNG**

In Selenium, Listeners is an interface. By these we can perform some actions in case an event has triggered. Listener "listen" the event defined in the script and behave accordingly Listeners are used to allows customizing reports or to generate logs (to provide some details about our test). There are two type of listeners in selenium those are WebDriver listener and TestNG Listener: WebDriver listeners are particularly useful where we need to capture and process events emitted by the WebDriver. TestNG listeners are particularly useful where we need to perform operations before and after a test method is executed. One of the most widely used TestNG listener is **@ITestListner** interface and it has methods like onStart, onTestSuccess, onTestFailure, onTestSkipped, onFinish etc. these are overridden methods. We need to implements this **@ITestListner** interface creating a listener class of our own. onStart() Will execute first and execute only once before starting all the tests and remaining all methods are responsible to create entries in the report (either fail, pass, or skip etc). Best Ex is **@ITestListner** concept is used in ExtentReporterManager class (Ref: API FW). To generate extent report by executing Tests only through testing.xml (bcz we are integrating **Test class** and **ExtentReportManager class** in testng.xml) then only report will generate. Before <test> we have to add <Listners>.

**Methods in ITestListener:**

onTestStart(ITestResult result): Called when a test method is about to start.

Parameter: ITestResult result - Contains information about the test method that is starting.

onTestSuccess(ITestResult result): Called when a test method is successfully executed.

Parameter: ITestResult result - Contains information about the test method that passed.

onTestFailure(ITestResult result): Called when a test method fails.

Parameter: ITestResult result - Contains information about the test method that failed, including the exception.

onTestSkipped(ITestResult result): Called when a test method is skipped.

Parameter: ITestResult result - Contains information about the test method that was skipped.

onTestFailedButWithinSuccessPercentage(ITestResult result): Called when a test method fails but is within the success percentage defined in the TestNG configuration. Para: ITestResult result - Contains information about the test method.

onStart(ITestContext context): Called before any test method in a test class is executed.

Para: ITestContext context - Contains information about the test context, including all test methods and their statuses.

onFinish(ITestContext context): Called after all test methods in a test class have been executed.

Para: ITestContext context - Contains information about the test context, including all test methods and their statuses.

**@. @Factory and @DataProvider** Ref: Java\_WD\_Ex\TestNG\TestNG\_with\_WD\FactoryAnnotation

**@Factory** is a TestNG annotation used in dynamic execution of Test cases. By this we can pass parameters to whole test class at run time. That passed parameters can be used by one or more test methods of that class. Internal operation is @Factory method creates instance of test class and run all the test methods in that class with different set of data.

**@DataProvider** is a TestNG annotation used to provide multiple sets of data to a single test method so that specific test method will execute multiple times. It helps to create data-driven tests.

**N:** Return type of TestNG @DataProvider is Object[][].

**@. To run Test Cases in parallel using TestNG**

In order to run the tests in parallel we just need to add these two key-value pairs to our test suite

* Parallel= ”methods/classes/tests”
* Thread-count= ”number of threads / max parallel executions we want to run simultaneously”

**Syntax:** < suite name="suite\_name" thread-count="5" parallel="classes" >

**N:** To achieve thread safety in our automation framework while running tests in parallel we have to use one of the approaches is **doing parallel test execution with TestNG**. By specifying parallel and thread-count parameter either at <suite> level or <test> level.

**@. Name an API used for logging? in java Use of logging**

Log4j is open-source API widely used for logging while script is executing. It can be configuring with the help of Log4j.xml or Log4j.properties file. It supports multiple levels of logging those are –TRACE, DEBUG, INFO, WARN, ERROR, FATAL. (Lowest to Highest: TRACE < DEBUG < INFO < WARN < ERROR < FATAL < OFF)

TRACE – Most detailed logging, used for fine-grained debugging.

DEBUG – Used for debugging information.

INFO – General informational messages.

WARN – Indicates potential issues but not an error.

ERROR – Serious issues that need attention.

FATAL – Critical errors that may cause the application to terminate.

OFF – Disables logging.

**Use of logging in automation:** Logging helps to debugging the tests when required and also provides test’s runtime behaviour with date and time stamp.

**Type of Loggers:**

Console Logger: Outputs logs to the console

File Logger: Writes log to a file

Database logger: Store log in database

Remote Logger: Send logs to a remote server

**@. H selenium runs**

1. First create WebDriver instance.
2. Launch browser
3. Enter URL to navigate to webpage.
4. Locate an HTML elements on the webpage.
5. Perform require actions on that HTML element with the WebDriver commands.
6. Anticipate the browser response to that action.
7. Record test results using a test framework.
8. Conclude the test (Quit WebDriver instance)

**@. To Run testng.xml through the command prompt ---** https://youtu.be/Dc1yvMdhF2U

Perform the following steps for running testng.xml through the command prompt:

1. Create a java project with Test Cases and generate testing.xml file (RC on class 🡪 TestNG 🡪 Convert to TestNG).
2. Go to project directory 🡪 copy bin path 🡪 back 🡪 search CMD
3. set classpath=bin\_path; lib\_path\\*

F:\LT eclipse-workspace\RunFromCMD\bin **--->** bin\_path --- Project path from eclipse (after open ‘bin’ folder)

F:\LT eclipse-workspace\RunFromCMD\lib **--->** lib\_path --- Here I kept whatever jars require for project in new folder

1. E 🡪 java org.testng.TestNG testng.xml 🡪 E

**@. To Run selenium WebDriver script from command prompt**

1. Add JAR files to project
2. RC on class 🡪 Export 🡪 in Export window ‘Expand ‘java folder’’ 🡪 click on runnable JAR File 🡪 Next 🡪 Under ‘Launch Configuration edit box’ 🡪 select particular project with class 🡪 Brows to select location of Exporting jars 🡪 save 🡪 Finish 🡪 ok 🡪 ok
3. Open downloaded jars folder (Resultant JAR is single JAR) 🡪 select on that jar and type CMD on folder navigation edit box 🡪 E 🡪 java -jar TestOne.jar #(TestOne --- Exported JAR name) 🡪 E

**@. Selenium3 vs Selenium4 WD Arch / Selenium is now W3C complaint / Interaction of the Selenium client library with the real browser / H selenium test script interact with real browser / H communication channel is extablished.**

**Selenium 3 Architecture is made up of four major components:**

* 1. Selenium Client library / Language Bindings
  2. JSON wire protocol over HTTP
  3. Browser Drivers
  4. Browsers
* **Selenium3** client library is available in multiple programming languages including JAVA, C#, Python, JavaScript, etc. These client libraries interact with Browser Drivers and Browsers using the JSON (JavaScript Object Notation) Wire Protocol over HTTP.
* Selenium architecture uses JSON wire protocol to communicate between test script and the browser. It provides a transport mechanism for transferring data between client and server on the web through various data structures like arrays and objects used to read and write data from JSON by creating **WebDriver driver = new ChromeDriver();** here one section id will get create and this will happening at server side.
* JSON is a RESTful web service using JSON over HTTP to exchange information between HTTP servers.

**N:** JSON wire protocol was used before Selenium version 3.8. After Selenium 3.8, W3C protocol was introduced.

**Selenium 4 Architecture is made up of four major components:**

* 1. Selenium Client library / Language Bindings
  2. W3C WebDriver protocol
  3. Browser Drivers
  4. Browsers
* W3C protocol was introduced from **Selenium 4** because all the web browsers and browser drivers followed the W3C standards it can be integrated with selenium with no compatibility issues and to standardise the communication, JSON wire protocol was replaced by W3C. This helped in better communication with the browsers, stability, and common code (i.e. no browser specific code required).
* Due to W3C there is a direct transfer of information between client and server. With W3C WebDriver Protocol, the communication happens directly without any encoding and decoding required.

**@. W3C and JSON wire protocol**

W3C is a standard protocol used in selenium 4.0 to communicate with browsers, offering better compatibility and consistency across different browsers.

JSON wire protocol used in Selenium 3.0 which has been deprecated in favour of the W3C WebDriver standard.

**N:** After executed driver.quit() we cannot use driver once again because section id will going to ‘null’. If multiple windows present after executed driver.close() the section id is not null so we can use driver once again on other window. In case single window present the section id will going to null.

**N:** In case of multiple windows if one window is closed but we try to run getWindowHandles() method we will get section I'd is null so it throws NoSuchSectionException.

**@. H Assert statement works**

Assert stmt used for debugging purposes and it checks a boolean expression. If it is false, an AssertionError will throw.

**@. Assertion in Selenium / Type of Assertions / Assert, HA and SA / do u prefer SA or HA / To continue exec when stops in middle** Ref: Java\_WD\_Ex\TestNG\TestNG\_with\_WD\SoftAssert\_HardAssert

Assertion is used for verification (checkpoints) that is for comparing the actual result with expected result. These help to determine whether a Test Case is behaving as expected by comparing the actual result with expected result so testers can understand if the tests have passed or failed. There are two major assertion types are there:

|  |  |
| --- | --- |
| **Hard Assert** | **Soft Assert** |
| When an assert statement is fails Hard Assertion will throw an assert exception immediately and remaining tests are skipped and test will not continue | When an assert statement is fails Soft Assert won’t throw any assert exception immediately but it will collect all errors and show at the end and test will continue with the next test |

**N:** If we click on one link if it is not open then we go to click other link without script fails for that situation we use SA.

**N:** While working with SA, If we want to know about the exceptions **softassertobj.assertAll()** method will use and it will placed as a last step in the test.

**N:** Comparing to Hard Assertion Soft Assertion is more preferred.

**@. Some common HardAssertions provided by TestNG**

.assertEquals(ActStr, ExpStr, “to pass String msg in console”) //if script fail then only print Error msg

Ex: .assertEquals(true, ele.isDispalyed())

.assertNotEquals(double data1, double data2, “to pass String message in console”)

.assertNull(Object bject)

.assertNotNull(Object bject)

.assertFalse(Boolean condition, “to pass String message in console”)

.assertTrue(Boolean condition, “to pass String message in console” )

.fail(Boolean condition, “to pass String message in console”)

**@. TDD and BDD**

Both TDD and BDD are the software development techniques.

|  |  |
| --- | --- |
| **TDD (**Test Driven Development**)** | **BDD (**Behaviour driven development**)** |
| TDD is a software development pactice and it is for Unit testing (here will test each method) | BDD is a software development methodology and it is for E2E testing (here will test each scenario) |
| focuses on writing tests before code, primarily for developers | focuses on the behavior of the appln, involving collaboration bn dev’s, test’s, & business stakeholders |
| It is achieved by using JUnit and TestNG | It is achieved by using Cucumber and SpecFlow |
| In TDD more focus on Unit Test Cases | In BDD more focus on Acceptance Test Cases (these are may or may not provide by product owner) |
| It can understand only who can have programming knowledge | Everyone can understand this framework |
| - In TDD test the certain piece of functionality using prog lang’s  - First running the test to make them fail to confirm that the test fails for the right reason (e.g., missing or incorrect functionality) and then write the code to make the test pass, and repeat this process.  - TDD is also called as Test First Development because where tests are written before the actual code is implemented.  - The goal of TDD is to ensure that the software meets its requirements and behaves as expected, maintainable, and testable code. | - In BDD the process starts by writing a scenario as per the expected behaviour.  - Test scripts are designed by describing the behaviour of the application from the user’s perspective.  **N:** Gherkin keywords will use plain English language.  **Kw’s:** Feature, Rule, scenario, Scenario Outline, Given, When, Then, And, But, Examples, @tagname |

**N:** In case of 100TCs we can execute only 50TCs / To disable or prevent test scripts from running in TestNG with the help of **@Test(enabled=false)** and in .xml use method include and exclude to test method

**invocationCount** allow to exe any test method multiple times. It allows to set int value i.e., **@Test(invocationCount = 5)**

**threadPoolSize** attributeis used to defines the number of threads to be used while running a test method. It is used in conjunction with **invocationCount** attribute **@Test(invocationCount = 7, threadPoolSize = 3)**

**N:** JSON is not a programming language it is for storing and executing the data.

**N: @Test(AlwaysRun=true)** – under this annotation method will execute always in case of dependency test like dependsOnMethods or dependsOnGroups fail or get skipped. Ref: Java\_WD\_Ex\TestNG\TestNG\_with\_WD\EX8\_alwaysRun

**@. @Listeners** Ref: TestNG\Custom\_TestNGAnnotation2

**@Listeners** annotation is used to specify the customized listener class **@Listner(Pac\_name.custListnerClassName.class)** for that we need to implement ITestLitener interface by creating a listener class of our own. After that using @Listener annotation, we can use specify for that for a particular test class, our customized listener class should not be used.

**@. Skip Test Case conditionally** Ref: Java\_WD\_Ex\TestNG\TestNG\_with\_WD\EX7a\_Skip\_TC\_UsingTestNG

By using SkipException class provided by TestNG we can skip the TCs conditionally and throw the skip exception

**@. Need to done before start automation testing**

* At least one cycle of manual testing complete
* Application should be stable

**@. C selenium integrate with Jenkins** ---- Yes, by running test suits through Jenkins job

**@. Jenkins / To execute Test Suite after every 4 hrs.** ci/cd tool for running selenium tests in a pipe line

Jenkins is an open-source build automation tool. It is mostly used by the DevOps team to automate the build, to deploy the build, and to test the build to know whether the new code is breaking something in the software or not. Gherkins programming language is used by Jenkins. For this DevOps team create a pipe lines in multiple stages those are:

1. Automation code will get from GitHub repository by URL
2. Creates the environment for execution.
3. Find some defects and those defects will logged into test management tools
4. Reports will get generated

**To schedule Test Suite execution:**

Configure a Jenkins Job 🡪 In the job configuration, under “Build Triggers”, select “Build periodically” 🡪 Use the cron syntax to schedule the job every 4hrs. **Ex:** **H\*/4\*\***

CI part contains: developer will continuously deploy the code and build the code and tester conduct testing. CI part can achieve with Jenkins.

CD part contains: after testing completes deploy/release that product to the customers

Pipeline: coding by developer -> build that code -> testing that build -> deploy the build

Most of the time sanity and regression TCs will execute in Jenkins.

**N:** Jenkins integrates with test framework to trigger test runs automatically in CI/CD pipelines. CI/CD help to automate testing in software development pipelines.

**Continuous Integration (CI)** is a practice of automatically integrating code changes from multiple developers into a shared repository multiple times a day. Each integration is verified by automated builds and tests.

**Continuous Delivery (CD)** extends CI by ensure that every successful build is automatically tested and ready for release to production at any time. However, the release is still a manual decision.

**Continuous Deployment (CD)** goes a step beyond Continuous Delivery by automatically deploying every successful build to production without manual approval.

**@. H framework integrated with CI/CD**

Our automation suite is integrated with Jenkins so that test execution will trigger automatically on every code push via the pipeline. We run test in parallel on Selenium Grid and push allure reports post-execution. Critical tests run on every build, while full regression runs nightly.

**@. If a particular Test Case does not get executed with in the specified time then how to mark the Test Case is fail and move forward how will achieve this using TestNG**

**timeOut** is attribute in TestNG and it is used to represents the max time a mtd can wait for exec If the Test Case does not get executed within the timeOut value then it will fail with the TimeOutExcp. It allows int value. **@Test(timeOut=90)**

**N:** This parameter can use if we want to complete the automation with in the specified time.

**@. Wn selenium server requires to run Tests in selenium WD / Is selenium server require to run WD scripts**

* When we are using remote or virtual machine to run WD scripts if that specific browser is not in our local machine.
* When we are using selenium grid to distribute our Test Cases Execution on different Remote or virtual machines.

**@. Verification points in selenium -----** TB – Text box

In selenium WebDriver there is no built-in features for verification points. It’s totally depending upon our coding style. Some of the verification points are: To check for Page Title, for certain Text, for certain element (TB, DD, button...…).

**@. Super interface of WebDriver** ----- SearchContext

**@. Use of creating the reference variable of type WebDriver (i.e., WebDriver driver)**

By creating the reference variable for WebDriver, we can use this same reference variable to work with any browser.

**@. To create Feature file in cucumber / Feature file format**

RC on package 🡪 new 🡪 others 🡪 select ‘file’ under General folder and save with .feature extension

Create feature file and the **run the Future file** by Run as ‘Cucumber feature’ all **unimplemented methods will generate** in console. To execute unimplemented methods must execute feature file not the TestRunner file. At that time

Feature file uses Given-When-Then steps Ex: Given user logs in When clicks profile Then sees profile page.

**@. Selenium 4 New Features**

Selenium 4 is the most recent update to the Selenium test automation framework, and it introduces a number of key enhancements and features that distinguish it from earlier versions, such as,

1. W3C WebDriver protocol
2. Upgraded Selenium IDE: Selenium IDE is record and playback tool, now comes with more robust cross-browser support and modern features like playback control, easier TC debugging, and a structured command interface.
3. Enhanced Selenium Grid
4. Enhanced Actions Class
5. Enhanced Browser Driver Management:
6. Enhanced Documentation (Selenium 4+ continues to improve its documentation, making it easier for users to understand and utilize the features and functionalities of the framework)
7. Relative Locators: Locating elements based on position relative to others those are above(), below(), near(), toLeftOf(), toRightOf().

**Ex:** WebElement ele = driver.findElement (RelativeLocator.with (By.id ("eleA")).below (By.id ("eleA")));

1. Improved browser Window and Tab Management.
2. Native Support for Chrome DevTools Protocol (CDP): It allows to capturing performance metrics, network logs, simulate network throttling, and console logs using chrome internals.
3. Deprecation of DesiredCapabilities (DesiredCapabilities is replaced by Options classes such as ChromeOptions, FirefoxOptions, etc.)

* Upgraded Selenium IDE: Previously it is supported as extension for FF browser only now Selenium IDE is add-on for FF, CH, MS Edge browsers.
* WebElements are locate nearby elements by specifying directions using Relative Locators. Those are: above, below, toLeftOf, toRightOf, near.
* Better window/tab management

Work with multiple windows/tabs in the same session.

We can open multiple windows/tabs without creating new driver object.

* Introduced new APIs for CDP (Chrome DevTools Protocol).

**Selenium 3:** Inspect Elements in DOM, Mock faster/slower N/W speeds, Mock geolocations of the user, etc. **Selenium 4:** Capture and monitor N/W traffic, simulate N/W conditions, perform geolocation testing, etc

**N:** The Chrome DevTools Protocol Integration is a significant enhancement introduced in Selenium 4 that allows testers and developers to interact with the Chrome DevTools Protocol directly from their Selenium scripts. This integration enables advanced debugging, profiling and analysis of web pages during automated testing.

* Enhanced Actions class: The Actions API in Selenium 4+ is designed to be more reliable and consistent across different browsers, enabling more complex user interactions.

Click on WebElement: moveToElement(onElement).click() 🡪 click(webelement)

Click on element without releasing click: moveToElement(onElement).clickAndHold() 🡪 clickAndHold(webelement)

Right click: moveToElement(onElement).contextClick() 🡪 contextClick(webelement)

release() function is now part of the Actions class

* Some updates in Selenium IDE.

(Older it is limited to FF only).

**@. To resolve merge conflict Git** ---- **https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/addressing-merge-conflicts/resolving-a-merge-conflict-on-github**

When people make different changes on same line on same file on different branches in Git repository. We can only resolve the merge conflict only on GitHub using the **conflict editor** or on CMD line.

Steps to resolve GMC: Pull requests 🡪 Resolve Conflicts 🡪 Our conflict code will show in ‘conflict editor’ 🡪

Decide that we want to keep only our branch's changes, keep only the other branch's changes, or make a brand-new change, which may incorporate changes from both branches. Delete the conflict markers <<<<<<<, =======, >>>>>>>  and make the changes we want in the final merge.

If we have more than one merge conflict in our file, scroll down to the next set of conflict markers and repeat steps to resolve the merge conflict.

click **Mark as resolved** 🡪 click **Commit merge** 🡪 click **Create branch and update my pull request** or **I understand, continue updating BRANCH** 🡪 click **Merge pull request**.

**Conflict will occur** mainly two reasons those are:

There are two separate branches got changes into the same line in the file.

When a particular file is deleted in one of the branches but it has got modified in another branch.

**@. Hooks in Cucumber**

In Cucumber, hooks are blocks of code that run before or after each scenario or test step. They are commonly used to set up preconditions or perform clean up tasks (teardown). Type of Hooks in cumber: @Before (Runs before each scenario), @After (Runs after each scenario), @BeforeStep (Runs before every step), @AfterStep (Runs after every step), @BeforeAll and @AfterAll.

@. Tagging in Cucumber

It is a powerful feature that helps to control execution (run or skip certain scenarios) (organizing), to group TCs (Ex: smoke, regression) (filtering), and to integrate with CI/CD pipelines for different test suites (running) specific sets of scenarios or features. Tags are simply annotations prefixed with @ and can be added above Scenarios, Scenario Outlines, or Feature headers in our .feature files.

**Running Tagged Scenarios:**

1. Single Tag: cucumber --tags @smoke

2. Multiple Tags - AND condition: cucumber --tags "@smoke and @regression"

3. Multiple Tags - OR condition: cucumber --tags "@smoke,@sanity"

4. Exclude a Tag: cucumber --tags "not @wip"

**@. Dry Run and Strict in cucumber**

In Cucumber, dryRun and strict are two important execution options that helps to control how features and step definitions behave during test execution.

**dryRun:** During development or debugging, it is used to check to verify that all steps in your feature file have corresponding step definitions without actually running the tests.

**Syn:** Set dryRun = true in the @CucumberOptions.

**strict in Cucumber (Deprecated in newer versions):** Fails the execution if any step is missing/undefined

**Syn:** Set strict = true in the @CucumberOptions.

**@. Does cucumber support TestNG**

No, Cucumber does not officially support TestNG. Cucumber is designed to work with JUnit as its default test runner, and most of its official integrations and plugins are built around JUnit. From recent versions (Cucumber 5.x and later) have dropped direct support for TestNG.

**@. BDD structure / Structure of a Feature File**

BDD structure typically includes:

**Feature** keyword is used to describes the high-level functionality or feature being tested.

**Scenario Outline** keyword is used to define a template for scenario’s with multiple sets of data (data-driven testing).

**Scenario** keyword is used to represents a single Test Case.

**Examples** keyword is used to Provide the data sets for Scenario Outline.

Actual Gherkin keywords used in “Steps” (Given, When, Then, And, But). Where,

**Given** keyword is used to describe the preconditions or context required for a scenario.

**When** keyword is used to describe the action or event that triggers a behaviour in the application. It describes the main action being performed in the test i.e. "what the user does" or "the event being tested" in a scenario.

**Then** keyword is used to define the expected outcome or result after the action is performed. Here we check that the application behaves as expected after certain actions have been performed.

**And** keyword is used to add more conditions or actions to any of the Given, When, or Then steps. It helps make scenarios more readable when multiple steps are of the same type.

**But** keyword is used to specify a negative condition or an exception in a scenario. It helps make the Test Case more expressive and clearer when describing the behaviour of a feature.

**@. Types of reports**

HTML Reports: Detailed and formatted reports viewed in web browser.

XML Reports: Structured data reports often used for further processing.

JUnit Reports: XML reports compatible with JUnit results.

Allure Reports: Rich, interactive reports with detailed test results and analytics.

**@. Interfaces and classes used in Automation.**

**WebDriver Interface:** Main Interface for browser interaction.

**WebElement Interface:** Represents HTML elements.

**TakesScreenshot Interface:** For capturing screenshot.

**JavavaScritExecutor Interface:** For

**Alert Interface:** For

**SearchContext Interface:** Parent of WebDriver and WebElement

**By Class:** used for locating elements.

**@. Interfaces and classes used to perform click, sendKeys, scroll, swipe, Drag and Drop.**

Click: WebElement.click(); WebElement - Interface

sendKeys: WebElement.sendKeys(); WebElement - Interface

scroll: JavaScriptExecutor.executeScript(); JavaScriptExecutor - Interface

swipe: TouchActions or Actions class Ref: Java\_WD\_Ex\src\WD\_Examples\EX68\_Swipe\_by\_ActionsClass

Drag and Drop: Actions.dragAndDrop(); Actions – class

**@. Deal flaky tests in selenium automation suite using TestNG**

First analyse the root cause of flakiness – is it the application (unstable locators), environment, or test script. A flaky test refers testing that generates inconsistent results, failing or passing unpredictably, without any modifications to the code under testing. A flaky test can handle by using explicit waits, improve locator strategies, stabilize test data, avoid hardcoded sleeps, isolate unstable tests, and also use RetryAnalyzer feature in TestNG by this a failed test should be retried based on certain conditions such as specific exception or test result statuses. This helps to improve reliability of the test suite by running failed tests automatically.

**@. Approach to implementing API tests for a new feature**

First review the API documentation to understand the end points, require parameters and expected responses. Then design the TCs covering various TSs, including positive and negative Test Cases, boundary conditions and error handling. Using tools like Rest Assured to automate these TCs and integrate them into CI/CD pipeline using Jenkins. Finally execute the tests with different environments and analyse the results to ensure the feature’s reliability and stability.

**@. C selenium locate elements inside shadow DOM**

**Shadow DOM** is nothing but a DOM inside DOM / hidden DOM. Selenium not directly locate elements inside Shadow DOM with standard locators we have to use JavaScript queries to access shadow roots. **Shadow Host**: node of the shadow DOM, **Shadow Root**: root node of shadow tree.

To find in DOM console **Syntax for JavaScript path:** document.querySelector('settings-ui').shadowRoot.querySelector('cr toolbar#toolbar').shadowRoot.querySelector('#search').shadowRoot.querySelector('input') ---- URL: chrome://settings search box field ----- Ref: WD\_Examples\EX63\_ShadowDOM

**@. Uses of JsE / To search JsE from DOM Console**

**Uses:** Capture WEs Screenshot, Border WEs, Enter Input text in edit box, Clicking WEs, flash WEs, Scroll-up & down, Scroll into element, open URLs in new tabs and windows, generate an Alert, Zoom-in & Zoom-out.

Js.executeScript(“location.reload()”); --- to refresh browser window

Js.executeScript(“document.getElementById(‘id\_value’).value=’Alphabin’;”); --- to send text

Js.executeScript(“alert(‘hello world!’);”); --- to generate Alert popup window

String text = Js.executeScript(“return document.documentElement.innerText;”).toString(); -- get inner text of a webpage

Js.executeScript(“return document.title;”).toString(); --- to get title of webpage

WebElement loginBtn = driver.findElement(By.id(“id\_value”));

Js.executeScript(“arguments[0].click();”, loginBtn); --- to click an element

Js.executeScript(“window.resizeTo(1024, 768)”); --- to set browser window size to particular dimension

**Cucumber Plugins require for IDE** – Cucumber Eclipse Plugin

**Required dependencies for cucumber:** https://cucumber.io/docs/cucumber/api/?lang=java

cucumber-core

cucumber-java

cucumber-junit

cucumber-jvm

cucumber-jvm-deps

cucumber-picocontainer

gherkin

junit

testng

**N:** We can use JUnit to run Cucumber tests add the cucumber-junit dependency to our project. If we are using JUnit 5, add the junit-vintage-engine dependency as well.

**N:** Cucumber-JVM supports parallel execution across multiple threads since [version 4.0.0](https://cucumber.io/blog/announcing-cucumber-jvm-4-0-0/), by achieving this requires some configuration. Cucumber can be executed in parallel using JUnit and Maven test execution plugins. In JUnit the feature files are run in parallel rather than scenarios, which means all the scenarios in a feature file will be executed by the same thread.

**Order of execution in cucumber:** Feature file contain number of steps and to implement feature file we need step definition file and this file internally call the Page Objects class.

Cloning in Jira: if want to clone particular testcase and if we want to add them to the different test cycle then we can do that.

**Colour indicators in Jira**

Green: no prob with particular user story i.e., clean without any defect

Orange: there is non-blocker issue

Blue: means not completed but not having any defects

Red: it has so many non-blocker issues going to reject

Managers will use these colour indicators QA engineers will not use

Feature file contain multiple features and each feature contain multiple scenarios and each scenario contain multiple steps and these steps will write in gherkin keywords

After creation of feature file every scenario and steps we need to create a step definition file. It is just java class. In step definition file we write the corresponding methods of each steps in feature file it's nothing but all the steps implementing in the step definition file. And run this particular step definition file.

If we have single feature file or specific scenario that can run directly. Suppose if you have multiple feature files then you have to create one test runner class (JUnit class) (main class). This test runner file will connect to feature file and this feature file and feature file steps will be read and these steps will again use in step definition file.

Feature file (nothing but folder) with .feature extension will create anywhere like project level, test level .......

step definition class and test runner class (JUnit class) will create under java folder

**N:** In eclipse we can’t run the Scenario alone but it possible in IntelliJ IDE

**N:** Always folders will create at project level

**Git Commands:**

go to project directory

RC --> open 'Git Bash Here'

git init - to initialize repository (It will create a local empty repository) -------- 1time activity

git config --global user.name "BalakrishnaAratipamula" (GitHub UserName) -------- 1time activity

git config --global user.email "balakrishnmvn@gmail.com" (GitHub repo email) -------- 1time activity

git status - it will show what are all the folders ready to commit

* untracked files - are not part of staging area and these are ready to add staging area or git repository (will show in Red colour)
* tracked files - are already part of tracked files or git repository

git add -A - -A represents add all files to staging area / If we want to commit specific file use git add file\_name command

git status – this time status is green colour.

git commit -m "must write some comment msg" - commit all files into local git repository (staging area to local Git repo) (((((stagging to Git repo))))) 🡪 Enter 🡪 all files will show in white colour

git status – nothing to commit

**::** Create on empty rep in GitHub (rep name is normally the same name of our project in eclipse) 🡪 copy this **Repo URL**

git remote add origin "https://github.com/BalakrishnaAratipamula/CucumberDemoProject.git (**Repo URL**)" - to integrate Git with GitHub (provide connection bn Git and GitHub) ((((working dir to staging area)))) -------- 1time activity

git push -u origin man/master / git push origin branch\_name - send all the files from git to GitHub repository ((((((git to GitHub repo))))) 🡪 100% complete

**::** If we want to switch branchX to main git branch -M main

------ Go to remote repository and refresh URL

Eclipse activity:

git clone GitHub\_Project\_URL - clone project into working directory which are not available in working directory

file --> Import --> General --> Existing projects into work space --> Next --> Browse --> Select project (which project we want to export) --> Next/Finish

copy work space location "F:\LT eclipse-workspace\CucumberA"

go to project directory

open 'Git Bash Here'

git pull origin master - remote repository to local system

clear - to clear

cd repo\_name – to focus that particular repository

git branch branch\_name – to create branch in repository

git branch – it will show all the existing branches in repository

git checkout branch\_name – to focus that particular branch

**N:** if we want create file in GitHub Add File --> create new File --> give extension .java if it is class --> commit new file

**N:** To disconnect Git connection: RC on proj --> Team --> Disconnect

**N:** Git status commands is used to show the difference between the working directory and the index

**Eclipse Shortcuts:**

ctrl+shift+t – Open Type (search for class to see **internal implementation**)

Shortcut for keywords in **Feature file**: ctrl+space

ctrl+shift+c – inspecting window **|** ctrl+shift+i – inspecting window

ctrl+shift+o – to import package

**@. Mtds avl in WD:** get(), cose(), quit(), findElement(), findElements(), switchTo(), manage(), getTitle(), getCurrentUrl()

**@. Interface available in Selenium:** WebDriver, WebElement, TakesScreenshot, JavaScriptExecutor, Alert

**@. Classes available in Selenium:** WebDriverManager, WebDriverWait, RemoteWebDriver, Actions, Select

**@. Classes available in Selenium:** AbstractWebDriverEventListener

**@. System.setProperty**

Here System is a class and setProperty is a method it accepts 2 arguments i.e. key and path. Where key represents in which browser we are going to test the application and path represents the location of the driver executable file.

**149. Write code to click on the Check box which is inside the Dynamic Table ----** Ex: date field <https://www.guru99.com/handling-dynamic-selenium-webdriver.html>

Git Install - <https://youtu.be/E6-YSidPCu0>

Git from CDM to capture file - <https://youtu.be/9FOuyNt0V8I>

Git from CDM to capture code - <https://youtu.be/NJPkHtvyAIQ>

Git from IDE to capture code and merge - [https://youtu.be/HCeBd5GKNO8](IQ%20on%20M&A&CJ&SQL.docx)

**Feature File:** A Feature File in Cucumber is a text file that contains a description of a feature or functionality of the systembeing tested. It is written in Gherkin syntax and contains one or more Scenarios that describe the behavior of

the feature.

**Back Ground Keyword:** Background is keyword used to define steps or series of steps that common to all scenarios in feature file. It allows to specify a set of steps that should be executed before each scenario, it can reduce duplication

and making Tests more concise

**Hooks:** Hooks in cucumber is used to block code that can be executed Before and After each scenario, Feature or

Step. They are used to perform setup and tear down tasks such as

->Setting the data before each scenario

-> Clear the data after each scenario

->logging information before, after each step

**Example Table:** Is a way to provide multiple set of data set for single scenario and it can allow you to test the same scenario with different set of data

**Data Table:** Data table in cucumber is used to provide the collection of data in table format and it used to pass the to the step Definition and it easy way the test the scenarios.

**Scenario Outline:** A Scenario Outline in BDD Cucumber is a template for a scenario that can be reused with different data sets. Itallows to running the same scenario with multiple sets of data using Examples.

Scenario. A Scenario in BDD Cucumber is a description of a specific behavior or feature of the system being tested. It

consists of a series of steps that describe the actions and expected outcomes.

**BDD Cucumber Framwork:**

**1. Feature Files**

- Located in: src/test/resources/features

- Contain: High-level descriptions of features and scenarios in Gherkin syntax (.feature files)

- Example: login.feature, search.feature

**2. Step Definitions**

- Located in: src/test/java/stepdefinitions

- Contain: Java code implementing the steps defined in feature files

- Example: LoginStepDefinitions.java, SearchStepDefinitions.java

8/21/24, 12:16 PM Online Notepad

https://onlinenotepad.org/notepad#google\_vignette 1/2

**3. Page Objects**

- Located in: src/test/java/pageobjects

- Contain: Java classes representing web pages or UI components

- Example: LoginPage.java, SearchResultPage.java

**4. Test Runner**

- Located in: src/test/java/testrunner

- Contain: Java class running the Cucumber tests (e.g., TestRunner.java)

- Annotated with: @RunWith(Cucumber.class)

**5. Configuration**

- Located in: src/test/resources/config

- Contain: Configuration files for Cucumber (e.g., cucumber.properties)

**6. Reports**

- Located in: target/cucumber-reports

- Contain: Extent Reports (for detailed HTML reports)

**8. Maven** is a build automation tool used to manage Cucumber projects. Terms used in maven:

Build-->Compile and package the project.

Test-->Run Cucumber tests.

Report--> Generate test reports.

**Maven Configuration for Cucumber**

pom.xml-->Maven's configuration file.

Dependencies-->Declare Cucumber and other dependencies.

Plugins-->Configure Cucumber plugins (e.g., maven-surefire-plugin for test execution).

8/21/24, 12:16 PM Online Notepad

https://onlinenotepad.org/notepad#google\_vignette 2/2

**@. Radio button and checkbox**

|  |  |
| --- | --- |
| **Radio button** | **Checkbox** |
| In radio button we have to select at least one option | In checkboxes we can select more than one option |
| For deselection we have to select another option present | For deselecting we have to select the same optn one more time |

**@. Base class method for clicking an WebElement**

Public static void btnClick(WebElement ele){ ele.click(); }

**@. Print a paragraph from webpage**

By using XPath functions text(), contains() or using getText() we can get the text **Ex:** ele.getText();

**@. moveToElement() and switchTo()**

moveToElement() will move to the particular element.

switchTo() is used to move the control to an alert, frame, or window.

**@. Methods used for scrollUp and scrollDown**

executeScript() method from JavaScriptExecutor interface is used for scrollUp and scrollDown operations.

arguments[0].scrollIntoView(true) ---- argument needs to passed for scrollDown operation.

arguments[0].scrollIntoView(false) ---- argument needs to passed for scrollUp operation.

**@. Code to print all options from dropdown**

getOptions() is used to get all options in dropdown and it will return List<WebElement> then iterating each WebElement using for loop to get the text using getText().

**@. Can we select multiple values in dd -----** Yes, we can select multiple values in a dropdown if it is multi select dd

**@. Return type of some methods available in Select class**

selectByValue(), selectByVisibleText(), selectByIndex(), deSelectByValue(), deSelectByVisibleText(), deSelectByIndex(), deSelectAll() --- **void** is return type

getFirstSelectedOption() ---- **WebElement** getOptions(), getAllSelectedOptions() --- **List<WebElements>**

**@. When multiple windows opened, H directly switch from parent window to 8th window**

List<String> allWindowsId = (List<String>)driver.getWindowHandles();

Driver.switchTo().window(allWindowsId.get(7));

**@. Code to display all column names in a web table**

List<WebEement> rowRef = driver.findElements(By.tagName(“tr”));

WebElement firstRow = rowRef.get(0);

List<WebEement> colRef = firstRow.findElements(By.tagName(“td”));

**@. defaultContent() and parentFrame()**

defaultContent() is used to returns the control to the main webpage

parentFrame() is used to return the control to the parent frame

**@. Ways to handle dropdowns**

We can handle dds using Select class for <select> elements which allow us to choose by index, visible text, or value

For non-standard dds XPath or CSS selectors combined with actions like click() can be used.

**@. To text mobile web application using selenium** **---** Selenium can work with Appium to test mobile browsers.

**@. To optimize selenium scripts for performance / To optimize test scripts to improve execution speed?**

Using only necessary waits and minimizing hard waits, using efficient locators, avoid unnecessary navigation, conduct parallel execution with selenium grid or Docker to speed up tests.

**Optimizing test scripts for execution speed** is a key part of making automation efficient and reliable. Below are the main strategies we can use:

1. Optimize Element Locators

* Use efficient locators: Prefer id, name, or cssSelector over complex xpath.
* Avoid brittle locators: Don’t depend on dynamic attributes like indexes or autogenerated IDs.
* Leverage relative locators (Selenium 4) where possible.

2. Minimize Waits

* Avoid Thread.sleep(): It adds unnecessary delays.
* Use Explicit/Fluent Waits only when needed.
* Set smart implicit wait just once instead of repeatedly.
* Use polling intervals with FluentWait to reduce idle time.

3. Reduce Browser Interactions

* Batch validations instead of multiple small assertions.
* Navigate once: Avoid repeated page reloads if not needed.
* Perform operations via API (setup/teardown data) instead of UI steps.

4. Parallel Execution

* Use TestNG / JUnit parallel execution to run tests simultaneously.
* Selenium Grid, Docker, or cloud platforms (like BrowserStack, SauceLabs) to scale tests.

5. Reuse Browser Sessions (if applicable)

* For login-heavy apps, use cookies/tokens to skip login for every test.
* Apply driver reuse strategies where isolation isn’t mandatory.

6. Optimize Test Data Handling

* Use test data providers (e.g., DataProvider in TestNG) instead of hardcoding.
* Generate test data with APIs or lightweight utilities instead of UI forms.

7. Page Object & Utilities

* Implement Page Object Model (POM) or Screenplay Pattern to avoid duplicate code.
* Reuse common utilities (e.g., custom waits, screenshot functions).

8. Selective Execution

* Use tags/groups in TestNG/JUnit to run only relevant tests.
* Maintain a smoke suite for quick checks, and a regression suite for full coverage.

9. Headless Execution

* Run tests in headless mode (Chrome/Firefox) where UI rendering isn’t needed.
* This can cut execution time by ~20–30%.

10. Continuous Monitoring & Refactoring

* Identify slowest tests via reporting tools (Allure, ExtentReports).
* Remove redundant validations or merge small tests.
* Regularly refactor scripts to adapt to app changes.

**@. Y choose hybrid framework, y not other frameworks**

Because my project is having many modules which also required test data to test the scenarios.

**@. Ur role / contribution in framework**

In framework I implemented few page classes, implemented extent reporting, implemented few generic methods (like waitForpageToLoad, waitForElement.

**N:** Without framework code will be duplicate, takes more time for automation and complex to modify the code to other members in team.

**@. @FindBys and @FindAll**

|  |  |
| --- | --- |
| **@FindBys** | **@FindAll** |
| All conditions must match.  @FindBys({  @FindBy(className = "header"),  @FindBy(tagName = "h1")  })  private WebElement headerTitle; | Any one of the conditions must match.  @FindAll({  @FindBy(className = "button-primary"),  @FindBy(className = "button-secondary")  })  private WebElement button; |

**N:** **Syntax of xml** ---- <suite> <test> <classes> <class> nothing but hierarchy

**N:** Common methods create in Excel utility file --- getRowCount(), getCellCount(), getCellData(), setCellData(), fillGreenColour(), fillRedColour().

**N:** Ways to execute dependent test scripts ---- 2 ways: **priority** and **dependencies**

**N:** Whenever we get build which test scripts we execute first --- Using grouping concept, we will execute smoke test first

**@. Batch execution and to achieve batch execution**

Collection of multiple tests is called batch and it can be achieved through testing.xml

**@. WebDriver and RemoteWebDriver**

**WebDriver** is the main interface in Selenium that represents a web browser instance. It is used to automate interactions with web applications. It is for local bowser automation.

**RemoteWebDriver** is a class that implements the WebDriver interface on the remote server that is designed to execute tests on remote server, such as a Selenium Grid or cloud-based testing platforms.

**@. Y can't we write "RemoteWebDriver driver = new ChromeDriver();"**

**Reasons:** Loss of WebDriver Methods **&** Polymorphism Best Practice (WebDriver driver = new ChromeDriver();)

**To generate GitHub token:** Settings - Developer settings - Personal access token - Tokens (classic) - Generate new token - Generate new token (classic) - enter GitHub password then hit confirm - give some identifier in 'note' edit box - select scopes (repo, admin:org, user:email) - Generate token

**@. Role of testing in software development**

Checking requirements, finding bugs, preventing new bugs, ensuring quality, improving security, making software faster, better user experience (ensure software is easy to use and user friendly), saving money.

**@. Selenium WebDriver: Core Interface, Classes and Utilities**

**1. Core Interfaces and Classes of Selenium WebDriver**

**1.1. WebDriver** (I)**:** The main interface for interacting with web browsers.

Browser-specific Implementations: CD, FirefoxDriver, EdgeDriver, SafariDriver, OperaDriver, InternetExplorerDriver

**1.2. WebElement** (I)**:** Represents an element on the web page.

Common Methods: click(), sendKeys(), getText(), isDisplayed(), isEnabled(), isSelected()

**1.3. By** (c)**:** Provides mechanisms to locate elements.

Common Methods: By.id(), name(), className(), cssSelector(), xpath(), linkText(), partialLinkText(), tagName()

**2. Action and Interaction Classes**

**2.1. Actions** (c)**:** Used for complex user interactions.

Common Methods: moveToElement(), clickAndHold(), dragAndDrop(), release(), build() and perform()

**2.2. Robot** (c)**:** Simulates keyboard and mouse actions at the OS level.

Common Methods: mouseMove(), keyPress(), keyRelease(), mousePress(), mouseRelease()

**2.3. Select** (c)**:** Handles dropdown elements with <select> tags.

Common Methods: selectByIndex(), selectByValue(), selectByVisibleText(), getFirstSelectedOption(), getOptions()

**3. Wait Classes**

**3.1. WebDriverWait** (c)**:** Used for explicit waits.

Common Methods:until()

**3.2. FluentWait** (c)**:** Provides flexibility with polling intervals and custom exceptions.

**4. Utility and Support Classes**

**4.1. JavascriptExecutor** (I)**:** Interface for executing JavaScript in the browser.

Methods: executeScript(), executeAsyncScript()

**4.2. TakesScreenshot** (I)**:** Interface for capturing screenshots.

Methods: getScreenshotAs()

**4.3. Cookie:** Used to work with browser cookies.

Methods: addCookie(), deleteCookie(), getCookies()

**4.4. Capabilities and DesiredCapabilities:** Specify browser properties and configurations.

**N:** DesiredCapabilities is deprecated; use Options classes like: ChromeOptions.

**4.5. ChromeOptions, FirefoxOptions, EdgeOptions** (c)**:** Configure browser-specific settings such as headless mode, arguments, and preferences.

**4.6. Alert** (c)**:** Handles browser alerts.

Common Methods:accept(), dismiss(), getText(), sendKeys()

**5. Selenium Grid Classes**

**5.1. RemoteWebDriver** (c) is a class that implements the WebDriver interface. Used for executing tests in a remote Selenium Grid setup.

**5.2. Sessionld:** Represents the unique session identifier for remote browser sessions.

**6. Logging and Event Classes**

**6.1. LogEntry:** Represents a single log entry.

**6.2. LogType:** Represents different log types (e.g., browser, driver, client).

**6.3. EventFiringWebDriver (Deprecated):** Used for listening to WebDriver events (replaced by other event listeners).

**7. Exception Handling Classes**

NoSuchElementException

TimeoutException

StaleElementReferenceException

ElementClickInterceptedException: Another element is overlying the intended click target.

WebDriverException

InvalidSelectorException: Indicates a malformed XPath or CSS selector.

SessionNotFoundException: When attempting to use a WebDriver instance after driver.quit() has called.

javascriptException: When an invalid script is executed.

**8. Other Classes**

**8.1. Dimension and Point:** Used to get or set the size and position of the browser window or elements.

**8.2. FileHandler** (c)**:** Used for file-related operations like saving screenshots.

Methods: copyFile()

**8.3. OutputType** (Type Parameter)**:** Used with TakesScreenshot to define the output type (e.g., FILE, BASE64, BYTES).

**9. Frequently Used Imports in Selenium**

import org.openqa.selenium.By;

import org.openqa.selenium.WebDriver;

import org.openqa.selenium.WebElement;

import org.openqa.selenium.chrome.ChromeDriver;

import org.openqa.selenium.firefox. FirefoxDriver;

import org.openqa.selenium.interactions.Actions;

import org.openqa.selenium.support.ui.WebDriverWait;

import org.openqa.selenium.support.ui.Select;

import org.openqa.selenium.Cookie;

import org.openqa.selenium.Alert;

import org.openqa.selenium.JavascriptExecutor; import org.openqa.selenium.TakesScreenshot;

import org.openqa.selenium.io.FileHandler;

import org.openqa.selenium.remote.RemoteWebDriver;

**N:** The **InvalidCertificateException** (or SSLHandshakeException in Java) occurs when a website's SSL/TLS certificate cannot be validated or trusted by our application.

**N:** If priority is not specified for test methods in TestNG, the test methods will be executed in their alphabetical order based on their method names. Ref: TestNG\_with\_WD\EX2b\_Priority\_not\_specified\_to\_testMethods

**N:** TestNG allows to assign negative priority values to test methods. TestNG executes test methods in ascending order of their priority, so methods with negative priorities are executed before those with positive or zero priority.

**@. Default priority of Test Cases in TestNG**

When not specified the priority the default value is ‘0’. For Ex if we have one TC with priority ‘1’ and one with out any priority then the test without any priority value will be execute first (because the lower priority will execute first).

**@. Execution format of tests in TestNG**

The execution of tests follows a specific order that depending on the configuration in xml file, annotations, and dependencies defined in the test suite.

**dependsOnMethods:** Under this annotation method execution is depends on successful execution of one or more specific test methods.

**dependsOnGroups:** Under this annotation method execution is depends on successful execution of all methods in specified groups.

**Ways to execute tests in TestNG:** Direct execution from IDE, using testing.xml file, using command line, by configuring maven-surefire-plugin in pom.xml we can execute tests through pom.xml file, integration with CI/CD.

**@. About Framework**

We have created a single hybrid framework that help us to test UI of our entire system

Our git branching model helps us making sure that everyone is working on the updated piece of code and the test code become part of CT pipe line.

Parameterized job in Jenkins:

The Jenkins plan that we have created as a parameterized Jenkins plan

Runtime arguments required for framework:

* 1. Test type (sanity / E2E)
  2. Component type (API / UI / Android)
  3. Environment type (Dev / QA / UAT)

These parameters are selected from the Jenkins project and pass to our framework. Runner Class (which has a main function inside it).

Configuration folder: All environment configurations are stored in this folder

DEV.properties

UAT.properties

QA.properties

Utility folder: It deals with reusable and common functions used across the framework. Common static functions like:

Fake data creator

Read Excel/csv file

Read properties file

Capture screenshot

Test Data component: Our framework can use test data from

Excel using Apache poi

CSV using Open csv

Database using JDBC

Fake data creation using JavaFaker

Report folder: Extent report generated reports are keep in this folder

Screenshots folder: All failed tests are stored in screenshot folder

Page classes component:

For UI automation e have used Page Classes

For all unique component of UI there is a specific page class present

Each page class consists of:

Private static final by Locators

A parameterized constructor

Functionality of that page

Each function of the page class returns some data or page object of the next UI page

**Listeners used in automation framework:**

**1) ITestListener:** It is used for generating reports, logs and taking screenshot of failed tests

**2) IRetryAnalyzer / IAnnotationTransformer:** used to rerun failed test dynamically.

@. Cucumber /Advantages

Cucumber is an open-source testing tool that supports Behaviour Driven Development (BDD). It allows the execution of feature documentation written in plain language.

**Adv:**

Easy to understand for non-programmers

Bridges the gap between technical and non-technical team members

Supports multiple programming languages

@. Gherkin keywords: Given, When, Then, And, But, Background, Scenario, Scenario Outline, Examples.

@. Feature file

A Feature file contains the high-level description of a software feature and its scenarios written using Gherkin language.

@. Scenario in Cucumber

A Scenario represents a single behaviour that needs to be tested.

@. Step Definition in Cucumber

Step Definitions are the scenario implementation steps written in a programming language like Java, Python, Ruby, etc.

**N:** A Scenario Outline is used to execute the same Scenario multiple times with different sets of test data.

@. Purpose of the Examples table in a Scenario Outline

The Examples table provides the different data sets for executing the Scenario Outline.

**N:** **Background** is used to define steps that are common to all scenarios in a feature file.

**N:** **Tags** are used to filter scenarios for execution. They can be applied to features, scenarios, or steps.

**N:** **StepDefinitions** are used to map steps to java code.

@. Set up Cucumber in a Java project

Add Cucumber dependencies in the build tool (Maven), create feature files, write step definitions, execute through Test Runner file.

@. Dependencies are needed to set up Cucumber with Maven

Cucumber-java, cucumber-Junit, and optionally cucumber-spring.

**N:** cucumber.options is used to specify various options like the path to feature files, step definitions, tags, etc.

@. To run Cucumber tests with JUnit

Annotate a test runner class with @RunWith(Cucumber.class) and specify the @CucumberOptions.

@. Cucumber Runner class

Cucumber Runner class is used to run the Cucumber tests. It config’s the path to the FeatureFiles and StepDefinitions. Runner class is creted using @CucumberOptions. Parameters used in Runner class: features, glue, tags, dryRun, monochrome, plugin.

@. Simple Scenario in Gherkin

Feature: Login

Scenario: Successful login with valid credentials

Given User is on login page

When User enters valid username and password

Then User should be redirected to the dashboard

@. Purpose of Given, When, Then in a Scenario

Given sets up the initial context, When describes an action, and Then describes an expected outcome.

@. To handle complex steps in Gherkin

Break down the steps into smaller, more manageable steps or use Scenario Outline for repetitive steps.

**N:** And and But are used to add more steps to Given, When, or Then.

**N:** Background is used for steps that are common to all scenarios in a feature file.

@. Sample Step Definition

@Given ("^User is on login page$")

public void user\_is\_on\_login\_page() {

// Code to navigate to login page

}

@. Pass parameters to a Step Definition

• Use placeholders in the step definition regex and method parameters.

@When("^User enters username (.+) and password (.+)$")

public void user\_enters\_credentials (String username, String password) {

// Code to enter username and password

}

**N:** @Before and @After hooks are used to execute code before and after each scenario.

**N:** To handle exceptions and log errors in Step Definitions we need to use try-catch blocks.

**N:** To share state b/n Step Definitions we have to use dependency injection frameworks like Spring or PicoContainer.

**N: Hooks** are blocks of code that can be executed before or after each scenario, similar to setup and teardown methods.

@. Sample Before Hook in Cucumber?

@Before

public void setUp() {

// Code to set up test data or environment

}

@. Sample After Hook in Cucumber

@After

public void tearDown() {

// Code to clean up test data or environment

}

@. H Hooks can be tagged in Cucumber

Yes, hooks can be tagged using @Before or @After with tag expressions.

@Before("@SmokeTest")

public void beforeSmokeTest() {

// Code to execute before scenarios tagged with @SmokeTest

}

**N:** @BeforeStep and @AfterStep hooks run before and after each step in a scenario.

@. Data Tables in Cucumber

Data Tables are used to pass a list of values or a table of values to a step definition.

When User enters the following details:

| username | password |

|user1 | pass1 |

| user2 | pass2 |

@. To handle Data Tables in Step Definitions

Using DataTable as a List of Maps



**N:** Data Tables are used for steps within a scenario and Examples Tables are used for Scenario Outlines to provide test data.

**N:** Data Tables can represent complex data structures like nested lists and maps.

**N:** To convert a Data Table to a List of custom objects we need to use the DataTable methods.

i.e., List<User> users = dataTable.asList(User.class);

**N:** To handle browser setup (initialize) and teardown (close the browser) in a Cucumber we need to use @Before and @After hooks.

@. To integrate Cucumber with Selenium?

Set up Selenium WebDriver in the step definitions to perform browser automation.

@. Capture screenshots in Cucumber

Use WebDriver's getScreenshotAs method in an @After hook.

**Ex:**

@After

public void takeScreenshot (Scenario scenario){

if (scenario.isFailed()) {

final byte[] screenshot = ((TakesScreenshot) driver).getScreenshotAs (OutputType.BYTES); scenario.attach(screenshot, "image/png", "screenshot");

}}

**N:** We can use explicit waits (WebDriverWait) to handle synchronization issues in Selenium with Cucumber.

**N:** We can use Page Object Model with Cucumber to create Page Object classes and use them in step definitions to separate test code from page-specific code and it promotes code readability, reusability.

@. Different types of reports in Cucumber: HTML reports, JSON reports, JUnit reports, and custom reports using plugins.

@.To generate an HTML report in Cucumber?

Use the html:target/cucumber-html-report plugin in @CucumberOptions.

@CucumberOptions(

plugin = {"html:target/cucumber-html-report"}

)

**N:** In Cucumber JSON report is used for further processing or integration with other tools like Jenkins or Allure.

**N:** Cucumber Reports: Generating reports using: pretty, html:target/htmlreport, json:target/jsonreport.

@. To integrate Cucumber reports with Jenkins

Install the Cucumber Reports plugin in Jenkins.

Configure the Jenkins job to execute the Cucumber tests and set the path to the JSON report.

Add a post-build action to publish the Cucumber report.

**N:** To configure multiple report formats in Cucumber use the plugin option in @CucumberOptions

**Ex:** @CucumberOptions(plugin = {"html:target/cucumber-html-report", "json:target/cucumber.json", "junit:target/cucumber.xml"}}

**N:** To provides rich HTML reports with detailed logs, screenshots, and various other features we need to use Cucumber Extent Reports plugin

**N:** To add custom logs to Cucumber reports we need to use the log method of the scenario object in step definitions.

**Ex:** scenario.log("Custom log message");

@. To generate an Allure report for Cucumber tests

Add Allure Cucumber dependencies to the project.

Configure the Allure report generation in the @CucumberOptions.

Run the tests and generate the report using Allure command-line tools.

@. Y it is important to keep feature files concise and focused

It improves readability, maintainability, and ensures that the scenarios are easy to understand for all stakeholders.

@. To ensure that our Gherkin scenarios are maintainable

Keep the steps DRY (Don't Repeat Yourself), use Background wisely, and avoid hardcoding data.

@. To handle common test setup code in Cucumber

Use @Before hooks for common setup code and Background in feature files for common steps.

@. Y should avoid using too many tags in Cucumber

It can lead to complex test execution configurations and make it harder to manage the scenarios.

To handle test data in Cucumber we need to use external data sources like CSV, Excel, or databases, and parameterize the scenarios to make them data-driven.

**N:** In Cucumber custom parameter types allow to define our own parameter types and transformations for steps.

**Ex:** ParameterType("currency", "USD|EUR|GBP", Currency.class, (String s) -> new Currency(s));

**N:** To reuse step definitions across different feature files place the step definitions in a common package that is accessible to all feature files.

@. Ways to share state between step definitions

Use shared variables in a context class, dependency injection, or use static variables (though not recommended).

@. To handle asynchronous operations in Cucumber

Use synchronization mechanisms like explicit waits or callbacks to ensure the steps wait for operations to complete.

**N:** @BeforeAll and @AfterAll hooks run once before all scenarios and once after all scenarios, respectively.

@. To integrate Cucumber with TestNG

Use cucumber-testng library and configure the Cucumber runner with TestNG annotations.

@. To use Cucumber with REST Assured for API testing

Combine Cucumber for BDD and REST Assured for API testing in the step definitions.

@. To manage configuration properties in a Cucumber project?

Use a configuration file (like config.properties) and load it in step definitions or hooks.

**N: @Transform** is used to apply custom transformations to step parameters.

@. W do if a step definition is not found

Ensure the step definition regex matches the step in the feature file, and check the glue path in @CucumberOptions.

**N:** To debug Cucumber step definitions use breakpts and a debugger in IDE, or add logging stmts to the step definitions.

**N: Undefined step error** occurs when a step definition is not found. Ensure the step text matches the regex in the step definition.

@. To handle flaky tests in Cucumber

Identify the cause (e.g., timing issues, environment issues) and use retries, better synchronization, or refactor tests.

**N:** To skip a Cucumber scenario use **tags** and configure the Cucumber runner to exclude the tagged scenarios.

**N:** BDD is an agile software development process that encourages collaboration among developers, QA, and non-technical stakeholders to create shared understanding of requirements. Cucumber support BDD by allows writing tests in a natural language format that non-technical stakeholders can understand, fostering collaboration.

@. Limitations of Cucumber

Can be verbose for complex scenarios, requires collaboration to write meaningful feature files, and might not be suitable for all types of testing.

**N:** To handle large Cucumber projects: organize feature files and step definitions into logical packages, use tags to manage test execution, and keep feature files concise.

**N:** Parallel execution allows running multiple cucumber scenarios or features simultaneously to reduce test execution time. It can achieve using plugins like cucumber- parallel, or frameworks like TestNG with Cucumber.

@. To handle test data setup and teardown in a CI/CD pipeline

Use @Before and @After hooks for test data setup and teardown. Manage test data using scripts or database tools in the pipeline.

**N:** **Cucumber Expressions** are a simpler alternative to regular expressions for matching steps in Cucumber. To write Cucumber Expression: Use curly braces to denote parameters (**Ex:** When User enters {string} and {string})

**N:** **Optional text parameters** allow matching text that may or may not be present in a step. (**Ex:** When User enters {string} and {string} in the {string?} field)

@. Use regular expressions in Cucumber step definitions

Use @Given, @When, and @Then annotations with a regex pattern.

**Ex:**

@When("^User enters (.+) and (.+)$")

public void user\_enters (String username, String password) {

// Code to enter credentials

}

**N:** Cucumber Expressions are simpler and more readable, designed for business-readable scenarios, while regular expressions are more powerful but complex.

**N:** @Transform annotation is used to apply custom transformations to step parameters.

**Ex:** @Transform(DateConverter.class) Date date

@. To create a custom transformer in Cucumber?

Implement the Transformer interface and override the transform method.

**Ex:** public class DateConverter extends Transformer<Date> {

@Override

public Date transform(String value) {

return new SimpleDateFormat("yyyy-MM-dd").parse(value);

}

@. To handle custom data transformations for complex types in Cucumber

Create a custom transformer class that implements the ParameterType interface.

public class ComplexTypeTransformer extends Transformer<ComplexType> {

@Override

public ComplexType transform(String value) {

// Custom transformation logic

return new ComplexType(value);

}}

**N:** @ParameterType annotation is used to define custom parameter types that can be reused in step definitions.

@ParameterType("customType")

public CustomType customType(String value) {

return new CustomType(value);

}

**N:** To use custom parameter type in a step definition define the custom parameter type and use it in the step definition.

@Given("^a custom type parameter (customType)$")

public void a\_custom\_type\_parameter(CustomType customType) {

// Use the customType parameter

}

@. To execute Cucumber tests in parallel in a CI/CD pipeline

Use tools or plugins that support parallel execution, like Maven Surefire or TestNG with Cucumber.

Configure the CI/CD pipeline to distribute tests across multiple agents or nodes.

@. To handle environment-specific configurations in Cucumber tests

Use environment variables or configuration files to manage different environments.

Load environment-specific properties in the @Before hook.

@. Benefit of integrating Cucumber with RestAssured for API testing

Allows for writing BDD-style tests for REST APIs. And simplifies testing by combining Cucumber's readability with RestAssured's powerful API testing capabilities.

@. To improve the performance of Cucumber tests?

- Optimize the setup and teardown processes.

- Use parallel execution where possible.

- Refactor step definitions to reduce redundancy and improve efficiency.

**@. Manual and Automation testing**

Automation testing is faster, more reliable, and can execute repetitive test cases. Where manual testing is better for exploratory and UI testing.

@. Automation test pyramid

Automation test pyramid consists of three levels

* Unit tests (bottom, most tests)
* Service/API tests (middle)
* UI tests (top, least tests)

**@. Role of Git in Automation testing:** Git helps version control test scripts and integrate with CI/CD tools.

**N:** To handle flaky tests by retry mechanism and use explicit waits instead of sleep.

**@. Working Under Pressure to Meet a Deadline**

Prioritize Critical Test Cases (Risk-Based Approach)

I identified high-risk areas that could impact core functionalities (e.g., checkout, payment processing, login).

Focused on running critical smoke and regression tests first.

Parallel execution in TestNG/Grid helped reduce test execution time

**@. To ensure reliability and maintainability of our automated tests**

To Ensure / improve test reliability: Use Stable Test Data, Avoid Hardcoded Waits, Ensure Tests are Independent, Avoid Flaky Tests.

To Ensure Maintainability: Follow the POM design pattern, Write Readable & Descriptive Tests, Use Proper Assertions.

**@. To measure the effectiveness of our test automation**

Test Coverage (%), Test Execution Time, Defect Detection Rate (%), ROI of Test Automation, Continuous Monitoring & Improvement (CI/CD Integration, Regular Test Audits)

**@. To Prioritize Test Cases in Testing Strategy**

Prioritizing Test Cases ensures that the most critical functionalities are tested first so that reducing risk and improving the efficiency. Criteria for Prioritizing Test Cases Based on: Business Impact (H, M, L), Risk-Based Testing (H, M, L Risk), Frequency of features use (Core (Ex: Login) vs. Rarely Used Features), Complexity & Dependency.

**Overall:** High-Priority Tests (Run First), Medium-Priority Tests (Run Next) and Low-Priority Tests (Run Later).

**@. To prioritize test cases when time is limited**

1. Identify Critical Areas: Focus on functionalities that are critical to the appln, such as login, payments, or data integrity.

2. Risk-Based Testing: Prioritize tests in areas prone to defects or those that have undergone recent changes.

3. Business Impact: Execute test cases that cover features with the highest business value or end-user impact.

4. Frequent Use Scenarios: Test features or workflows that are used frequently by users.

5. Regression Testing: Ensure that previously working features are not broken by recent changes.

**N:** By fetching the title or aria-label attribute we can validate the **tool-tip**

**N: Selenium standalone server** runs locally and **Selenium Grid** distribute tests across multiple machines.

**N:** ‘Test data management in Automation’ is nothing but a process of creating, maintaining and managing test data for automation scripts.

**@. Some best practices for Selenium Automation**

Use POM, handle waits efficiently, write reusable methods, and maintain the clear folder structure.

**@. AutoIT and Robot class while uploading files**

While using Robot class we have to enter file path as individual character with keypress() and keyRelease() methods. Here we can specify unique characters and by using Up-Down arrows we can select the desired file.

While using AutoIT we can directly specify the full file path (desired file).

**@. Generate reports in selenium**

To generate reports in selenium we have to use any one of the reporting frameworks like ExtentReports, TestNG reports and Allure reports to generate detailed test reports, including screenshots on failure, test status, Logs and time stamp.

**@. Handle file upload in selenium / When sendKeys() wont work**

By using sendKeys() to send the file path directly to the input element for uploading.

**Ex:** driver.findElement(By.id(“uploadFile”)).sendKeys(“F:\\file.txt”);

If the upload button opens a windows file dialog box. Selenium cannot control it (outside browser scope). To handle this testers use eternal tools: Robot class, AutoIT, Sikuli.

**@. Use of ChromeOptions** Ref: WD\_Examples\EX70\_UseOfChromeOptions

ChromeOptions class is used to customize and configure a ChromeDriver session. It can be used to perform various operations like opening chrome in Headless browser mode, set browser size, open browser in maximized mode, etc.

**@. Y WebDriver called as an interface**

* Interface define methods that other classes should implement, Interface mechanism achieves abstraction, Interface has abstract methods without body.
* WebDriver is public interface because it defines a certain set of methods.
* The implementation of WebDriver is provided by the browser-specific classes (Implementation classes are CH, FF driver, etc)

WebDriver interface providing methods: get(), close(), quit(), getWindowhandle(), getWindowHandles(), getTitle().

**@. To verify whether a webpage contains 404 error code during page launch**

Using getPgeSource() we can verify whether the page contains any 404 error codes during launch

Assert.assertTrue(driver.getPageSource().contains(“404”))

**@. When go for Absolute XPath strategy / Handling dynamic WebElements**

* Suppose when there are no direct ways to locate WebElements we can go for Absolute XPath strategy.
* From root node to desired WebElement we can do parent-child traversal and locate WebElements.

**@. H to handle dynamic dropdowns in selenium**

* First use wait techniques to make sure dropdown is loaded.
* Once dropdown is visible, locate the dropdown using any locator like id, XPath…
* Select values from dd using methods like selectByValue(), selectByVisisbleText() or selectByIndex().

Dynamic dropdown: allows a second dropdown field to display values based on the selection made in the first dd field.

**@. If we have multiple @Test annotations in a class file which test will be running first**

In this scenario, Test runs in alphabetical order (a to z) (i.e. testName starts with ‘a’ will execute 1st, ‘b’ will exe next)

**@. H will prioritize tests using TestNG**

We can prioritize TCs by providing priority=0, 1, 2… here the test script given priority ‘0’ will executed first and ‘1’ will be executed next and goes on.

**@. Use of testing.xml file**

* To run multiple tests in a single execution we use testing.xml
* It helps us to include and exclude the test methods and groups.
* It helps in the execution of parallel test execution.
* It helps us to add listener tags.

**@. Grouping in TestNG**

* Groups are the collection of multiple Test Case methods combined into one single unit.
* Suppose if we want to execute only smoke Test Cases we can execute smoke suite by providing group name as ‘smoke’ to all the smoke Test Cases. Ex: @Test(groups={“group1”})

**@. To create dependencies in TestNG**

Suppose if we want to execute particular method before executing another method we can use ‘dependsonMethods’ parameter. Add ‘dependsonMethods’ parameter to the @Test annotation.

**@. Parameterization in TestNG**

Parameterization runs a test method multiple times with different values. Another name for this is process is data-driven testing in TestNG.

We can achieve parameterization in TestNG in two ways: Those are through @Parameters annotation with testng.xml file and through ‘@DataProvider’ method for multiple sets of data.

**@. Type of automation frameworks we use**

Data-driven, keyword driven --- used for more complex applications or BDD framework --- used for small scale applns.

**@. Components in framework**

BaseClass, resources package, Object repository, Tests Package, Reporting, Dockerfile, yaml file for CI/CD execution, pom.xml, testing.xml.

**@. Per sprint, H many Test Cases u automate**

2-3 high complex TCs or 5-6 medium complexity TCs or 8-10 low complexity TCs

**@. Explain ur automation process**

Review requirements -> Create test scripts -> send it for review -> After approval create PR and merge it to master branch.

**N:** WebDriverIO is an open-source testing automation framework written in JavaSCript on NodeJS. It is particularly useful for testing web and native mobile applications for iOS-enabled devices.

**@. While using click command, can we use screen coordinate**

Yes, in selenium clickAt command is used to click on a specific part of an element. The element locator and x, y coordinates are accepted as arguments for the clickAt(locator, cordString) (cordString spec the x,y position (i.e. - 10,20). **Ex:** clickAt("//li[@id='tab-panel']/a[2]/em/span","0,0")

**@. To confirm the specific position of a WebElement in the page**

verifyElementPositionLeft/Right/Top/Bottom etc can be used to identify the position of the element with respect to the web page. The commands use pixel comparison to verify the positions.

**@. WebDriver.findElement vs WebElement.findElement**

WebDriver.findElement(): This finds the element from the entire page using the given selector.

WebElement.findElement(): First, it generates the WebElement. Then, the child elements of the given element are searched based on the given selector.

**@. Design patterns in the Selenium framework** **--** POM Pattern, Factory Pattern

**N:** Page Factory is a class in Selenium WebDriver it makes easier to create page objects (keeping object repository) or page classes separate from test classes. Here we use ‘@FindBy’ annotation to find WebElements and ‘initElements’ to initialize WebElements.

**@. To handle Windows-based alerts or pop-ups in Selenium**

Selenium only supports web applns and does not provide a way to automate Windows-based applns. However, we can use the Java-based Robot class utility to simulate keyboard and mouse actions. The KeyPress and KeyRelease methods simulate the user pressing and releasing a specific key on the keyboard.

**@. To use regular expressions in Selenium**

Regular expressions in Selenium can be used within various contexts, including locators, assertions, and data validation. For ex, we can use **regex** in XPath or CSS selectors to find elements with specific attributes or text that follow a particular pattern.

**@. We need to retrieve the message in an alert box, without using the getText() method. H will handle this situation**

storeAlert() command can be used to retrieve the message from the alert pop-up. The value is stored in a variable that can be returned using a System.out.println statement.

**@. You are given a set of automated scripts that are highly confidential and must be run on the client’s remote server only. You are asked to modify the scripts based on privacy needs, what will you do**

I understand that this scenario requires the tests to be run on a separate machine. Instead of using the WebDriver library and commands, I will use the RemoteWebDriver command to execute these tests. To run a remote WebDriver client, we need to configure it. This can be done by pointing the URL to the address of the remote server to execute the tests. Capabilities can be added to configure the tests further. Here is a sample code:

**@. U have been asked to perf aut testing on a website that loads at a slow pace. You are able to execute the scenario manually without any impediments other than the speed issue. But while automation, NoSuchEleExcp is throwing.**

driver.get commands are perfect for a website that loads slowly, as it waits for the page to get loaded fully and then proceeds to the next steps. However, while using driver.navigateto() method, the driver does not wait for the page to be loaded completely, and many components are loaded yet, which throws the NoSuchElement exception.

@. To enable a disabled textbox

We can enable a disabled HTML element, be it a button, input field, text area, etc, by changing the attributes using JsE **To disable:** document.getElementById("id\_name").setAttribute("disabled", true);

**To enable:** document.getElementById('id\_name').removeAttribute('disabled');

**Ex:** String enable ="doc.getElementsByName("name').removeAttribute('disabled');";

javascript.executeScript(enable);

**@. John is not sure if the warning text label is displayed on the web page of a blog. H will u help him using Selenium**

We can use the method isDisplayed() to confirm the presence of the element on the screen.

**Ex:** Boolean buttonPresence = driver.findElement(by.id("sample").isDisplayed();

**@. Ur team needs to check if a warning msg is displayed when the password entered by the user is moderately secure**

In this scenario we cannot use Hard Assert commands as it halts the further steps of the execution, in case of an assertion failure. Instead, we can use the verify command in this case. Soft Assert commands, check the conditions if they are true or false, and also all the test phases get executed.

**@. XPath locator to identify paragraph ele’s that are the immediate child of a div ele, or the descendent of a div ele**

An immediate child in XPath is indicated using “/”. To find paragraph elements that are descendants to any div element. We can use “//” in the XPath.

**@. To manage session persistence between multiple test cases in Selenium, especially when running tests in parallel**

It will vary depending on testing frameworks and tools we use. While running tests in parallel saving and reusing session cookies to maintain a logged-in state across multiple Test Cases is important for that retrieving cookies with getCookies(), storing them, and reapplying them in subsequent tests with addCookie().

**@. To simulate keyboard events using Selenium WebDriver**

To simulate keyboard events, we must need to use Actions class. Here use the sendKeys() on a WebElement to input text or trigger keyboard shortcuts. It is useful in scenarios like filling in forms or interacting with KB-based functionality.

**@. To deal with dynamic wait times in Selenium tests**

Dynamic waits are essential for handling unpredictable delays. We must use explicit waits with ExpectedConditions to wait for specific conditions before proceeding. It ensures that the test waits only as long as necessary and avoids unnecessarily long wait times.

**@. To execute JavaScript code using Selenium WebDriver**

To execute JavaScript code, use the executeScript() method provided by WebDriver. This method allows to run custom JS code within the context of the current page. It's useful for scenarios where direct manipulation of the DOM is req.

**@. Chromium DevTools Protocol Integration in Selenium 4, and how does it enhance web testing capabilities**

The Chromium DevTools Protocol Integration is a significant enhancement introduced in Selenium 4. Testers to interact with the Chrome DevTools Protocol directly from their Selenium scripts. This integration enables advanced debugging, profiling, and analysis of web pages during automated testing.

**@. Imagine that we are testing an e-commerce application using Selenium. Due to a pending deployment, the team has decided to not test the modules related to payment gateways in the application. H will efficiently alter the Selenium automation testing scripts in this case**

Since we would need to test the module again, skipping the particular test method or block of code would be the best option for now. To skip a test method, the ‘enabled’ parameter in the @Test annotations can be set to false. (The default value is true). **Ex:** @Test(enabled = false)

**@. Ur test scripts have passed, but u want to have a look at the appln’s UI during a particular step. H can this be done**

We can always take a screenshot of the UAT by using the TakeScreenshot function. To save the screenshots, use the getScreenshotAs() method.

**@. To identify dynamic images in Selenium**

We can identify dynamic elements in Selenium using the following methods:

* By using Relative XPath with “contains” or “starts-with” methods. This method is highly preferred if the changes in the values of attributes like ID or class of the element change a similar pattern. Count of messages in the mailbox, user welcome messages, etc are a few examples. **Ex:** //input[starts-with(@id, 'submit\_')]
* By identifying the elements by their index. For example, there may be two register buttons with the same locator value with an id starting with “login”. In this case, we can use the findElements method and locate the element using the index. **Ex:** driver.findElements(By.xpath(//\*[contains(@id, 'login')).get(1).click();
* By adding multiple attributes to locate the WE. **Ex:** //button[starts-with(@id, 'login') and contains(@class, 'success')]

**@. Which design patterns used in framework** ---- POM and Factory pattern.

**@. Factory design pattern in java**

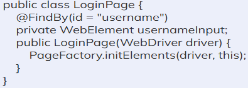
Creates objects without exposing creation logic. Useful for browser and driver initialization i.e. Dynamically create browser drivers based on the input parameters.

**@. PageFactory** Ref: Java\_WD\_Ex\PageFactoryEx

Page Factory is a class provided by Selenium WebDriver. It is an enhancement over POM design pattern. This is used to initialize web elements declared in Page Objects. It uses @FindBy annotation with initElements() i.e., **PageFactory.initElements()** to automatically initialize the elements with lazy loading/ initialization (elements are created only when used). It provides an optimized way to execute the POM.

Here By() method is used to perform POM implementation. PageFactory is used to initialize the elements of the Page Object. @FindBy method is used to find WEs and the initElements method is used to initialize WEs from the PageFactory class. Using the Page Factory class, we can reuse our Selenium automation testing scripts and it req less maintenance.

Adv: Initialization, Improved Readability, Maintainability and Reusability.

**N:** PF will use lazy initialization technique through java reflection and dynamic proxies. When we declare elements with @FindBy and initialize the using PageFactory.initElements(), selenium doesn’t immediately find those elements instead it creates a proxy object for each WebElement.

**@. To create reusable components in selenium** ---- By using POM, utility classes, and BaseTest setup.

**@. To lunch different browsers in selenium WebDriver**

A browser can be launched by creating an ‘driver’ instance of a particular browser.

**@. There is a scenario whenever “Assert.assertEquals()” function fails automatically it has to take screenshot. H can achieve this?** Ref: Java\_WD\_Ex\TestNG\TakesScreenshot\_wh\_AssertFunction\_Fails\_Automatically

**@. In a web page, h will ensure that the page has been loaded completely** Ref: WD\_Examples\EX71\_Ensure\_PageLoadedCompletely

We can execute the JavaScript code having document.readyState parameter to check whether the ‘complete’ status of the page is returned or not.

**@. if there are failures in our suite execution and wt is ur approach**

While executing the automation scripts, TCs may fail for several reasons. To optimize our next runs, we need to re-run only failed TCs using TestNG in two methods those are:

**M1:** By using testng-failed.xml file in test-output folder. **M2:** By implementing TestNG IRetryAnalyzer.

**@. Difference we observer when working with different browsers? (except set up or configuration level)**

Below are the few differences we can see across working with diff browsers: Font size mismatch in different browsers, JavaScript implementation can be different, CSS, HTML validation difference can be there, Some browser still not supporting HTML5, Page alignment and div size, Image orientation, Browser incompatibility with OS. Etc.

@. To reduce page load time if any particular data taking some time

One of the best ways of making a script faster is by using only explicit waits as HTML elements are accessed as soon as they become available.

**@. Our script performing some actions on a list elements, in between if page get refreshed, h to handle the situation**

This scenario is related to a page which has an ajax poll and automatically refreshes when some condition is reached on the server. Here we have to wait for the Ajax request to finish for that we will to use different wait commands. Then we continue after the ajax request is done.

**@. If we execute the scenarios in multiple times, will the reports override? If they override, H will take backup of previous test report**

We can override reports and also, we can create our file name with the current timestamp. This way, it will be easy to have a unique name for our report file.

**@. Wt will do when we have more number of lines of code having repeated code**

We can avoid the duplicate code by using custom methods, Inheritance etc.

**@. In Selenium proj, we use hierarchy like interface, followed by abstract class, followed by a class. Can’t we directly use interface followed by a class?**

Yes, we can do that; When we implementing the interface through class we can implements all the mtds from interface.

**@. Problem with Thread.Sleep in code**

* It is a Static wait: Thread.sleep() increases the execution time in cases where elements are loaded within the time (If given a wait of 5000 milli.sec (5 sec) and an element just take just 1-2 seconds to load, script will still wait for another 3 seconds which is bad)
* When we mention the wait time using thread.sleep(5000) in advance, there is no guarantee that the element will be displayed in that specified wait time, there may takes more than 5 secto load and again the script would fail.
* If we want to wait for two web elements, we need to write Thread.sleep() twice just before locating web elements.

**@. A scenario that 5 TCs are there I need to execute first and last 3 (means 2 should not be executed)? H u make changes in testing.xml file?**

Two more ways are given below:

First mtd is to setup the condition enabled or disabled in Test ann: @Test(enabled=true)

Second method is to setup the group details at Test annotation: @Test(group = {functionalTest} )

**@. We have 2 interfaces and both have print methods, in my class I have implemented the print method, H u will get to know that I have implemented the first interface and h u will use it.. if u want to use it?** Ref: OOPS\_Abstraction\Interface\_IQ

Since the method signature is same, and both are merged in the class, we can cast the object to the desired interface and call the method.

**@. If element is loaded by taking much time, how to handle this situation in selenium?** ---- Apply fluent wait.

**@. Want to store 10000 elements in a collection which collection to use?**

|  |  |
| --- | --- |
| **Requirement** | **Best Collection** |
| Fast access by index | ArrayList (**Ex:** List<String> list = new ArrayList<>();) |
| Unique elements (no duplicates) | HashSet (**Ex:** Set<String> set = new HashSet<>();) |
| Key-value storage | HashMap (**Ex:** Map(Var)<Integer, String> map = new HashMap<>();) |
| Sorted order | TreeSet / TreeMap |
| Frequent insertion/removal | LinkedList |

**@. First time TC passed second time aborted. What could be the reason?**

* Timing Issues / Asynchronous Code
* Residual Data from Previous Run
* Session / Login Expiry
* Browser/Driver Not Restarted Properly (Selenium)

**@. There are 300 TCs, I want to execute TCs in some custom order, H to change the order of execution without doing changes in testng.XML and in code (.class files). If we can do, tell the logic and if we cannot do, justify with reason?**

NO - not directly at runtime without changing either testng.xml or code.

TestNG determines execution order at runtime based on:

* testng.xml (suite, test, classes, methods sequence).
* @Test(priority=...) or method dependencies (@Test(dependsOnMethods=...)) in code.

**@. Is default polling frequency 250 milli.sec applies to implicit wait or explicit wait or for both types of wait**

The default polling frequency for Selenium’s implicit and explicit wait is every 500 milli.sec, not 250 milli.sec.

Selenium will poll the DOM every 500 milli.sec until the element is found or the timeout expires.

However, when we use Fluent Wait, we can customize the polling interval (e.g., set it to 250 milli.sec if we want).

@. Ur are assigned to automate the login functionality of a web application. H to approach this task

* First, I would identify the elements involved in the login process such as username field, pwd field, and login button.
* Then, I would choose an appropriate automation tool such as Selenium WebDriver.
* Next, I would write Test Scripts using the chosen tool to automate the login process.
* I would also incorporate data-driven testing to test different combinations of valid and invalid credentials.
* Finally, I would run the automated tests regularly as part of the CI process to ensure the login func’y remains intact.

@. U are testing an e-commerce website where users can add products to their cart. H to automate the testing of the add-to-cart functionality

* Firstly, I would identify the key steps/elements involved in adding products to the cart, such as product listings, add-to-cart buttons, and the cart icon.
* Using automation tool, I would write test scripts to automate the process of adding products to the cart.
* I would also include verification steps to ensure that the correct product is added with the desired quantity.
* Additionally, I would validate the cart contents after adding products to ensure they reflect accurately.
* Finally, I would parameterize the test data to cover various scenarios like adding multiple products, adding out-of-stock products, etc.

@. U are tasked with automating the testing of a search functionality on a website. H to approach this?

* I would start by identifying the search input field and the search button on the website.
  + Using automation tools, I would write test scripts to automate the search process.
  + I would include test data with various search queries, including valid and invalid ones, to cover different scenarios.
  + Additionally, I would verify that the search results are displayed correctly and relevant to the search query entered.
  + I would also test for advanced search options if available, such as filtering or sorting search results.
  + Lastly, I would ensure that the search functionality works seamlessly across different browsers and devices.

@. U need to automate the testing of a checkout process on an e-commerce website. H to approach this task

* I would identify the key steps involved in the checkout process, such as adding items to the cart, entering shipping and billing information, selecting payment methods, and placing the order.
* Using automation tools, I would write test scripts to automate each step of the checkout process.
* I would include validation checks at each step to ensure that the correct information is entered and that the process flows smoothly.
* Along with, I would test for error handling scenarios, such as entering invalid payment info or encountering server errors during checkout.
* I would also validate the order confirmation page to ensure that the order is processed successfully.
* Finally, I would run the automated tests across different browsers and devices to ensure compatibility.

@. Ur testing a web application that involves interactions with dynamic elements such as dropdown menus, pop-up windows, or tooltips. H will automate testing for such elements

* + I would use Selenium WebDriver along with explicit waits to handle dynamic elements effectively.
  + For dd menus, I would use the Select class in Selenium to interact with dd options and verify selected values.
  + For pop-up windows, I would use WD's switchTo() method to handle alerts, modals, or new browser windows.
  + I would write test scripts to handle various scenarios involving dynamic elements, such as selecting options from dropdowns, dismissing pop-up windows, or interacting with tooltips.
  + Additionally, I would include validation steps to ensure that the expected actions are performed based on interactions with dynamic elements.

@. Ur testing a web application that includes functionality for user profile management. H will automate testing for profile updates and validations

* + Firstly, I would identify all the fields and actions associated with profile management, such as updating personal information, changing passwords, and uploading profile pictures.
  + Using automation tool, I would write test scripts to automate the process of updating profile information.
  + I would include TCs to validate the handling of both valid and invalid inputs, such as entering special characters in name fields or incorrect formats for email addresses.
  + Additionally, I would verify that changes made to the profile is reflected correctly on the UI and persisted in the DB.
  + I would also include tests to verify pwd strength requirements and the functionality for resetting forgotten pwds.
  + Finally, I would ensure that any profile-related notifications or confirmation messages are displayed appropriately.

@. U r testing a web application that involves interactions with third-party APIs for payment processing. H to automate these API integrations

* I would use tools like Postman or RestAssured to automate testing for the third- party API integrations.
* Firstly, I would identify the endpoints and methods exposed by the third-party APIs for payment processing.
  + Using the chosen tool, I would write test scripts to send HTTP requests to these endpoints with various payment scenarios, such as successful payments, failed payments, and refunds.
  + I would include assertions to verify the status codes, response payloads, and transaction statuses returned by the APIs for different payment scenarios.
  + Additionally, I would test for error handling and edge cases, such as network timeouts, server errors, or invalid payment data.
  + I would also include tests to verify the security measures implemented, such as encryption of sensitive payment information and compliance with PCI-DSS standards.

@. U r testing a web application that includes real-time chat functionality. H to automate testing for this chat feature

* + Firstly, I would identify the elements and actions associated with the chat functionality, such as sending messages, receiving messages, and displaying online/offline status.
  + Using Selenium WebDriver, I would write test scripts to automate the process of interacting with the chat interface
  + I would include validation steps to ensure that messages sent by the user are displayed correctly in the chat window and received by other participants in real-time.
  + Additionally, I would test for scenarios such as concurrent chat sessions, handling of multimedia messages, and notifications for new messages.
* I would also include tests to verify the handling of edge cases such as network interruptions or server timeouts during chat sessions.

@. U r testing a web application that includes a subscription-based service with different membership tiers. H to automate testing for subscription management?

* Firstly, I would identify the elements and actions associated with subscription management, such as selecting membership tiers, upgrading/downgrading subscriptions, and cancelling subscriptions.
* Using aut tool to write test scripts to automate the process of managing subscriptions with diff user accounts
* I would include validation steps to ensure that users can successfully upgrade or downgrade their subscriptions and that the changes are reflected accurately in their account details.
  + Additionally, I would test for scenarios such as applying discount codes or promotional offers during subscription upgrades.
* I would also include tests to verify the handling of subscription cancellations, including any confirmation prompts or cancellation fees.
* Finally, I would ensure that users receive appropriate notifications or emails regarding subscription changes.

@. Ur testing a web application that includes functionality for user-generated content, such as comments, reviews, or ratings. H to automate testing for user-generated content moderation

* + I would identify the elements and actions associated with user-generated content moderation, such as submitting comments, deleting comments, or flagging inappropriate content.
  + Using aut tools write test scripts to automate the process of submitting and moderating user-generated content.
  + I would include validation steps to ensure that users can submit comments or reviews successfully and that the content appears correctly on the website.
  + Additionally, I would test for scenarios such as flagging inappropriate content and verifying that flagged content is removed or hidden from public view.
  + I would also include tests to verify the functionality for moderators or administrators to review and approve/reject flagged content.
  + Finally, I would ensure that users receive appropriate notifications or feedback regarding the status of their submitted content (e.g., approval pending, content deleted).

@. Ur testing a web application that includes functionality for scheduling appointments or bookings. H to automate testing for appointment scheduling and management

* + I would identify the elements and actions associated with appointment scheduling and management, such as selecting appointment slots, entering customer details, and confirming bookings.
  + Using Selenium WebDriver or similar tools, I would write test scripts to automate the process of scheduling appointments with different user accounts.
  + I would include validation steps to ensure that users can successfully schedule appointments for available slots and that the bookings are reflected accurately in the system.
  + Additionally, I would test for scenarios such as rescheduling or cancelling appointments and verifying that the system updates availability accordingly.
  + I would also include tests to verify the handling of overlapping appointments, booking conflicts, or appointment reminders.
  + Finally, I would ensure that users receive appropriate notifications or confirmations regarding their scheduled appointments.

@. Ur testing a mobile appln that includes push notification functionality. H to automate testing for push notifications

• Utilize mobile automation frameworks like Appium to automate testing for push notifications.

• Identify the triggers that generate push notifications within the application, such as new messages, reminders, or updates.

• Write test scripts to simulate the actions that trigger push notif’ns and verify that notifi’ns are received by the device.

• Include validation steps to ensure that the content of the push notification is accurate and matches the expected behaviour.

• Test scenarios such as receiving notifications while the application is in the background or when the device is locked.

• Verify that users can interact with push notifications, such as opening the associated message or dismissing the notification.

• Test the handling of mul notifications and any grouping or stacking behaviour implemented by the operating system.

@. U need to automate the testing of a login API that requires authentication using OAuth 2.0. H to approach this task

• First, I would familiarize myself with the OAuth 2.0 authentication flow, including obtaining access tokens and making authenticated requests.

• Then, I would design TCs to cover the authentication process, including scenarios like obtaining valid access tokens, refreshing expired tokens, and handling invalid or revoked tokens.

• Additionally, I would include TCs to validate the behaviour of protected endpoints, ensuring that they reject unauthenticated or unauthorized requests.

• Using a testing framework like RestAssured or Postman combined with a programming language like Java or JavaScript, I would automate these TCs to simulate the OAuth 2.0 authentication flow and verify the expected outcomes.

@. Ur have to automate the testing of a web appl's localization & internationalization features. H to approach this task

* + Firstly, I would identify the languages and locales supported by the web appln and understand the localization and internationalization mechanisms implemented, such as language selection dropdowns and locale-specific content.
  + Then, I would design TCs to cover various scenarios such as switching between different languages and locales, validating translated content, and testing date/time formats, currency symbols, and number formats.
  + Additionally, I would include TCs to verify the application's behaviour when handling right-to-left (RTL) languages and Unicode characters.
  + Using automation tool, I would automate these TCs to simulate user interactions with the language and locale settings of the web application and ensure its proper localization and internationalization support.

@. U need to automate the testing of a web application's error handling & logging mechanisms. H to approach this task

* Firstly, I would identify the types of errors and exceptions that the web application may encounter, such as 404 Not Found, 500 Internal Server Error, or custom application-specific errors.
* Then, I would design TCs to cover various error scenarios, including triggering errors through invalid input, API failures, and database connection issues.
* Additionally, I would include TCs to validate the logging of errors and exceptions, ensuring that relevant information is captured in the application logs.
* Using automation tool, I would automate these TCs to simulate error conditions and verify the proper logging and handling of errors by the web application.

@. Ur have to automating the web applications data encryption and decryption mechanisms. H to approach this task

* Firstly, I would understand the encryption algorithms and key management practices used by the web application, such as AES encryption with key rotation or RSA encryption with public-private key pairs.
* Then, I would design TCs to cover various encryption and decryption scenarios, including encrypting sensitive data before storage, decrypting data for display or processing, and verifying data integrity after encryption and decryption.
* Additionally, I would include TCs to validate error handling & excp handling in case of encryp’n or decryp’n failure.
* Using automation tools, I would automate these TCs to simulate encryption and decryption workflows and verify the security and reliability of the web application's encryption mechanisms.

**Concurrent database transactions:** Multiple users accessing and updating the same data simultaneously.

@. Ur testing a mobile appln that includes functionality for image recognition. H to automate testing for image recog’on

• Utilize mobile automation frameworks like Appium to automate testing for image recognition within the mobile appln.

• Identify the elements and actions involved in image recognition, such as

capturing images, processing image data, and recognizing objects or patterns within images.

• Write test scripts to simulate scenarios such as capturing images using the device's camera, submitting images for recognition, and verifying that recognized objects or patterns are identified correctly.

• Include validation steps to ensure that image recognition algorithms produce accurate results, and that associated actions or workflows are triggered based on the recognized objects or patterns.

• Test scenarios such as handling scenarios where image recognition fails due to poor image quality or environmental conditions, verifying that recognized objects are annotated or labelled correctly, and testing the application's behaviour when multiple objects are recognized simultaneously.

@. Ur testing a web application that includes functionality for file versioning and revision history. H to automate testing for file versioning and revision history?

• Identify the elements and actions involved in file versioning and revision history, such as file upload/download interfaces, version control systems, revision logs, and comparison tools.

• Utilize automation tool to automate interactions with the file versioning and revision history interface.

• Write test scripts to simulate scenarios such as uploading files with different versions, downloading previous versions of files, viewing revision history logs, and comparing changes between file versions.

• Include validation steps to ensure that file versioning processes are accurate and reliable, that revision history logs are comprehensive and accessible, and that users receive appropriate permissions and notifications when accessing previous file versions.

• Test scenarios such as handling scenarios where file versioning conflicts occur, verifying that file metadata (e.g., timestamps, authorship) is recorded correctly, and testing the application's behaviour when users attempt to revert or restore previous file versions.

@. Ways to handle dropdowns

1. Using Select class (for standard <select> tags)

2. Using Click and Send Keys

**Ex:** WebElement dropdown = driver.findElement(By.id("dropdownInput"));

dropdown.click();

dropdown.sendKeys("Option Text");

dropdown.sendKeys(Keys.ENTER);

3. Using Click and Custom XPath (Non-<select> Dropdowns)

4. Using Actions Class (for hover-based dropdowns)

5. Using JavaScript Executor (when click fails with normal Selenium)

6. Using Robot Class (for OS-level dropdowns)

7. Using Select2 or Custom JS Plugin Handlers (using Select2, Chosen, or other JavaScript libraries)

**Ex:**

driver.findElement(By.xpath("//span[@class='select2-selection']")).click(); // Click to open dropdown

driver.findElement(By.xpath("//li[contains(text(),'Option Text')]")).click(); // Select item

@. Scenario Outline and Data Tables in cucumber

In Cucumber, both Scenario Outline and Data Tables are used to handle test data — but they serve different purposes. Here’s a clear comparison:

|  |  |
| --- | --- |
| **ScenarioOutline** | **Data Tables** |
| Used for running the same scenario mule times with diff sets of data (testing same flow with mul input combi’ns) | Used to pass a set of data (complex input) (table format) to a single step in a scenario |
| Yes, once for each data row | No, the scenario runs only once |
| Data used Across all steps via placeholders (<placeholder>) | Data used within a single step (passed as a DataTable obj) |
| **Syn:** Scenario Outline + Examples | **Syn:** Plain Scenario + table under a step |

@. Typical Tasks Written Under @BeforeSuite

1. Initializing Reporters

2. Initializing Logging

3. Database Connection Setup

4. Reading Global Configuration

5. Starting a Selenium Grid / Docker containers

6. Validating Environment Setup

@. Typical Tasks Written Under @BeforeTest

1. Launching the Browser

2. Reading Configuration Files

3. Connecting to Database

4. Setting up Test Data

5. Environment Setup

6. Logging Setup

@. Skip a TC in cucumber

To skip a TC in Cucumber, we can use any of the following methods depending on our context (Java + Cucumber):

1. Use the @Ignore annotation (JUnit only) (@Ignore works only with JUnit. If you're using TestNG, this won't apply)

2. Use Tags to Skip a Scenario: Mark the scenario with a specific tag like @skip, and then exclude it from your test run using the Cucumber @CucumberOptions.

3. We can throw a PendingException or conditionally skip based on some logic.

@. When using the TestNG H to failed TCs in the same execution Ref: Java\_WD\_Ex\TestNG\ReRunFailedTCs\_immediatlyAfterExecution

In TestNG, to re-run failed test cases in the same execution, you can use a feature called retry analyzer. This allows us to automatically retry a failed test a specified number of times during the same execution. Rerun after execution is not possible with test-output/testng-failed.xml.

@. To handle, a time when u had a disagreement with a team member? / To prioritize TCs in manual testing

In one of my previous projects, I had a disagreement with a developer regarding a defect I had logged. The issue was causing intermittent failures in the application, but the developer initially marked it as "Not Reproducible" and closed it, stating that everything worked fine on their end.

Rather than pushing back emotionally, I decided to take a calm and constructive approach. I reproduced the bug multiple times with clear steps, provide evidence by recorded screen capture, and attached detailed steps along with logs and environment details. I also suggested pairing up to debug it together.

Once we sat down and walked through it, the developer realized that the issue only occurred when a specific feature flag was enabled — something not set in their environment. They acknowledged the defect, reopened the ticket, and worked on the fix.

The experience taught me that clear communication, evidence-based reporting, and collaboration are key to resolving disagreements in a productive way. It actually helped improve our working relationship and mutual respect moving forward.

@. Give an example of a time when you had to adapt to a change in project requirements

In one of my recent projects, we were developing a web-based inventory management system. Initially, the client requested only desktop browser compatibility. So, our design and testing strategies were tailored for larger screens and mouse-based interactions.

However, midway through the development, the client decided to include mobile and tablet support due to a shift in their business strategy. This change significantly impacted the UI/UX design, TCs, and even some backend logic related to responsiveness.

**How I adapted:**

* I collaborated with the UI/UX team to quickly understand mobile-specific changes.
* Updated the test plan to include mobile browsers and devices using tools like BrowserStack.
* Adjusted automation scripts to be compatible with different screen sizes and touch gestures.
* Coordinated with developers to ensure that responsiveness and layout issues were fixed before final QA.

@. Describe a situation when u had to work under pressure to meet a deadline. H did u manage ur time and tasks

During my previous project, we had a critical product release scheduled, and just a week before the deadline, we discovered a set of high-priority bugs during the final round of testing. The timeline was tight, and the pressure was high because any delay would affect the client’s go-live plan.

To manage this situation, I first prioritized the issues using severity and impact. I coordinated with the development team to get quick fixes for the most critical bugs. Then I broke down my tasks into smaller chunks and created a daily checklist with clear goals.

I worked extended hours when necessary, and regularly communicated status updates to my team lead to ensure transparency. I also collaborated with teammates to distribute testing tasks efficiently.

As a result, we managed to complete the testing and validation on time, and the release was deployed successfully without any major post-release issues.

@. Give an Ex of a situation where ur automated tests found a critical bug before it reached production. H to handle it

In one of my previous projects, we were developing a web-based e-commerce platform. I had set up a suite of automated regression tests using Selenium integrated with a CI/CD pipeline (Jenkins). One of the critical TCs involved validating the checkout flow with multiple payment methods.

A few days before a major release, one of the nightly builds failed. On investigating the failure, I found that the automated test for the credit card payment flow was failing due to a redirect error — the system was looping back to the cart page after entering valid card details, instead of proceeding to the confirmation page.

I immediately verified the issue manually and raised a P1 bug in the bug tracking system (JIRA), attaching the logs, screenshots, and video from the test run. I also collaborated with the development team to help them pinpoint the recent code changes that introduced the issue. It turned out to be a logic error in a newly implemented discount validation module that was rejecting valid payment attempts.

Because the issue was caught before it reached production, we were able to fix it quickly, re-run the test suite, and proceed with the release on schedule. This incident reinforced the importance of having reliable end-to-end test coverage and CI feedback early in the process.

@. To run TCs parallel in cucumber

To run test cases in parallel in Cucumber, you need to configure your project with a test runner and a framework that supports parallel execution, such as TestNG, JUnit 5, or Cucumber JVM Parallel Plugin.

1. Cucumber with TestNG (Java)

Step 1: Use @DataProvider for parallel execution: Cucumber does not natively support parallel execution, but using TestNG's DataProvider allows you to run scenarios in parallel.

Step 2: Enable parallel execution in testng.xml

2. Cucumber with JUnit and Maven (Using cucumber-jvm-parallel-plugin)

3. Cucumber with JUnit5 + @Suite and Parallel Execution: If we are using JUnit 5, we can define parallel execution in junit-platform.properties

@. Explain contents of runner file in cucumber

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Basic Runner File using JUnit: | | | Runner File with TestNG: | |
| **Option** | **Description** | |
| features | Path to the .feature files (Gherkin scenarios). | |
| glue | Path to your step definition files. | |
| plugin | Enables report generation. E.g., HTML, JSON, pretty (Cucumber). | |
| monochrome | If true, removes unreadable char’s from the console o/p | |
| tags | Filters which scenarios to run by tag. | |

@. E WebDriver create statement line / WD creation statement in Selenium - WebDriver driver = new ChromeDriver();

@. If u want to execute Test Case again and again and again

1. Using TestNG RetryAnalyzer (Recommended for Failures)

2. Re-running Failed Tests using testng-failed.xml

3. Looping the Test with Custom Logic (Not best practice)

4. Use @DataProvider for Repeat Runs

**5. Using TestNG invocationCount**

6. Using while(true) or Infinite Loop (Not Recommended for Tests)

@. Scenario and Scenario Outline in cucumber

In Cucumber, both are used to write test cases in Gherkin syntax, but they differ in how they handle test data.

**Scenario (Single set of steps):** Used to write a Test Case with one specific set of inputs. It runs once with the provided steps and data.

**Scenario Outline (Multiple sets of steps with different data):** Used when the same scenario needs to be executed multiple times with different data. Requires an Examples table that provides different input values. Each row in the table triggers a separate execution of the scenario.

**@. To wait for specific ele to be clickable in selenium:** Using Explicit Wait i.e. WebDriverWait with ExpecetedConditions

**@. To verify the presence of an element on a page**



@. XPath to locate specific menu elements on a webpage using selenium java

1. Locate all menu items under a specific menu: //ul[@id='main-menu']/li/a

2. Locate a specific menu item by visible text: //ul[@id=' main-menu']//a[text()='Services']

3. Locate a menu item by partial text: //ul[@id=' main-menu']//a[contains(text(), 'Serv')]

4. Locate by attribute (e.g., href): //ul[@id=' main-menu']//a[@href='/services']



@. Window Handle and Window Title

Window Handle: A unique identifier assigned to each window opened by the browser.

Window Title: The name displayed in the title bar of the browser.

**isDiaplayed():** Checks if element is visible on page.

**isEnabled():** Checks if element is enabled for interaction.

**isSelected():** Checks if a check box or radio button is selected.

@. iframe / H to handle it

An iframe is an inline frame used to embed another document within the current HTML document. I can handle by using switchTo().frame().

@. Some selenium alternatives for browser automation ----- Cypress, TestCafe and Palywright

@. To run TCs in multiple browser using selenium

Use WD interface for different browsers **or** use tools like TestNG with browser parameters for cross-browser testing.

@. What are WebElements in selenium

WebElements represents elements on a web page (buttons, input fields) selenium can interact with those.

@. Can selenium handle HTTPS certificate issues

Yes, selenium can handle HTTPS certificate issues using browser-specific settings or desiredCapabilities.

@. Role of JavascriptExecutor in selenium

JavascriptExecutor allows executing JavaScript code with the browser, which can be useful for handling dynamic elements or scrolling.

JavaScriptExecutor is an interface that allows to execute JavaScript code directly within the context of the browser. It is used when certain actions cannot be performed using standard WebDriver methods or when you need to interact with elements that are not directly accessible through WebDriver.

@. navigate().refresh() and driver.get(navigate.getCurrentUrl())

navigate().refresh(): Refresh the browser window

driver.get(navigate.getCurrentUrl()): Loads the current URL again, which also results in a page refresh.

@. W r capabilities

Settings to configure WebDriver

**N:** To disable extensions add “disable-extensions” cmd

@. To store config or environment data?

@. Wr Maps commonly used in frameworks? ---- Step names, locator mapping, config, headers

@. To convert between List, Set, and Map?

List to Set: Set<T> set = new HashSet<>(list);

Set to List: List<T> list = new ArrayList<>(set);

Map keys to List: new ArrayList<>(map.keySet());

@. To iterate a Map with key-value pairs?

@. When to use TreeMap or TreeSet?

• TreeMap - When keys need to be sorted

• TreeSet - When values must be unique and sorted

@. H HashMap differs from LinkedHashMap?

• Quick way to detect duplicates in ID, email, etc.

• Works well in response validations or assertions

@. To handle multiple API responses?

• Store each response in a Map<String, Response>

• Label by test step or API endpoint

• Helps when chaining token generation → user access- validations

• HashMap - Fast, no order guarantee

• LinkedHashMap - Maintains insertion order

@. To make Collections thread-safe / To ensure thread safety in collections

• Use Collections.synchronizedList(), synchronizedMap().

• Use thread-safe collections for multi-threaded tests

@. When to use ConcurrentHashMap in tests?

• When sharing data like tokens across threads

@. To use CopyOnWriteArrayList in tests?

• Thread-safe alternative to ArrayList

• Good for read-heavy test scenarios

• Safe to iterate even when modified by other threads

@. Y do we use ArrayList in selenium

For dynamic sizing, fast access using index, and common for storing List<WebElement> from findElements().

@. H do u store key-value data in automation tests

Use HashMap<String, String> Ex: Map<String, String> testData = new HashMap<>(); (store config values, test data, API headers..).

@. Wr use Maps in Automation framework

Store: element locators, input data from Excel/JSON, or Expected vs Actual result.

@. Wr use TreeMap and TreeSet in Automation framework

TM: When sorted keys are needed (Ex: Module-wise test data)

TS: To store sorted, unique test identifiers or names

@. To store test steps or reusable locators

Use HashMap<String, By> and Map<String, String> (Map helps to manage element locators by logical names)

@. Pipeline in Jenkins

Pipeline defines stages and steps for CI/CD. Written in Jenkins file using Groovy.

@. To configure Maven with Jenkins -- Install Maven plugin, configure MAVEN\_HOME, and use pom.xml to trigger build.

@. Git commands used daily in QA common ---- Git clone, pull, add, commit, push, checkout, and branch

@. Element is visible but click fails ---- Use JsE to click or check if another element overlays it.

@. If multiple builds failing randomly ---- Check for flaky tests, timing issues, stale elements, or environment problems.

@. Approach to writing TCs for a Login page ---- Include valid, invalid, blank input, UI validations, and security checks.

@. To prioritize what to automate vs what not to automate

I prioritize based on risk, business impact, frequency of use, and stability of the feature. Critical business flows areas and high-risk areas come first. I avoid automating highly volatile features or one time use cases where ROI is low.

@. Share example wr u debug a complex automation failure

We once had random failures in CI - TCs were timing out inconsistently. After analysis, I found session timeouts due to a recent server config change. I collaborated with DevOps to adjust settings, implemented retry logic, and stabilized the tests.

@. To ensure test code is maintainable and scalable

Apply design patterns (like POM), and modularize utilities. I enforce code reviews, ensure proper naming, and keep tests independent. I also maintain good documentation and version control for stability.

@. To collaborate with DevOps, Developers, and Manual QA in Agile teams?

I actively participate in sprint ceremonies, work closely with Devs to identify test hooks, and align with QA to avoid duplication. I collaborate with DevOps on CI/CD optimization. Regular sync-ups ensure that automation aligns with sprint goals.

@. To measure the effectiveness of your automation suite

I track pass rate trends, failure rate, test coverage, execution time, and defect leakage post-release. We also review flaky test rates. These metrics are reported via dashboards and discussed with the team to drive continuous improvement.

@. To manage test data across multiple environments?

I use a combination of static datasets for consistent validation, dynamic data generators for robustness, and environment-specific configs. For sensitive cases, I leverage mocks/stubs. I ensure data isolation across parallel runs to avoid conflicts.

@. To handle test data parametrization in your framework?

I use TestNG's @DataProvider to pass data to test methods. For external data, I use Excel/CSV/JSON readers, keeping data separate from test logic. This allows easy scalability and better maintainability of TCs.

@. To implement reusable waits in framework Ref:

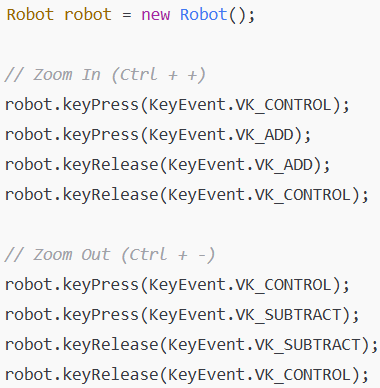
I create a WaitUtils class with reusable methods using WebDriverWait or FluentWait to handle dynamic elements. This avoids flaky tests and improves reliability.

@. H to handle if an element is visible but not interactable

It may be overlapped, hidden behind another element, or not at ready to handle this situation we have to use WebDriverWait, JsE, or Scroll into view.

@. C selenium simulate keyboard keys ---- Yes, using Actions class or sendKeys(Keys.ENTER)

@. To run selenium scripts in Docker container ---- By using official selenium docker images with docker-compose.



@. To perform zoom in / zoom out in selenium

By using JsE,

JavascriptExecutor js = (JavascriptExecutor) driver;

js.executeScript("document.body.style.zoom='50%';");

By using Robot class,

@. To highlight an element ---- Use JsE to set a border via DOM

@. To scale selenium tests ---- Use Selenium Grid, Docker, or cloud providers

@. To trigger tests after each build ---- Configure Jenkins with post-build actions or pipe line scripts.

@. Options classes in selenium 4 ---- Browser specific classes like ChromeOptions and FirefoxOptions.

@. WebDriverManager ---- A library that automatically downloads and manages browser drivers.

**@. Best practices for selenium automation** ---- Use POM, avoid hard waits, handle exceptions, modularize code, and use CI/CD

@. To validate presence of an element in selenium Ref: WD\_Examples\EX79/b/c\_ValidatePresentsOfAnElement\_

1. Validate presence in the DOM: To make sure the ele is exists in the HTML DOM (even if it’s hidden) use List and size().

2. Validate element is displayed (visible to user): If we want to check the element is present and visible use isDisplayed().

3. Using ExplicitWait (recommended for dynamic pages): If the element may take time to appear use ExplicitWait with ExpectedConditions presenceOfElementLocated

@. To handle multiple elements with similar attributes in selenium

1. Access by index

2. Filter by text or other attributes

3. Use parent/child relationships

@. To Scroll and validate lazy-loaded content (e.g., infinite scroll) Ref: WD\_Examples\EX81\_ScrollAndValidate\_LaztLoadedContent

Scrolling and validating lazy-loaded content by using JsE along with WebDriverWait strategies to scroll the page and then verifying the presence or state of the newly loaded elements.

@. To test responsive layouts in selenium

A responsive website should look and work correctly on different screen sizes (mobile, tablet, desktop). In Selenium, we can simulate this by changing the browser window size or using mobile emulation.

1. Change the browser window size

2. Use Chrome DevTools Emulation (Mobile Emulation)

3. Assertions for responsive elements:

* Verify visibility of elements (isDisplayed()).
* Verify CSS properties (getCssValue()).
* Verify element location/size (getSize() and getLocation()).

4. Combine with WebDriverWait

@. To automate captcha-based logins in selenium

We cannot directly handle CAPTCHA with Selenium, because CAPTCHA images/audio are intentionally made to block automation from accessing sensitive information. Since Captcha requires human input to verify that a user is not a bot, an automation test engineer must manually enter the Captcha while Selenium can fill in the other fields automatically.

1. Use a test environment without CAPTCHA (developer may disable CAPTCHA or provide a test bypass key)

2. Use a backdoor / pre-authenticated session (for tests)

3. Use a CAPTCHA solving service (not recommended for official tests)

4. Mock CAPTCHA on your dev environment (If CAPTCHA is mandatory in app even in QA): Developers can configure the CAPTCHA component in “test mode” (always returns a fixed token like TEST\_CAPTCHA\_OK).

@. To validate email verification links sent during signup using selenium

Selenium alone cannot read emails, but we can absolutely automate email verification flows by combining Selenium with an email‑reading approach.

1. User signs up via our UI (Selenium fills the form & submits).

2. Application sends a verification email with a link or token.

3. Use a test email inbox accessible by automation

4. Test retrieves the latest email from a test mailbox

5. Use Selenium to open the link

@. Self-healing in automation and is it possible with selenium

When we write automated UI tests, our locators (XPath, CSS, ID, etc.) often break if developers change the UI. Self‑healing automation means our test framework is smart enough to detect a broken locator at runtime and automatically try alternative locators or strategies without failing the whole test. Plain Selenium WebDriver does not have self‑healing built in. Selenium will throw an exception if a locator is invalid.

@. To control execution order using TestNG

1. Using priority attribute in @Test

2. Using dependsOnMethods

3. Using testng.xml to control class/method order

@. To create Custom annotations using TestNG Ref: TestNG\Custom\_TestNGAnnotation

Step 1: Create a custom annotation

Step 2: Use the custom annotation in our TestNG test

Step 3: (Optional) Read annotation details at runtime

@. In TestNG, h to ensure screenshots are taken only for failed TCs

Use ITestListenerinterface and implement onTestFailure() to capture screenshots.

@. Suppose 49 of 200 tests failed, h to collect only failed test data using TestNG

Using Listeners or IReporter, to fetch failed tests and log/store them separately, or using gtestng-failed.xml

@. To work with configuration files in selenium

Use java properties class to read key-value pairs from .properties file for reusable configurations like URLs, credentials,…

@. To scroll to the bottom of the page ---- jse.executeScript(“window.scrollTo(0, document.body.scrollHeight)”);

@. To manage WebDriver instance in parallel execution Ref: TestNG\Manage\_WD\_Instance\_inParallelExecution

When running Selenium tests in parallel, the main challenge is ensuring that each test thread gets its own WebDriver instance without interfering with others. If multiple tests share the same driver object, we’ll get problems like test failures, wrong browser actions, or session clashes. The following steps has to follow.

1. Use ThreadLocal<WebDriver> class to ensures each thread gets its own copy of the driver (i.e. WebDriver instance), avoiding conflicts parallel execution in TestNG or CI/CD pipeline.

1b. Usage in test setup: By using @BeforeMethod and @AfterMethod

2. Configure Parallel Execution in TestNG: Add parallel="methods" or parallel="classes" in your testng.xml and specify thread-count.

3. Avoid Global Static WebDriver: i.e. 'public static WebDriver driver;' then multiple threads will overwrite each other’s driver instance, leading to flaky tests.

4. Centralize Driver Creation with a Factory.

@. Different annotations used in Selenium

JUnit Annotations- Test, BeforeClass, AfterClass, Before, After, Ignore, RunWith

TestNG Annotations- BeforeSuite, BeforeTest, BeforeClass, BeforeMethod, Test, AfterMethod, AfterClass, AfterTest.

@. W WebDriver-supported Mobile Testing Drivers

Various WebDriver for mobile testing by Selenium are: AndroidDriver, IphoneDriver, OperaMobileDriver.

@. To deal with pop-ups based on Windows

Selenium is an automation testing tool that only enables web application testing; it does not support testing of Windows applications. However, while Selenium cannot alleviate the matter on its own, it is overcome with the support of a third party. Along with Selenium, there are various third-party solutions available for managing window-based pop-ups, such as AutoIT, Robot class, and so on.

@. W exactly is an Object Repository? In Selenium, H to construct an Object Repository?

The phrase "Object Repository" refers to a collection of web elements belonging to the Application Under Test (AUT) as well as their locator values. As a result, the location value may be fetched from the Object Repository whenever the script uses the element. Instead of hardcoding locators into scripts, Object Repository stores them in a centralised location. An Excel sheet saves the object in Selenium, which can then be populated within the script as needed.

@. W r regular expressions, and H to use them in Selenium? Ref: WD\_Examples\EX80\_RegExpression\_or\_regex

Regular expressions are specific text strings that represent search patterns. Regexp is a keyword that can be used as a prefix to treat a text as a regular expression in Selenium.

**Ex:** driver.findElement("//input[contains(@name,'somestring')]")

* It will look for the input tag that has the 'name' property and include the text 'somestring'.
* Here the method “starts-with()” implies that the string begins with the provided text

@. Strategies to improve the performance of Selenium tests?

There are some strategies for improving the performance of Selenium tests:

We can use explicit waits instead of Thread.sleep()

We can implement parallel execution using TestNG or JUnit

We can use headless browser testing for faster execution

We can minimize browser window for quicker rendering

We can use the Page Object Model(POM) for better maintainability

We can implement proper test data management

We can use appropriate locator strategies (prefer ID and CSS over XPath)

We can regularly update Selenium and browser drivers

@. To handle dynamic web elements in Selenium?

We can handle the dynamic web elements using various strategies:

We can use partial locators( contains(), starts-with(), and ends-with()) in the XPath.

We can use WebDriverWait for explicit waits

We can use JavaScript Executor to find elements

We can implement the custom expected conditions

@. To handle tests that need to verify email content in Selenium?

Testing email content requires integration with email APIs or services. We can follow these steps:

We need to set up a test email account or we can use a temporary email service.

We can perform the action that triggers the email (e.g., user registration).

We can use JavaMail API or a REST API to fetch the latest email.

We can parse the email content (or we can use libraries like JSoup for HTML emails).

We need to assert the email content.

@. To capture browser console logs in Selenium

Use LogEntries from driver.manage().logs().get("browser") (works in Chrome).

@. To ensure your Selenium tests are not flaky

* Use Explicit Waits (not Thread.sleep).
* Avoid hard locators (use robust selectors).
* Stabilize environment (mock data, retry logic).

@. C Selenium handle CAPTCHA or OTP-based logins

No, Selenium cannot solve CAPTCHAs/OTPs directly. Solution for this.

* Disable CAPTCHA in test environments.
* Use third-party APIs (e.g., OCR, image recognition) in rare cases.
* Fetch OTP from DB/Email/SMS mocks.

@. C Selenium handle the Windows File Upload dialog

No (Selenium only controls the browser). Use Robot class, AutoIT, or SendKeys to the <input type="file">.

@. If an element is present in DOM but not visible on the UI

isDisplayed() → checks visibility.

Clicking such elements throws ElementNotInteractableException.

Fix: Wait until visible, scroll into view, or interact via JavaScript.

@. Diff bn driver.findElement(By.xpath(…)) and using JavaScript document.querySelector?

Selenium locators → WebDriver API.

JSExecutor → Direct DOM execution (faster but less portable).

@. To test an application with no documentation

Start with exploratory testing. Walk through the app manually, take notes, map out key workflows, and document assumptions. Ask clarifying questions when needed. Treat it like reverse-engineering.

@. To decide what not to test?

Use risk-based testing. Prioritize based on user impact, complexity, and likelihood of failure. Document your reasoning and involve stakeholders when needed.

@. A bug is found in production. Wt check first

Check logs and monitoring tools. Look at recent deployments. Try to reproduce in lower environments. Focus on containing the issue and communicating impact.

@. Wt would you test if you were the only QA on the team

Prioritize critical flows. Start with smoke tests and regression coverage. Build automation for high-value tests. Focus on quality culture, not just test cases.

@. Our tests are taking too long in Cl. Wt would do

Look into test parallelization. Split tests by tags or type. Run fast checks early (like smoke tests) and move full suites to nightly jobs. Optimize setup/teardown.

@. Wt would we do if you found a bug the day before release

Assess severity and risk. If it's critical, escalate immediately. If it's minor, log it, communicate clearly, and track it post-release. Focus on impact.

@. To test a third-party integration

Validate data exchange, failure scenarios, and boundary conditions. Use mocks if the third party is unreliable. Confirm error handling is in place.

@. The app is stable but users keep reporting bugs. Wt would we do

Look for environment or data differences. Reproduce using real user flows. Analyze logs and monitor usage patterns. Consider usability or edge cases.

@. To know when testing is "done"

Use test coverage, risk assessment, and stakeholder confidence as indicators. If major flows are covered and there are no blockers, it's reasonable to stop.

@. U find multiple bugs in the same area. Wt does that tell us

It suggests weak coverage or a fragile part of the codebase. Consider targeted exploratory testing and deeper regression around that area.

@. To handle duplicate bug reports

Link them together. Combine details to get a fuller picture. Communicate updates in one place to avoid confusion and redundancy.

@. Ur joining a QA team that has no automation. What's ur plan

Start small. Automate smoke tests first. Pick a scalable tool. Get early buy-in. Focus on maintainability, not just speed.

@. A developer wants to skip a test to release faster. Wt do u do

Ask why. Communicate the risks. Offer alternatives like running a reduced suite. If needed, escalate with context.

@. Ur given a new feature to test with only a title in Jira. Wt now

Ask for clarification. Get acceptance criteria or a demo. Use exploratory testing to build a base. Collaborate with the dev or PM.

@. If u suspect a bug, but it's not consistently reproducible

Try to isolate variables (data, timing, environment). Add logs or screen recordings. File the bug with all findings and label it as intermittent.

@. To communicate QA value to a product team

Talk about risk reduction, user trust, faster feedback, and issue prevention. Show examples where QA found issues early or prevented rework.

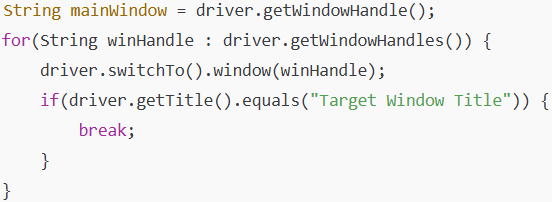
@. What's ur process when handed a legacy test suite u didn't write

Audit the suite. Run it locally. Read the code to understand structure. Identify flaky or redundant tests. Refactor slowly with small wins.

@. Login page where the username and pwd fields sometimes take extra time to load. H will handle synchronization?

Use WebDriverWait strategy with ExpectedConditions.visibilityOfElementLocated

@. There are multiple windows opening during a test. H will switch to a specific window by its title



@. Upload a file, but the upload button is not a normal <input type="file">. It opens a native file dialog. H to handle it

Use Robot class with keyPress() and keyRelease() methods along with keyEvent class providing methods.

@. To test a responsive web application (different screen sizes). How would you do that in Selenium

Use Chrome DevTools options or set window size: driver.manage().window().setSize(new Dimension(375, 812));

@. To handle a situation where an element is not clickable because another element overlays it?

Scroll the element into view: ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView(true);", element); Or click via JavaScript: ((JavascriptExecutor) driver).executeScript("arguments[0].click();", element);

Or wait for overlay to disappear with ExpectedConditions.invisibilityOfElementLocated().

@. To validate email verification links sent during signup

Selenium cannot access emails directly. Possible approaches:

* Use Java Mail API to fetch emails.
* Integrate with temporary email APIs (like Mailinator).
* Extract the verification link and open in driver.get(link);.

@. Common challenges faced in automation testing, and to overcome them?

Handling dynamic elements: Used XPath with contains() or explicit waits.

Cross-browser compatibility issues: Tested scripts on different browsers using WebDriverManager.

Test data management: Used Excel/JSON files for external data storage.

Script maintenance: Followed Page Object Model (POM) to keep test scripts organized.

Overcoming these challenges improves automation efficiency and reduces flaky tests.”

@. To manage test execution order in a Selenium-Cucumber framework?

In Selenium-Cucumber framework, execution order can be controlled at multiple levels, depending on whether we want to manage the order of features, scenarios, or steps:

**1. Feature File / Scenario Order:**

* By default, Cucumber executes scenarios in the order they are defined in the feature file(s).
* If we want strict control across multiple feature files, we can:
* Prefix feature/scenario names with numbers (e.g., 01\_Login.feature, 02\_Checkout.feature).
* Configure your runner class to pick features in a specific order.

**2. Using Tags for Ordered Execution:** We can assign tags and control execution order from the runner or command line.

**3. Controlling Execution with JUnit/TestNG:** If we are running Cucumber with TestNG, you can control order using:

@Test(priority=1) annotations in glue code.

testng.xml file to define suite/parallel execution order.

**4. Step Execution Order:** Inside a single scenario, step definitions always execute in the order written in the .feature file. If dependency exists between scenarios (not recommended in BDD), we can use: Hooks (@Before, @After) with order attribute.

@. To handle NoSuchElementException

NoSuchElementException will occurs when WD is unable to find the element on the page using the locator we provided.

**Common Causes:**

Wrong locator (XPath/CSS/ID not matching).

Element not yet rendered (page loads slowly, AJAX, dynamic content).

Element inside iframe but you didn’t switch to it.

Stale page or navigation happened before finding the element.

Hidden elements (not visible in the DOM yet).

**Handling Strategies:**

1. Use Explicit Waits

2. Check Locators

Verify the locator in browser DevTools (F12 → Inspect → $x("...") or document.querySelector("...")).

Prefer unique locators (id, name) before using long xpath.

3. Handle iframes: If the element is inside an iframe

4. Graceful Exception Handling (try-catch)

5. Use Fluent Wait (Retry with Polling)

6. Page Synchronization: Make sure navigation is complete before searching



7. Debugging Step: Print driver.getPageSource() if element is not found → helps check if element exists in DOM.

@. Types of navigation commands --- to(), back(), forward(), and refresh()

@. Exception test in Selenium ---- Ref: TestNG\TestNG\_Features\E16\_ExceptionTestExample

In Selenium, an exception test is used when you expect a specific exception to occur during a test. If a TestCase is designed to throw a particular exception, we can use the @Test annotation in TestNG and specify the expected exception type in its parameters. This way, the test will pass only if that exception is thrown. If no exception or a different exception is thrown, the test will fail.

@. H to wait until a web page has been loaded completely in Selenium

In Selenium, we can use two main types of waits to ensure elements load before interacting with them: Implicit Wait, Explicit Wait. These waits help manage delays in loading, making tests more reliable.

@. What is Selenium WebDriver

Selenium WebDriver is a robust open-source framework for automating web browsers, primarily aimed at easing the testing and verification of web applications. As an important part of the Selenium suite, WebDriver offers a programming interface to interact with web browsers, allowing developers and testers to automate browser actions seamlessly.

@. Is Selenium WebDriver a library --- Selenium WebDriver is not a library.

@. H Selenium 4 is different from other Selenium versions

Selenium 4 is the latest version of the Selenium testing tool, which bringing new features and improvements over older versions. It offers better support for modern web testing needs, with updated browser compatibility, enhanced debugging, and user-friendly updates for testers.

@. Wt will happen if I execute this command? driver.get --- This command tells the Selenium WD to open the given URL in the web browser.

@. To scroll down a page using JavaScript?

scrollBy() allows to scroll a webpage by a spec’d num of pixels. Syn: executeScript("window.scrollBy(x-pixels, y-pixels)");

x-pixels: Horizontal scroll amount (0 if you don’t want horizontal scrolling).

y-pixels: Vertical scroll amount (positive values scroll-down, negative values scroll-up).

@. To retrieve CSS properties of an element?

To retrieve CSS properties of an element in Selenium WebDriver, we can use the getCssValue() method

For Ex Call the getCssValue("property-name") on the found element to get the value of the desired CSS property.

// Retrieve the background color of an element with ID "header"

String backgroundColor = driver.findElement(By.id("header")).getCssValue("background-color");

This code finds the element with ID "header" and retrieves its background color. we can replace "background-color" with any other CSS property name to get its value.

@. H does Selenium handle Windows-based pop-ups

Selenium is mainly used for testing web applications and doesn’t support Windows features on its own. However, you can use third-party tools like AutoIT or Robot to work with pop-ups and other Windows-based.

@. Wt is a Selenium Maven project

A Selenium Maven project is a software project that uses Maven to handle the project's dependencies (like libraries needed for testing) and to manage the build process (how the project is compiled and packaged). This makes it easier to organize and run Selenium tests efficiently.

**@. Challenges of Parallel Execution in Selenium**

Parallel execution in Selenium brings many benefits, like faster test execution and improved test coverage, but it also comes with its own set of challenges:

**1) Test Dependencies:**

Data isolation: Tests that share data or rely on specific states can interfere with each other when running in parallel. Careful test design and data management are crucial.

Test order dependencies: Some tests might need to run in a specific order due to logical dependencies. Parallel execution might disrupt this order.

**2) Resource Management:**

WD limitations: Not all WebDrivers support parallel execution. For example, EdgeDriver requires special handling.

Resource contention: Running multiple browser instances simultaneously can strain your machine's resources (CPU, memory, network). Consider using grid-based solutions or cloud services for larger-scale testing.

Test stability: Parallelization can introduce flakiness in tests due to unforeseen interactions between tests.

**3) Other Challenges:**

Synchronization issues: Tests might access shared resources, leading to race conditions and unpredictable behaviour. Proper synchronization mechanisms are needed.

Debugging difficulties: Identifying the root cause of failures in parallel execution can be complex due to multiple test interactions. Logging and reporting tools are helpful.

Test environment setup: Setting up and managing multiple test environments for parallel execution can be complex and time-consuming.

@. To integrate Selenium with Jenkins for Continuous Integration

1. Setting Up Jenkins:

* Install necessary plugins: Install plugins like "Pipeline", "Git", and any relevant browser-specific plugins depending on your needs (Ex: BrowserStack Plugin).
* Configure JDK: Define a Java installation for running your Selenium tests.

2. Preparing your Selenium Tests:

* Organize our tests: Structure our tests into well-defined modules or classes depending on your testing framework.
* Manage dependencies: Ensure your test scripts have access to required libraries and resources.
* Consider using a testing framework: Frameworks like TestNG or JUnit help manage dependencies and parallel execution.

3. Creating a Jenkins Job:

* Choose a job type: Select "Pipeline" for declarative builds or "Freestyle Project" for a more traditional config.
* Define job triggers: Choose how you want the job to run (e.g., manually, upon code changes, scheduled builds).
* Configure build steps:
* Download code: Use the "Git Plugin" to fetch code from your version control system.
* Install dependencies: Download and install required libraries for your tests (e.g., Maven build step).
* Execute tests: Use a "Shell" or "Execute Windows batch command" step to run your Selenium tests.
* Use environment variables to specify browser configurations and other test parameters.
* Capture test results and logs for reporting.
* Publish reports: Utilize plugins like "JUnit" or "XUnit" to publish test results in Jenkins dashboard.

4. Additional Considerations:

* Parallel execution: Utilize tools like Selenium Grid or cloud testing services for efficient parallel execution.
* Security: Securely store credentials and sensitive information using Jenkins credentials management.
* Reporting & analysis: Integrate reporting tools and dashboards for better visualization and analysis of test results.

@. WebDriver and WebElement.

|  |  |  |
| --- | --- | --- |
| **Feature** | **WebDriver** | **WebElement** |
| Purpose | Controls the browser itself | Represents an element within a web page |
| Scope | Browser window or tab | Specific element like a button, text field, or image |
| Interactions | Opens/closes browsers, navigates to URLs, refreshes pages, manages windows | Clicks, enters text, retrieves text, submits forms, interacts with element attributes |
| Selenium mtds | get(), navigate(), quit(), switch\_to(), etc | click(), sendKeys(), text(), getAttribute(), isDisplayed() |

@. Challenges and Solutions of Handling Dynamic Tables

Challenges: Finding the right element, Extracting the right info

Solutions: Think like a detective, JavaScript to the rescue.

**@. To design a scalable & maint’le Sele aut FW? Discuss the key components and design patterns u would incorporate**

Designing a scalable and maintainable Selenium automation framework requires focusing on modularity, reusability, configurability, and parallel execution.

**Goals:**

Scalable: Can easily add more tests, browsers, or environments.

Maintainable: Easy to update locators, logic, or configurations.

Reusable: Common utilities and methods are shared across tests.

Readable: Test cases clearly express business intent.

Integrable: Works smoothly with CI/CD tools, reporting, and version control.

**Key Components:**

1. Base Layer (Driver Management): Centralized WebDriver setup and teardown. And Supports multiple browsers, parallel execution, and remote drivers (Grid, Docker, or cloud).

Design Pattern Used:

Singleton Pattern (for driver instance control)

Factory Pattern (for creating browser instances dynamically)

2. Page Object Model (POM): Encapsulates locators and page actions inside dedicated classes. And can be extended to Page Factory (for lazy element initialization)

3. Test Layer: Holds test scripts that use page objects to perform actions and assertions. And Keep tests readable and free of UI details.

4. Utilities & Helpers:

Reusable functions for: Waiting (explicit waits), Reading config files, Handling Excel/JSON test data, Screenshot capture

Ex Utilities: WaitUtils.java, ConfigReader.java, ExcelUtils.java

5. Test Data Management

Use externalized data sources: .properties, .json, or .xlsx files. Or DataProvider in TestNG for parameterization.

Design Pattern Used: Data-Driven Testing Pattern

6. Reporting & Logging

Integrate: ExtentReports / Allure for HTML reports, Log4j / SLF4J for logs, TestNG Listeners for capturing test events (pass/fail/screenshots).

Design Pattern Used: Observer Pattern (listeners notify report/log components)

7. Parallel Execution: Use TestNG parallel execution, Selenium Grid, or Docker (Selenoid), Thread-safe driver handling via ThreadLocal, Enables horizontal scalability.

8. Continuous Integration (CI/CD):

* Integrate with Jenkins, GitHub Actions, or GitLab CI.
* Trigger tests on code commits or scheduled builds.
* Publish HTML reports as artifacts.

9. Configuration Management:

* Store environment URLs, browser settings, and timeouts in a config file.
* Allow overriding values via command-line or CI variables.

**Workflow:**

* TestNG → Reads test suite configuration
* DriverManager → Initializes browser driver
* BaseTest → Sets up environment before each test
* Page Objects → Encapsulate locators & logic
* Utils → Handle waits, logs, and data
* Listeners → Capture results and screenshots
* Reports → Generate summary HTML/Allure report
* CI/CD → Executes framework headlessly in pipeline

**Best Practices:**

* Use Explicit Waits (avoid Thread.sleep)
* Keep locators resilient (avoid absolute XPaths)
* Version control your test code (Git)
* Tag tests for grouping (smoke, regression)
* Maintain clean separation between test logic and framework logic

@. H do u approach TestNG process if u r in day 1 of the sprint

* Understand Requirements and Scope
* Set Up / Update TestNG Framework
* Create or Update Test Cases
* Plan Test Execution Strategy
* Define how tests will run: Smoke tests → Daily, Regression tests → Before release, Feature/specific tests → On story completion
* Review and Optimize
* Daily Follow-up in Sprint

@. Which priority test suites you run in automation

In automation, I don’t always run all P1, P2, and P3 test suites together.

The suite selection depends on the stage of the sprint, the type of build, and the purpose of execution.

Typically, I prioritize test execution based on risk, frequency of use, and stability of features.”

* During Daily Builds (CI Pipeline): I usually run P1 (Smoke or Sanity) tests only. These cover critical functionalities — login, navigation, payment flow, API health checks, etc. Purpose: Quickly validate that the build is stable and the environment is functional. **Ex:** @Test(groups={"Smoke"}) - Run automatically after every deployment.
* During Sprint Automation Execution: I run P1 + P2 suites. These include both critical and major functional test cases related to user stories delivered in that sprint. Executed once the build is stable or during regression mid-sprint.

**Ex:** @Test(groups={"Smoke", "Regression"})

* During Full Regression (Pre-release / End of Sprint): I run P1 + P2 + P3 suites (i.e., complete regression). P3 cases usually cover non-critical, edge, or low-risk scenarios. Purpose: Validate the entire application to ensure no impact on older modules.

**N:** For Hotfix or Quick Deployment: Only P1 or a subset of impact-based tests (using tagging or selective suites). Ensures quick feedback and faster turnaround.

**N:** This whole thing is managed in TestNG

@. To reduce bugs when implementing automation scripts

1. Design a Robust Framework

2. Start with Stable Locators (Use stable attributes)

3. Implement Code Reviews

4. Use Proper Synchronization

5. Implement Logging and Reporting

6. Add Exception Handling and Validation techniques

7. Use CI/CD and Run Regularly

8. Continuous Learning and Maintenance

@. Approach to finding and locating links on a web page using selenium java

A link in a web page is represented by the <a> tag. Selenium can locate these links using several locator strategies those are By LinkText, PartialLinkText, TagName, XPath, CSS Selector.

@. To integrate cucumber with TestNG

By default, Cucumber runs using the JUnit runner. However, we can integrate it with TestNG to achieve the following TestNG advantages:

* Use TestNG features (like priorities, parallel execution, groups, Data providers, listeners, Reporting)
* Run Cucumber scenarios as TestNG tests
* Integrate better with CI/CD and reporting tools

**Steps to integrate cucumber with TestNG:**

1. Required Maven Dependencies (in pom.xml): selenium-java, cucumber-java, testng, cucumber-testng
2. Create a Sample Feature File
3. Step Definition File
4. Create a TestNG Runner Class
5. Run Test Using testng.xml

@. Automate a scenario to verify if a user is able to scroll down a webpage until the footer section is visible

* Use JavaScript to scroll down the page.
* Check Footer Visibility: Verify if the footer is displayed.

Ex: WebElement footer = driver.findElement(By.id("footer"));

assertTrue(footer.isDisplayed());

end#2

**Scenario Based Qns**

**@. In case Yesterday code executed properly but today not working and H It will troubleshoot**

Network issue, Server issue (because of this page not loading prop), Browser updated

**Check for Env changes:** Ensure there have been no changes in the testing environment, such as browser updates, driver updates, or changes in the test data.

**Analyse Logs:** Review the logs to identify where the failure occurred. Logs often provide clues about what went wrong.

**Inspect the Application:** Check if there have been any updates or changes to the application itself, which could affect the locators or the workflow.

**Check Locators:** Verify if any locators have changed or if elements have become dynamic. Use tools like the browser's developer tools to revalidate the locators.

**Debug the Code:** Use breakpoints and debug the script to step through the code and identify the point of failure.

**@. In production yesterday code worked properly and today not working properly getting very slow what is your root cause analysis.**

First check the load

Code breakage in existing code

Users side may be internet issues are there

Yesterday browser version is 72 and night updated to 74 which is not compatible by the website got deployed

May be browser changed

**N:** Whenever root cause analysis term came give answer like this: I will refer the log files

**@. Development env application working fine and in production env application facing some issues**

1. Proper deployment maybe not done in the production env

2. Production database not updated properly after each and every run

**@. If a Test fails, what will be ur next step**

Re-run the Test: Sometimes, tests fail due to temporary issues. Re-running can help confirm if the failure is consistent.

Analyse the Failure: Review the failure message and logs to understand the cause.

Check for Flakiness: Determine if the test is flaky and fails intermittently. If so, investigate why.

Fix the Issue: Based on the analysis, fix the identified issue in the script or report a defect if it's an application issue.

Update the Test Case: If the application has changed, update the test case and the corresponding script.

**@. If the application has minor changes, what would be ur approach to modifying the Automation scripts**

Impact Analysis: Identify which scripts are affected by the changes in the application.

Update Locators: Modify the locators if elements have been moved or renamed.

Refactor Code: If needed, refactor the code to handle new workflows or additional validations.

Re-run Tests: After updating the scripts, run the tests to ensure they work correctly with the new changes.

**@. Automate login functionality for a website**

Locate Elements: Identify the username, password fields, and login button using different locator strategies.

Write the Script: Use Selenium WebDriver to interact with these elements.

Validate: Add assertions to verify successful login, such as checking for the presence of a logout btn or a welcome msg.

**@. Automate a Test scenario where u need to check if an email is sent after a user registration**

**Steps:** Ref: ExtentReports\_v5\_2\src\test\java\Pr\_RegistrationEmailTest\RegistrationEmailTest

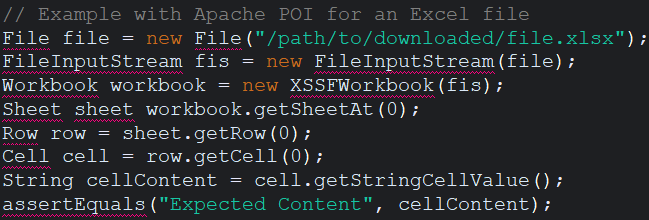
* Perform registration on the web application using Selenium.
* Connect to the email inbox using JavaMail API (To access inbox (Gmail, Outlook, etc.) or another email library’s like GreenMail, MailSlurp, Mailinator API.
* Search for the expected email (subject, sender, or content).
* Validate that the email was received and possibly contains a confirmation link or correct text.
* (Optional) Click or extract the verification link inside the email.

**@. If there is a scenario that takes a long time to execute, would u prefer Manual Testing or Automation Testing? Y?**

Prefer Automation: For long-running scenarios, automation is preferred as it can run unattended, saving time and resources.

Efficiency: Automation scripts can be executed multiple times without additional effort.

Reliability: Automated tests reduce the risk of human error and provide consistent results.

**@. Automate a scenario wr u need to validate the contents of a downloaded file after clicking a button on a webpage**

Trigger Download: Automate the click action to download the file.

Wait for Download: Ensure the file is down’ded completely, possibly by checking the file sys

Read File: Use appropriate libraries to read the file content.

**@. Automate a scenario where you need to verify a specific color, font, and position of an element on a webpage**

- Locate Element: Identify the element using locators.

- Retrieve CSS Properties: Use Selenium to get the CSS properties.

**@. U've been asked to automate a legacy application. Wt is ur approach**

Understand the Application: Study the application's workflow and technology stack.

Identify Challenges: Determine potential challenges like lack of modern web elements or non-standard interfaces.

Choose Tools: Select appropriate tools that can handle the legacy technologies.

Incremental Automation: Start with small, high-impact areas and gradually expand the automation coverage.

**@. Automate a functionality that is not yet fully developed. H to handle this**

Mock Data: Use mock data and services to simulate the incomplete functionality.

Design Flexible Scripts: Write scripts that can be easily modified as the functionality evolves.

Collaboration: Work closely with developers to understand the expected behaviour and changes.

@. Ur automation scripts are running slowly. H can u improve the speed

Optimize Locators: Use efficient locators like IDs instead of complex XPaths.

Reduce Waits: Minimize the use of `Thread.sleep` and use explicit waits where necessary.

Parallel Execution: Run tests in parallel using tools like TestNG or JUnit.

Headless Browsers: Use headless browsers like Chrome or Firefox to speed up execution.

@. Ur automation script is failing due to a change in the application. H to handle this

Update Locators: Modify the locators to reflect the changes in the application.

Refactor Code: Adjust the script to accommodate new workflows or element properties.

Run Regression Tests: After making updates, run regression tests to ensure no other scripts are affected.

Version Control: Use version control to track changes and rollback if necessary.

**HR Smart Ans**

@. Do u have any plans to go abroad

"No, I don't have any plans to go abroad right now. My focus is on building my career here and contributing long-term to the organization."

@. Where do u see yourself in 5 years

"In 5 years, I see myself growing within this organization, handling bigger responsibilities, and being part of important projects. My aim is continuous learning and career growth here."

@. If u get a better offer after joining us, will you leave

"No, I value stability and growth. Once I commit to a role, I focus on learning and contributing long-term, not jumping for short-term benefits."

@. Y should we hire u

"Because I have the required skills, I am quick to leam new tools, and I can adapt to team and project requirements. I'm confident I can add value to your testing team and contribute effectively."

@. Wt is ur biggest weakness

"Sometimes I spend too much time on details, but I've been working on balancing perfection with deadlines by prioritizing tasks better."

@. Wt if u don't get selected

"I'll take it as a learning experience, work on improving myself, and keep trying. But I believe I have the right skills for this role."

@. Do u prefer working alone or in a team

"I enjoy teamwork bcz it brings diverse ideas, but I can also work independently and take full responsibility if needed."

@. R u ready to work under pressure or late hours

"Yes, I can handle pressure when required. I believe proper planning and teamwork help in meeting deadlines effectively."

@. Y do u want to join IT/testing after X years of experience in another field

"I gained valuable skills in my previous field, especially in handling clients and requirements. But my interest has always been in IT/testing, and I have upskilled myself with tools and technologies to make this career shift successfully."

@. Y is there a gap in ur career

"During that time, I focused on upskilling myself, learning tools like Selenium, Postman, and SQL. Now I'm fully prepared to apply my skills in a real project environment."

@. Wt will u do if a developer refuses to accept a bug

"I would provide clear proof with screenshots, logs, and steps to reproduce. If the conflict remains, I would escalate it respectfully with proper documentation."

@. If you find a critical bug just before release, what will do

I would immediately report it with evidence, explain the potential impact, and discuss with the lead/manager whether to delay the release or provide a quick fix."

@. H do u handle repetitive or boring test cases

A: "I stay focused by breaking tasks into smaller parts. If possible, I look for automation opportunities to reduce repetitive work."

@. Wt if u and another tester give different results for the same TC

"I would recheck the steps, environment, and data. Then discuss with the other tester to align our findings. If needed, I'd involve the lead for clarity."

@. Wt if u miss a bug in production

"I would accept responsibility, analyse why it was missed, and improve the test cases or process to avoid similar misses in the future."

@. H do u prioritize testing when deadlines are tight

"I would focus on high-risk and business-critical areas first, then cover medium and low-risk areas as time allows."

@. If a client reports a bug that you cannot reproduce, what will do

"I will ask for more details like environment, inputs, or screenshots. I'll try to replicate it under similar conditions. If still not reproducible, I'll involve the developer or client for clarification."

@. Wt motivates u to work in testing

"The challenge of finding issues before customers face them motivates me. I enjoy improving product quality and being the user's advocate in the development process."

@. Do u prefer manual testing or automation

"Both are important. Manual testing is essential for usability and exploratory testing, while automation is powerful for regression and repetitive scenarios. I can work with both as per project needs."

@. Suppose u don't get onsite opportunities, will u be disappointed

"Not at all. My priority is to learn, grow, and add value in the role I'm given. Onsite is an additional opportunity, but not a deciding factor for my commitment here."

**MANUAL**

**@. Software Testing / Various testing types**

Software Testing is a process of identifying the bugs in the software productne. Here Test engineer will check whether the software product was developed as per the client’s requirements or not. Here we can ensure for quality.

**Under Testing Approaches:** Structural or WBT (done by Developer), Functional or BBT (done by Tester) and GBT.

**Under Testing Types:** Functional Testing (Smoke, Sanity, Regression Testing, etc.), Non-Functional Testing

**@. SDLC / Phases available in SDLC / Project Execution Flow (SDLC)**

SDLC is a process of developing various quality software’s. It is a process for project execution flow. To develop and deliver any software product SDLC process can be followed. It describes the development process of software product to fulfil the client requirements within the specified cost and time. Which contains various phases like:

* Requirements capturing and analysis (Gather and document detailed requirements from stakeholders. For this Business Analyst (BA) / PM Conducts a detailed discussion with the client to understand their business needs. After that analyze these requirements and documents their requirements in formats like BRD (Business Requirements Document), user stories, or use cases)
* Planning (Estimate time, cost and resources)
* Design (Design the system architecture. Design documents like HLD and LLD)
* Coding (Developer starts writing the code for the application)
* Testing (Tester teste the software to find and fix bugs)
* Deployment / Implementation / Live Environment / Production (Mainly deployment team and from testing side key resource guy may has to involve) (After proper testing the application release that software to users - Deploy the software to the production environment (server) or end-users. Train end-users if required)
* Delivery / Maintenance / Support (Tester activity) (Monitor software performance, Address user-reported issues and bugs, Implement updates or enhancements after release)

**@. Error, Defect, Bug, failure and Issue**

**Error:** Mistake in a coding is called ‘Error’ it’s like syntax error or logical error. Along with n/w failure and sys downtime.

**Defect:** The deviation between expected result to actual results in AUT (Application Under Test).

**Bug:** Defect which is suspected by the development team is called Bug.

**Ex:** Clicking [Login] with valid credentials doesn’t redirect to dashboard **/** Error message shows the wrong information **/** UI overlaps on smaller screen sizes **/** App crasher when we upload a large file

**Failure:** A deviation from the specified or expected behavior that is visible to End-users is called “failure”

**Issue:** TL/TM/PM/Client point of view Bug, Defect, Error, failure all are considered as issues.

**@. Re-testing and Regression testing / For Regression testing, how to decide what to include**

**Retesting:** Re-testing the application to verify whether the defects have been fixed or not.

**Regression:** Re-executing some or all Test Cases in each build. Whenever new build is release, we have to verify that fixes or changes or added new features may have not introduced new errors. Regression testing is done in 3 situations: **a)** After fixing the bugs **b)** If a new change request came from client **c)** When environment changes

**Inclusions:**

1. Test cases for modules impacted by code changes.

2. Core features of the application.

3. High-priority defects fixed in recent builds.

4. Tests for integrations with external systems.

**@. Smoke and Sanity Testing**

**Smoke testing/BVT/TAT:** It is a first level of testing on any newly released build to check the **most basic** and **crucial functionalities** of the application is working fine. Smoke testing is done once the build is ready from the development environment before releasing it to the testing environment. The resultant of this testing is used to decide whether the build is ready for further testing or not. It is done by developers and testers. It is typically scripted, although it can sometimes be unscripted. **Ex:** Can u open the app? **/** Can u log in? **/** Is the homepage loading?

**Sanity testing** is done once the build is released to the testing environment to verify the **new functionalities** and **bug fixes** to ensure the system works as expected. It is done when the build is stable. The main purpose of sanity test is not to break the test execution and to make sure that end-user requirements are met or not. It is done by testers only. Mostly it is scripted or not scripted depending upon the organizations. **Ex:** A bug was fixed in the “Add to Cart” button 🡪 Sanity testing to check: Is the “Add to cart” working? **/** Did it break the Cart page?

Smoke: Basic check before full testing.

Sanity: After minor changes.

Regression: Ensure no break due to new code.

Re-testing: Recheck failed TCs after bug fix.

**@. Performance, Load and Stress Testing**

**Performance testing:** Performance testing is a non-functional testing. It is a process of checking whether the application is maintaining the targeted response time for the specific load along with speed, responsiveness and stability of an application under different conditions. These are 2 types those are: Load testing and Stress testing

**Load Testing:** Load means number of users. The process of increasing the load within the limits (for ex if application limit is 100 users, we will test from 1, 10, …. Up to 99) on the application and performance test engineer will check the response time of the application is known as **Load Testing**.

**Stress Testing:** The process of increasing the load beyond the limits (for ex if application limit is 100 users, we will test from 101, 110….) on the application and performance test engineer will check the response time of the application and at what point (load & time) the application is taking **more time** to process the request is known as **Stress Testing.**

**@. Usability testing / Wt aspects will check**

Usability means easy to use and learnability of a human made objects. Here Test Engineer will check how easy to use (user-friendly), efficient and also whether the application is maintaining user friendliness with the End users or not is known as **Usability Testing**. It is a type of non-functional testing. **Ex TCs:** Ensure navigation menus are accessible and actions are clear. Aspects to Check:

1. Navigation: Ensure menus, links, and buttons are easy to find and use.

2. Content Clarity: Verify labels, error messages, and instructions are clear.

3. Consistency: Check uniformity in font styles, colors, and layout.

4. Accessibility: Ensure the application is usable for differently-abled users (e.g., screen readers, keyboard navigation).

5. Performance: Validate response times for user actions.

**N:** **Accessibility testing** is the sub set of Usability testing here will validate the amount of ease to use application. It ensures that the applications comply with accessibility standards and provide equal access to users with various impairments, including visual, auditory, motor, and cognitive disabilities.

**Ex for Accessibility Testing:** Testing for screen reader compatibility and keyboard navigation.

**@. AUT** stands for Application Under Test. The application which we are testing is known as AUT.

**@. Showstopper Defect** ------- A defect which is not permitting to continue further testing is called **SSD**.

**@. TC Design techniques / ECP, BVA and Error guessing**

**1. Error guessing:** Based on our testing experience we know that particular thing is not working properly this will know only if we already worked on such type. It’s nothing but guessing defects which was there in the software.

**Ex:** Entering special characters in a text field

**2. ECP:** ECP stands for Equivalence Class Partition. This technique is used to deriving the Test Data where it is not possible to develop the Test Data with all the positive and negative flows i.e. valid and invalid partitions. Equivalence class Partition divides the test data of a system into equal and valid partitions where TCs are designed to test only one value from each partition, assuming other will behave similarly. By using ECP technique**:**

* + First divide the test data equally into valid and invalid partitions.
  + Test the field with valid data so that the field should accept it.
  + Test the field with invalid data so that the field should not accept it.

**Ex:** Assume Age field accepts only between 18 to 60 (valid: 18-60 and invalid: <18 & >60)

**3. BVA:** BVA stands for Boundary Value Analysis. This technique is used to deriving the Test Data to test the values at boundaries have a higher probability of detecting error for ex, when we are planning to test the range like 0-100 or 0-1000 or 1Lac to 2Lac etc., It is very difficult to test the field with all the values and even to write the Test data for all the positive and negative flows. The following are the reasons to introduce BVA:

* + Verify faults at near boundaries
  + Test values on both sides of boundaries
  + Good place to look for faults

We will use BVA as**:** test the field with min, min+1, middle, max-1 and max values. If the field is accepting, then we can conclude that the test has been passed.

And test the field with min-1 and max+1 value. If the field is not accepting, then we can conclude that it is accepting only the range.

**Ex:** If a valid range is 1-10 (test values: 0, 1, 2, 9, 10, 11)

**4. Decision table:** It is used for complex business logic. It is used to test the system behavior for different input combinations and their corresponding system behavior (Output) are captured in tabular form. It is also called as Cause-effect table where cause and effects are captured for better test coverage.

**Ex:** Inputs vs actions in a table

**@. Client Server Application testing and Web Application testing**

In **Client Server Application testing**, there are two different components to test. Those are application is loaded on the server machine and application .exe is installed on every client machine. Testing is done broadly in categories like GUI, func, non-func on both sides, Client-Server interaction and backend testing. Most of the CSA’s are Intranet networks.

In **Web Application testing**, the application is loaded on the server machine and no application .exe is installed on the client machine. Web Applications can test on different browsers and OS platforms. It is tested mainly for browser and OS compatibility, GUI, functional, non-functional and backend testing, etc.

**Intranet network:** An **intranet network** is a **private network** that uses Internet Protocol technologies to securely share any part of an organization's information or system operations within that organization.

**@. Use Case, Test Case and Test Scenario** --wdt

**Use Case** is a graphical representation of actions, those describes the behaviour of a system to do a particular task.

**Test Case** consists of set of test data, TC ID (unique identifier of TC), TC name, objective/Description, test steps, Test type (Regression / sanity), input values, preconditions (setup required before execution), postconditions, Actual results, Expected Results, severity, comments and status (pass/fail) are developed to cover certain test Condition. A Test Case is executed to verify the functionality of application using this a test engineer can determine whether a software product is functioning as per the client requirements or not.

**Test Scenario:** Scenario means a flow or story. Requirement will be divided into multiple scenarios to test the application with all possible flows. A Scenario is defined to identify all possible areas to be tested. It gives high-level idea on what we need to test. Which describes End-to-End functionality to be tested.

|  |  |
| --- | --- |
| **Test Scenario** | **Test Case** |
| It is High-level; focuses on more "what to test" **than** "how to test". | It is Low-level; focuses on both "what to test" **and** "how to test.". |
| Test scenarios are derived from test artefacts like FRS. | Test Case is mostly derived from Test Scenarios. Multiple Test Cases can be derived from a single Test Scenario |
| Comparatively less time and less resources are required for creating and documentation of Scenarios. | More time and more resources are required for creating, documentation and execution of Test Cases. |

**For Ex:**

**Test Scenario:** Verify login functionality for valid user

**Test Case:** Test Login with valid credentials, invalid credentials, empty fields…

**@. Positive and Negative TCs covering functional, UI, and validation aspects.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Case Description** | **Test Steps** | **Test Data** | **Expected Result** |
| TC\_Login\_01 | Login with valid credentials | 1. Enter valid email & pass 2. Click Login | Email: [user@example.com](mailto:user@example.com) Password: 123456 | Redirect to Dashboard |
| TC\_Login\_02 | Login with invalid password | Enter valid email, wrong password | [user@example.com/](mailto:user@example.com/) wrong pass | Show error: "Invalid credentials" |
| TC\_Login\_03 | Login with invalid email format | Enter email without @. Ex: Email: “user.com” | Email: user.com/any | Show validation error |
| TC\_Login\_04 | Login with blank email and password | Leave both fields empty | (blank) | Show validation message: "Email and password required" |
| TC\_Login\_05 | Login with SQL injection | Enter: admin' OR '1'='1 | SQL code as input | Show error/prevent login |
| TC\_Login\_06 | Check "Remember Me" functionality | 1. Check box  2. Login  3. Close & reopen browser | Valid login | Email stays saved |
| TC\_Login\_07 | Forgot Password link redirects correctly | Click "Forgot Password" link | - | Redirect to Reset page |
| TC\_Login\_08 | Password field is masked | Type in password field | any | Characters should be masked (....) |
| TC\_Login\_09 | Login button disabled if fields are blank | Leave email and password blank | - | Login button remains disabled |
| TC\_Login\_10 | Error disappears after correct login | 1. Try invalid login 2. Try correct credentials | - | Error message clears & user logs in |

**@. QA and QC ----** ippcge

|  |  |
| --- | --- |
| **QA** (Tester) | **QC** |
| Identify the standards and guidelines | Implement the standards and guidelines |
| Process oriented to ensure the quality | Product oriented |
| Preventing problem | Detecting problem |
| Continuous improvement | Final check point before delivery |
| Goal of QA is to improve development and testing processes so that defects will not arise when the product is being developed | Goal of QC is to Detecting defects after a product is developed and before it’s released |
| Verification is an example of QA | Validation is an example of QC |

**@. Levels of testing in SDLC / Software testing methodologies**

Unit Testing, Integration Testing, System Testing and Acceptance Testing.

**@. Environments in software industry**

1. Dev Environment: Development team will work on this environment

2. QA/ Testing Environment: Testing team perform the executions on this environment.

3. Staging/ Lab Environment: After testing, the build kept on staging/lab environment. Here UAT will done before releasing the software.

4. Production Environment: Here real user sees uses. After all the testing activity is done, that build is deployed on production environment for the actual users.

**@. Unit Testing / Component Testing**

Validating individual components or modules within the system is called **Unit** or **Component Testing**. It is performed by developers. Deliverable of this Unit Testing is: software unit is ready for testing with other system component. Unit testing tools are Junit and TestNG.

**@. Integration testing? Types of Integration testing**

Integration means combining once all the modules are developed by the developers, the developer will combine all the modules in hierarchical order is known as **Integration**. **Integration Testing** will test the interaction between integrated modules or components to check how different components of a program work together this can achieve by verifying the data flow between one module to other module is navigating properly or not or how different modules or components work together (i.e. data communication and co-existence relation between modules) is called “**Integration testing**”. Ex: Testing APIs or communication between modules. It is mostly performed by developers and some cases involve testers also. There are 4 diff types of IT’s are there those are:

**Big bang Int Testing:** Here testing will start after integration of all the modules.

**Top-down Int Testing:** Here testing will start from the top-level modules to lower-level module in the hierarchy.

**Bottom-up Int Testing:** Here testing will start from the lower-level modules to top-level module in the hierarchy.

**Hybrid Int Testing:** Here testing will start from the middle layer and testing is carried out in both the direction.

**@. System testing**

System testing includes both functional & non-functional tastings. Here will check overall functionality and performance of a complete and fully integrated software system against the specified requirements to identify bugs. It tests whether the system meets the client specified requirements or not and also if it is suitable for delivery to the end-users. This type of testing is performed after the Integration testing and before the Acceptance testing.

**N:** **Response time** is the time taken between the request and response. System testing is performed by the testers.

**@. Acceptance testing and UAT**

Both are closely related, but they are not exactly the same. Both are usually performed by end-users or clients. It may be performed before or after deployment based upon organization standards.

Acceptance testing is performed to determine whether the system meets the end-user requirements (Verifying functionality against specified requirements) or not, to establish the confidence on the application, to check acceptance criteria, and is ready for deployment or not. It is typically conducted at the final stage of the development process, involving end-users, clients, or stakeholders to verify that the software works as intended in real-world scenarios.

In real words, Acceptance testing means testing the application in front of customer before the product release. In this case we have explain the functionality by giving the demo with real time scenarios whether actually fulfills the end-user requirements.

Different types of Acceptance testing: UAT

**1. UAT** stands for User Acceptance Testing and it is also known as **Client Acceptance Testing**. UAT is done to check whether the application meets the user needs (It focuses on real-world scenarios to validate the software meets their needs), along with expectations and business requirements, it is usually performed by end-users or clients. Basically, UAT will conduct when the application is before going to live/production.

As per software testing standards once the application is stable, the client will provide some UA Test Cases to the testing team. The test engineer has to execute all the Test Cases. If all the Test Cases are passed, then only the client will accept the build and that build will be deliver to client. UAT is two types those are:

**1a. Alpha Testing:** UA Test Cases will be executed in Test environment by internal testing team before releasing the product to customers.

**1b. Beta Testing** is conducted by end-users / client’s team / third-party TE’s in their real-time environment to get feedback before releasing the product to customers.

**N:** Before proceeding build to UAT we make sure that the business flow is working perfectly or not by mapping the req’s

**N:** UAT Test Cases mostly Business-oriented scenarios, E2E Testing Scenario and on usability & functionality

**@. System testing and Acceptance testing**

System testing is performed to test E2E functionality of the software. It can be performed by developers and testers.

Acceptance testing is performed to test whether the software is confirming the specified and user requirements or not. It can be performed by testers, stakeholders and customers.

**@. In the software defects, how Priority and Severity defined / priority say in Test Case document**

**Severity** defines the impact of a defect on the functionality. Responsibility of testing team. Each and every defect will have the severity and classification could be.

**CRITICAL / Fatal Defect / Blocker Defect:** A major functionality is failed so testing cannot continue **Ex:** login func fails.

**MAJOR:** A major issue in functionality of system component is not working but testing can continue **Ex:** Internal links or buttons are not working.

**MODERATE:** A minor issue in functionality of system component is not working that imposes some loss of functionality but testing can proceed without interruption **Ex:** Cosmetic issues like slight deviation in Button and Edit box positions.

**MINIMAL:** Under this will comes usability and UI issues. **Ex:** cosmetic glitch, spelling mistake, minor UI miss alignment (like slight change in image / logo colors).

**Priority** defines how soon the defect needs to be fix. Responsibility of developing team. Each and every bug will have priority and it describes the importance of the Test Cases. Based on the severity the priority will be assigned. Priority defines in which order the defect has been fixed by the developer. The classification could be P1-HIGH, P2-MEDIUM, P3-LOW.

**P1 (HIGH):** The Test Case describes about the main functionality.

**P2 (MEDIUM):** The Test Case describes about the field level.

**P3 (LOW):** All the GUI issues.

**@. Test Plan / What will write in Test Plan**

Software Test Plan is a strategic document which describes how to perform the testing in effective and efficient manner and also describes the scope, approach, resources planning, schedule of testing activities, identifies test items, features to be tested, testing tasks who will be doing, any risks, etc. It is prepared by the Test Lead / Test Manager.

**N: Test Execution** is a process of running Test Cases, comparing the expected and actual results and logging the results.

**@. Installation testing**

Install and uninstall the application using various operating systems and browsers (preferably it’s based on user env.)

**@. Compatibility Testing**

Testing the application in multiple environments like multiple browsers, multiple devices and multiple operating systems and check whether the application is working as expected in all the environments or not is known as **Compatibility Testing**. This kind of testing is very useful for Internet based applications like Facebook, G-mail, etc. It is a type of non-functional testing.

**@. End-to-End testing and walk through h to perform end-to-end testing in an enterprise environment?**

E2E testing checks the entire application from start to end. It verifies that all components of a system can run under real-world scenarios. The goal is to find bugs that appear when all parts of the system work together.

**Ex:** Place an order -> Payment processing -> Generate invoice -> Email confirmation

Scenario: Testing a hospital management system: Patient Registration Appointment Billing Reports.

Approach:

- Start with real-life test data (new patient).

-Verify workflow: patient registration appointment booking consultation billing report generation.

- Involve APIs (fetch doctor availability) and database validation (billing details stored correctly).

- Use different roles (Admin, Doctor, Receptionist) to test access control.

- Record results and screenshots for demo to stakeholders.

**N:** **Test data** consists of inputs used during testing to verify the functionality of application.

**@. Localization Testing**

Test engineer will test the application in all the local languages like Hindi, Bengali, Telugu, Tamil etc., and check that whether the application is working as expected or not is known as **Localization Testing** and it is also called as **L10N** testing, here 10 is the num of letters b/n **L** and **n** in the Localization word. @**Ex:** google.com avl in all the Local lang’s.

**@. Internationalization / Globalization Testing**

Test Engineer will test the application in all the international languages like Spanish, Chinese, Japanese etc. and check that whether the application is working as expected or not is known as **Internationalization testing**. and it is also called as **I18N** testing, and here 18 is the number of letters between **I** and **n** in the Internationalizationword. @**Ex:** gmail.com available in all the Internationalization languages.

**@. Monkey Testing (or Gorilla Testing)**

Monkey Testing means testing the application by performing abnormal actions like: continuously click on any field for a longer period of time and check that whether the application is throwing any error or not. It’s nothing but enter the invalid data and check whether the application is crashing or not.

**@. Bug which is High Severity and Low priority**

High severity bugs will be having high priority. But if the bug is not related to the current release, then the priority of the bug will become low priority. The development lead having the permissions to change the priority.

**Ex:** If some crash is happening in the application but that particular module release to customer after some time.

**@. Bug which is Low Severity and High priority**

Low severity bugs will be having low priority. But if the bug is related to the current release, then the priority of the bug will become high priority. The development lead having the permissions to change the priority.

**Ex:** Slight change in logo colour or spelling mistake in About / Help page.

**@. Low Severity and Low priority** ----A spelling mistake in a page and that page is not frequently navigating by users.

**@. High priority and High severity** ---- Issue with login functionality.

**@. Endurance Testing (or Soak Testing) and Volume Testing**

Both will comes under Performance Testing.

|  |  |
| --- | --- |
| **Endurance Testing (or Soak Testing)** | **Volume Testing** |
| Testing the application with minimum load for 1 to 2hrs and check that whether the application is maintaining the targeted Response time or not is known as **Soak Testing**. | Testing the application with maximum load for 1 to 2hrs and check that whether the application is maintaining the targeted response time or not is known as **Volume Testing.** |

**@. Defect Age**

**Defect Age** is the difference in time between the current date (if the defect is still open) or Defect fixed date to the Defect Detection Date **for Ex**: If a defect was detected on 01/01/2009 10:00:00 AM and closed on 01/04/2009 12:00:00 PM, the Defect Age is 74 hours.**@@@**Defect Age = Current Date (or Defect Fixed Date) – Defect Detection Date

**Masked defect** is commonly known as **defect** that hides the other **defects** in the system so that the **defect** is not detected at a given point of time.

**@. Bug Release and Bug Leakage**

**Bug Release:** When a specific version of the software is released into market with some known bugs is known as Bug Release. These will be fixed in the later versions. These are low priority Bugs and this will mention in the Software Release Note documents and sharing with end-users.

**Bug Leakage:** When the tested software is released into the market and the end-user finds bugs in it is known as Bug Leakage. During the test execution phase testing team missed to test these bugs. Bug Leakage will occur due to tester followed the wrong requirements, Developer developed the code with wrong requirements, Reviewing the TCs was not proper, client requirements not updated correctly. Generally, we perform UAT before release application into production to find defects which are not identified or which are moved to next phase. Bug Leakage is not part of Software Release Note documents because these bugs are missed by tester.

**@. GUI and UI Testing**

UI testing is mostly GUI testing. GUI testing is a subset of UI testing. **GUI** Testing will focus on the graphical user interface (GUI) by testing elements such as icons, buttons, menus and text boxes are functionating correctly and also the logo colours. **UI** testing will test the user interface (graphical, voice or touch) through which users interact with a website or an application.

**GUI Testing Test Cases:**  On visual elements (Check alignment, front size and colours), Usability (Validate labels, navigation and button clicks), Error messages (Ensure appropriate message for invalid inputs), Cross-browser testing (Verify GUI on different browsers).

**Ex:** Verify that the ‘submit’ button is visible and clickable on the login page.

**@. Defect Density ----** Ref for Ex: https://economictimes.indiatimes.com/definition/defect-density

Defect Density is the number of confirmed bugs (Defect count) in a software application or a module during the period of development, divided by the size of the software. It enables to decide if a piece of software is ready to release. Defect density is counted per thousand lines of code, also known as **KLOC**. **@@**DD = Defect count/ size of the release. Size of the release can be measured in terms of LOC.

**Ex:** Module 1 = 10bugs, Module 2 = 20bugs, Module 3 = 10bugs ==> Total bugs = 60

LOC in each module: Module 1 = 1000loc, Module 2 = 1500loc, Module 3 = 500loc ==> Total loc = 3000

**@. Latent defect ----** Defect which is identified in later stage due to hidden problem.

**Test Set** is a group of tests derived from the internal structure of a component in order to achieve 100% coverage of specific criteria. For Ex: **Tests** that focus on a certain part of the application such as GUI or Database and **Tests** that belong to a certain task such as Regression or Sanity.

**@. SDLC and STLC ----** rfp ntgp

|  |  |
| --- | --- |
| **SDLC** | **STLC** |
| SDLC is mainly related to software development. | STLC is mainly related to Software Testing. |
| It focuses on both Development and Testing activities. | It focuses only on Testing the software application. |
| SDLC involves some phases like req capturing, analysis..... | STLC involves some phases like Requirement Analysis, Test Plan, Test Case Design, Test Environment setup, Test Execution, Defect Reporting, Test Closure RTM..... |
| In SDLC, more number of developers are required for the whole process. | In STLC, comparing to developers less number of testers are required for the whole process. |
| In SDLC, Development team makes the plans and decisions. | In STLC, Testing team makes the plans and decisions. |
| Goal of SDLC is to developing a good quality software. | Goal of STLC is to testing the s/w product more effectively. |
| SDLC phases are completed before the STLC phases. | STLC phases are performed after SDLC phases. |

**@. Iterative process**

An iterative process is a process for calculating the desired result (expected results) by repeated cycle of operations. If number of iterations increase, it should come closer to desired result.

**@. Static and Dynamic testing----** cop ctc cv LT – Longer time

|  |  |
| --- | --- |
| ST checks the requirements, design documents and reviewing code for errors without execution | DT checks the defects and functionality of a software by executing the application. |
| Objective is to discover verity of bugs and to prevent the bugs in software without executing the code | Objective is to find and fix the bugs in software by executing the code |
| It is performed at early stage of software development | It is performed later stage of software development |
| ST is less cost ---- cop ctc cv | DT is high cost |
| ST generally takes shorter time | DT generally takes LT (bcz it involves running several TCs) |
| ST involves checklist for testing process | DT involves TCs for testing process |
| It involves different activities like requirement review, design review, code review, TP review and TC review. | It involves different activities like functional and non-functional testing |
| It is a verification process | It is a validation process |

**@. WBT, BBT and Grey Box Testing ----** icp hmsc

|  |  |
| --- | --- |
| **Blackbox Testing (Testers)** | **Whitebox Testing (Developers)** |
| Blackbox Testing is a [Software testing method](https://www.softwaretestingclass.com/what-is-software-testing/) which is used to test the application without knowing the internal structure of code. ---- icp hmsc | Whitebox Testing is a software testing method which is used to test the application by knowing the internal structure of code. |
| This type of testing is carried out by Testers. | This type of testing is carried out by Developers. |
| Implementation / Programming knowledge is not required to carry out Blackbox Testing. | Implementation / Programming knowledge is required to carry out Whitebox Testing. |
| Testing is applicable on higher levels of testing like System testing, Acceptance testing. | Testing is applicable on lower level of testing like Unit testing, Integration testing. |
| Blackbox testing means Functional Testing | Whitebox testing means Structural / Interior Testing. |
| BBT can be started based on Req Specification doc’s | WBT can be started based on Detailed Design doc’s. |
| Functional, Behaviour, Close box testing is carried out under Blackbox testing, so there is no programming knowledge required. ----- fbc | Path, Logic, Loop, Code, statement, decision coverage, techniques are carried out under WBT, so compulsory programming knowledge required. ----- plcs  WBT techniques: |

**Grey Box Testing** is a software testing technique that combines elements of both White Box Testing (where the tester has knowledge of the internal workings of the system) and Black Box Testing (where the tester only focuses on the system's external behaviour) approaches. Here tester have partial knowledge of the internal working (structure and logic) of the system and also focus on functional specifications. It is used by both developers and testers.

**Entry Criteria** define the **pre-requisite** items that must be complete before **testing can begin**. For Ex: Test environment is ready, TCs are reviewed, Requirements are finalized.

**Exit Criteria** defines the **post-requisite** items that must be complete before **testing can be concluded**. For Ex: All critical TCs are passed, No high severity bugs open, Test summery is completed.

**@. Testing techniques in software testing**

**1. Under Black box testing, testing techniques:** ECP, BVA, State transition testing and Decision table testing.

**2. Under White box testing, testing techniques:**

Statement coverage: Ensuring all code statements are executed

Branch coverage: Testing all decision branches in the code

Path coverage: Testing all possible execution paths

Condition coverage: Testing all logical conditions

Loop testing: Focusing on loop constructs and iterations

**3. Under Grey box testing, testing techniques:** Integration testing, penetration testing, API testing and security testing.

**4. Under Functional testing, testing techniques:** Unit, integration, system, UAT, smoke, sanity and regression testings.

**3. Under Non- functional testing, testing techniques:** Performance, load, stress, security, usability, compatibility and scalability testing.

**@. Dynamic Testing**

Testing the dynamic behaviour of code / elements is called **Dynamic Testing** or **Dynamic** **Analysis**. It refers the physical response from the system to variables are not constant and these are change with time.

**@. Positive and Negative Testing**

**Positive Testing:** Testing the application by providing the **valid data** **as input and** checks whether an application behaves as expected with positive inputs. This test is done to check the application response if it is supposed to do. For Ex.

**Negative Testing: Testing the application** by providing the **invalid data as input** and checks whether an application behaves as expected with the negative inputs. This test is done to check the appl response if it is supposed to do. For Ex.

[](https://www.guru99.com/images/p1(1).png)[](https://www.guru99.com/images/p2(1).png)

**@. Test Cases you have written in a project -------** Around 250 to 300 Test cases I written for my modules

**@. Techniques are followed to write test cases -----** In BBT we use techniques like ECP, BVA and Error Guessing

**@. Domain knowledge ----** Banking,Capital market, ERP,E-comers, Finance, Health care, Insurance, Telecom & Travel

**@. Requirement Traceability Matrix (RTM)**

It captures all the requirements proposed by the client or software development team and their traceability in a single document that delivered at the conclusion of SDLC. It maps all the requirements with the Test Cases to check if any functionality was not checked during Software testing and also ensure that all the requirements are covered by TCs and helps to track test coverage. Components in RTM**:** Req ID, TSs, TC ID/Name, TC Description, TC with status. RTM helps to ensure 100% test coverage.

**@. Test coverage / Types of Test coverage techniques / To ensure Test Coverage**

Test coverage measures the Test Cases cover the applications functionality. To ensure test coverage, we can use variety of techniques including Test Case design, Test Automation, and Manual testing. This helps to identify untested code and ensures critical areas of the application are tested. Test coverage techniques are:

Statement/block coverage: measures how many stmts in the source code have been successfully executed and tested.

Decision/branch coverage: measures the how many decision control structures were successfully executed and tested.

Path coverage: ensures that the tests are conducted on every possible route through a section of the code.

Function coverage: how many functions in the source code have been executed and tested at least once.

(or) in general

1. Understand Requirements: Thoroughly analyse requirements and create a traceability matrix to map them to TCs.

2. Categorize Test Scenarios: Cover all functional, non-functional, edge cases, and integration points.

3. Include Positive and Negative Tests: Test normal workflows as well as scenarios with invalid or unexpected inputs.

4. Use Equivalence Partitioning and Boundary Value Analysis: Divide test inputs into valid and invalid partitions.

5. Perform Exploratory Testing: Execute unscripted tests to identify gaps in predefined test cases.

6. Review by Peers: Conduct peer reviews of test cases to ensure no requirements are missed.

**N:** Test Case coverage: Map Test Cases to requirements using traceability matrices.

**@. Software testing metrics / Test Metrics**

Software Test Metrics are quantitative measures used to evaluate the effectiveness of the testing process such as defect density or Test Case execution rate.

**Some important testing metrics are:**

* Total number of defects found, ordered by their severity.
* Total number of bugs fixed.
* Bugs find and fix rate over time.
* The average time taken by a bug since it’s found and fix.
* Total time spent on new feature development vs time spent on resolving bugs and failure.
* Number of outstanding bugs before release.
* Bgs/failure reported by customers vs those found by the testers.

**@. H will provide priority for a Test Case / Advantage of priority**

We can assign priority to a Test Case based on importance of the functionality w. r. to client business needs. DL/TL can provide priority to a test case

**Adv:** Based on priority order the developer can fix the bugs and we can select Test Cases for execution when there is no time to execute all the Test Cases.

**@. Test Cases will write / prepare in a day**

It always depends on the project and requirements. But an average we can prepare 5 to 6 Test Cases per day. It means we are taking around 50 mins to prepare 1 Test Case it’s include review and analyse the requirement and to update it in Test Case template.

**@. Approach to write Test Cases**

* + First Study the Requirement Doc or Use Cases
  + Prepare Test Scenarios
  + Develop Test Cases for the particular scenarios

**@. Test Cases can be executed manually per day**

It’s always depending on the application and complexity of Test Cases but also depends upon below 2 approaches.

1. If we written the TCs then we know the functional knowledge of the appln. So, we can execute 10 to 11 TCs per day.

2. If we not written the TCs, for example client has given the TCs. In this situation we can execute 7 to 8 TCs per day.

**@. Responsibilities of TE -----** ufs cpe iLr tsm aerc

* + Understanding the requirements and functional specifications of the applications
  + follow the standards and guidelines of project while testing the application
  + Identify the require Test Scenarios for project
  + Design and prepare Test Cases and Test Data for identified scenarios
  + Participate in peer reviews
  + Execute the Test Cases to validate the application without any issue
  + Identify the defects and maintain the bug reports for defect tracking
  + Logging and tracking the defect
  + Re-test the fixed defects on previous builds ----- ufs cpe iLr tsm aerc
  + Perform various types of testing assigned by TL (functional, usability, system, UI, compatibility, Regression, etc.)
  + Prepare daily status report, daily defect report, weekly status report, monthly status report, Re-testing report and any other assigned tasks.
  + Participate in regular team meetings conducted by Team Lead
  + Preparing automation test scripts for Re-testing and Regression Testing
  + Based on end-user perspective providing enhancements (improvements) to the project
  + Provide recommendations whether the application is ready for production or not
  + Maintain good communication with team members

**@. Status reporting in organisation**

In my organisation all Test Engineers have to report status to Test Lead but sometimes may also report to Project Manager that depends on resource availability in the project so, team may report to TL or PM

**@. To share or sent a file (size-50MB) in the organization**

If the file size is more, then we can’t send it through a mail so, in my organizations we having “Common Repository”. so, copy the file and paste it in the Common Repository so any one can access it.

**@. STLC phases**

Software testing is a process that helps to make quality software and to find defects in the software. STLC process using as a testing process in my organisation. There are seven steps involved in this testing process those are:

**1) Requirement Analysis:** Testing team gather SRD and plan all the necessary tests as per requirement.

**2) Test Planning:** Here will write the test plan, which includes the activities that will be carried out throughout the testing process. Product that will deliver as part of Test plan. Test plan will prepare by the Test Lead or Test Manager.

**3) Test monitoring & control:** Here will compare the progress with what is stated in the Test Plan. Is there any deviation between what was planned and what is actually done, we perform control activities to resolve this deviation. In test monitoring & control we may write a “test progress report” that is used by stakeholders to monitor the progress of testing and decide whether the software is ready to be delivered or not.

**4) Test Env Setup** is done by the senior Tester and it requires a group of essential software and hardware components.

**5) Test Analysis:** Preparing Test Scenarios by reviewing the requirements, user stories, or design doc.

**6) Test Design:** Preparing the Test Cases from Test Scenarios. Test Cases are more detailed than Test Scenarios and take more time to write. In some projects will skip the Test Scenario writing step and just writing Test Cases.

* + - 1. We can design Test Cases with the help of Use Case and Test Scenario’s
      2. Using Test Case design techniques like ECP, BVA and Error Guessing technique
      3. Using Test Case Template

**7) Test Implementation:** Here will ask ourselves “Are we ready to execute our tests?”, and if the answer is no, then begin to work on turning this “No” into “Yes”. Test implementation includes many activities like setting up the test environment to execute our tests.

**9) Test Execution:** Here will execute our TCs. When defects are found, we write defect reports and report them to developers.

**8) Test Completion:** Here will make sure that all defects are fixed and write test summary reports which summarize all testing activities happened during the testing process.

**Software Release Note Document (SRN)** describes about the build details like Build version number, build location, Modules changed, Installation steps, known issues and Bug fixes.

**Deployment Document (DD)** describes how to install/deploy the appln before testing and how-to setup the Test Env.

**Defect Reporting:** Some of the components will require for defect reporting those are:

-Defect Template

-Knowledge to assign Severity and Priority

-Bug Life Cycle (BLC)

**@. Bug lifecycle / stages of Bug lifecycle**

It is a step by step process to resolve the bug. The following are the stages in Bug lifecycle.

**New:** A bug is identified and reported for the first time.

**Assigned:** The bug is identified, approved by the testing lead, posted by the tester, the test lead or QA manager assigns the bug to the developer for resolution.

**Open:** The assigned developer begins analysing the defect.

**Fixed:** The developer resolves the issue and implements a fix.

**Re-test:** The tester re-tests the bug to confirm the fix works as expected.

**Verified:** The bug is confirmed to be fixed and no longer present.

**Closed:** The bug is considered completely resolved and is no longer active.

**Optional Stages in the Bug Lifecycle:**

**Reopened:** The bug is reopened if it reappears after being fixed or closed.

**Deferred:** The defect is acknowledged but postponed for future releases.

**Duplicate:** The bug is marked as a duplicate of an existing one.

**Rejected/Not a Bug:** The defect is considered as invalid.

**@. Wn do prefer Stub and Driver, h it will work**

A Stub and Driver simulates module which has all the capabilities of the unavailable module.

**Stubs** are developed by the developers to use them in place of lower-level modules are needed but are currently unavailable (due to the respective modules aren’t developed, missing / under construction in developing stage) when Top-down integration testing is performed on modules.

**Drivers** are developed by the developers to use them in place of higher / lower-level modules are needed but are currently unavailable (due to the respective modules aren’t developed, missing / under construction in developing stage) when Bottom-up integration testing is performed modules.

**@. Agile Methodology / Agile process / H Agile Methodology different from traditional methodologies**

AGILE methodology promotes **continuous iteration** of development & testing throughout the Software Development Life Cycle of the project. In this both development and testing activities are concurrent unlike the Waterfall model.

* It is an iterative and incremental framework to develop applications ------ ircc pst hmet
* This model allows rapid changes in requirements
* Customer satisfaction by rapid delivery of working software frequently (weeks rather than months)
* Close and daily communication between business professionals and developers
* In Agile model project is divided into various sprints
* Each sprint contains highest-priority requirements
* A time period for sprint is typically 2 to 4 weeks
* Here will have scrum teams, scrum master and product owner
* In agile model, will conduct daily scrum meeting with team members to share status and potential issues
* Each sprint is released to customers
* It can be used for time-critical applications

The agile model software development is mainly containing two core values those are: --- ir

* Individual and team interactions over process.
* Responding to change over following a plan.

**N:** In Agile methodology we have to follow: how to follow scrum and how to implement them.

**@. ‘3 Amigos’ in Agile** -- A meeting with Developer, Tester, and Product owner to discuss stories before implementation

**@. Verification and Validation**

Software testing is a combination of both Verification and Validation. These are the basic ingredients of SQA activities

**Verification** is a process to check Are we **developing** system right or not (It involves process like review (Requirement review, design, code, TC review), walk through and inspection)

Hence, Verification techniques can be used in Peer review, code walkthrough, Inspection, etc.

**Validation** is a process to check Are we **developed** system right or not (It involves actual testing (Functional, system, integration, Ad-hoc, smoke, Regression))

Hence, Validation techniques can be used in Unit testing, Integration testing, System testing, etc.

**@. Meeting point in V-model ---- Coding** is the meeting point in V-model

**@. Activity will be perf. to check project is acceptable or not to develop** -- Feasibility Study or Kick off meeting

**@. Optimal Testing**

Validating the appl with **best possible** Scenarios (best permutations and combinations) by using ECP and BVA technics.

**@. Exhaustive Testing**

Validating the application with **all possible** Scenarios (all permutation and combination).

**Ex:** Let Consider ‘UserName’ field accepts 4-16 char’s with alphanumeric. There are 26\*26\*26\*26 input permutations for alphabets only with 4 characters so there are more input combinations so it is time consuming and expensive.

**@. Dynamic changes in Req’s --** If client continuously sending changes in requirements while developing the application

**@. When defects will arise while developing appln ------** mistakes in Design **&&** mistakes in Coding **&&** incorrect req’s

**@. Wn do think testing activities should start in software development process why**

Testing activities should start parallelly with development activities because early stages identified defects will take less time and less cost to solve those defects compare to later stages identified defects.

**@. Prototype / h it will helpful for TE**

A Prototype is a model / blueprint of the application. Based on approved SRS, software architect will design model of the project and it is helpful to foresee the future implementation of application.

**@. Skill set required for a TE**

-knowledge on Manual testing

-knowledge on any one of the automation tools like selenium, UFT, etc

-knowledge on any one of the automation tools supporting programming language (i.e., coding knowledge)

-knowledge on any one of the defect reporting tools like QC, Bugzilla, JIRA, etc

-knowledge on database (how to retrieve data from a database)

-Basic knowledge on different Domain applications

**@. Y do perform Software Testing / Advantages of Software Testing -------** rmb

-to deliver reliable product to customer

-to reduce maintenance cost of a project

-to deliver bug free application

**@. Burndown chart in Agile**

A **Burndown chart** is a graphical representation of **work left to do** versus **time**. The o**u**tstanding work (or backlog) is represented on the **v**ertical axis along with time on horizontal axis. Basically, it is a clear representation about how much work was completed and how much work is pending by the team.

**@. Test Plan and Test Strategy ----** dpc cid

|  |  |
| --- | --- |
| **Test Plan** | **Test Strategy** |
| It is derived from SRS | It is derived from BRS |
| It is prepared by the Test Lead or Test Manager. | It is prepared by the Project Manager or Business Analyst. |
| Test Plan id, test techniques, testing tasks, features to be tested, features pass or fail criteria, test deliverables, responsibilities, T schedule, etc. are the components in TP. | Objectives and scope, documentation formats, testing processes, team reporting structure, client communication strategy, etc. are the components in Test Strategy. |
| If there is a new feature or a change in the requirement need to update the Test Plan document. | Test strategy maintains the standards while preparing the document so it is also called as **Static Document**. |
| We can prepare the Test Plan individually for diff projects | We can use Test Strategy to multiple projects. |
| It describes what to test, how to test, who will test and when to test. | It describes what type of testing technique can follow and which module to test. |
| Test plan is a detailed doc specific to a particular project. | Test strategy is a high-level doc that outlines the general approach and principles for testing within the organization. |

**@. Test Deliverables**

Test Deliverables are the artifacts or documents that are created and shared during the testing lifecycle. These deliverables help to track the progress, coverage, and quality of the testing process and provide stakeholders with insights into the system's reliability and readiness for release.

**Types of Test Deliverables:** Test deliverables can be categorized based on the phase of the testing lifecycle:

**1. Before Testing (Planning Phase):** These deliverables are prepared before testing begins and define the scope, approach, and resources for testing.

Test Plan: Outlines the testing strategy, objectives, scope, schedule, resources, and risks.

Test Scenarios: High-level descriptions of what to test, derived from requirements and specifications.

Test Environment Setup Document: Details the configurations and requirements for the test environment.

Requirement Traceability Matrix (RTM): Maps Test Cases to requirements to ensure complete coverage.

**2. During Testing (Execution Phase):** These deliverables are generated while the testing process is ongoing and capture the execution results and progress.

Test Cases and Test Scripts: Step-by-step instructions for executing tests, including input data, er, and ar.

Test Data: Input data sets used during testing.

Test Execution Reports: Records of executed tests, including pass/fail status and any encountered issues.

Defect Reports: Logs of identified defects, including details like severity, steps to reproduce, and assigned developers.

**N:** Instead of arguing, I gathered evidence, including logs, screenshots, and a step-by-step reproduction of the issue.

**3. After Testing (Closure Phase):** These deliverables summarize the results, findings, and overall effectiveness of the testing process.

Test Summary Report: A comprehensive document summarizing testing activities, coverage, defect status, and overall results.

Defect Metrics and Analysis: Quantitative data about detected defects, their severity, and resolution.

Test Closure Report: Indicates that testing activities are complete and includes a summary of achieved goals and unresolved issues (if any).

User Acceptance Test (UAT) Results: Feedback and approval from end users validating the system for release.

Release Notes: describes about the build details like Build version number, build location, Modules changed, Installation steps, known issues and Bug fixes.

**@. Test Case and Test Script-----** stdd

|  |  |
| --- | --- |
| **Test Case** | **Test Script** |
| It is a step-by-step procedure to test an appln manually | It is a set of instructions or code used to automate TC execution |
| The term Test Case is used in the manual testing env. | The term Test Script is used in automation testing env. |
| It is done manually. | It is done by scripting format. |
| It is developed in the form of templates and it includes TC ID, TD, Test procedure, AR, ER, etc | It is developed in the form of scripting. Here will use different commands and logics to develop the Test Script |

**@. Y Performance Testing required ----** rsr

* It helps in checking the reliability of application (whether application is providing correct and consistent output)
* It helps in evaluating the scalability of application (conduct performance tests with different number of concurrent users, we can check if the application is capable of scaling up to a higher number of users)
* It helps in checking the robustness of application (with Stress Testing, we can check the robustness of application).

**@. Software Testing Principles -----** teedp ta

**Adv:** There are seven principles in software testing to maintain uniform and consistency in output those are:

1. **Testing shows presence of defects:** While testing only, we can identify that the application has any defects.
2. **Exhaustive testing is not possible:** Testing all feature and functionalities with all inputs and outputs is impossible.
3. **Early testing:** Early stage identified defects will take less time and less cost to fix compare to later stage id defects
4. **Defect clustering** is nothing but if small module might contain most of the bugs detected.
5. **Pesticide paradox:** If we try to test google form for that we written 10 TCs in the 1st we get some bugs and 2nd also we get some bugs… those repeated usage of the same TCs without modifying it till no longer yield the new bugs.
6. **Testing is context dependent:** To test different kind of applications, we will take the help of various kind of testing’s, different techniques, approaches, and multiple methods.
7. **Absence of errors fallacy:** If the software is unusable (or does not fulfil users' wishes) then it does not matter how many defects are found and fixed – because it is still unusable.

**@. Test Scenario and Test Condition**

**Test Scenario** is a process to test an application with all possible ways. These are the input for creation of Test Cases. A Scenario can contain single or group of Test Cases. By writing scenario, we can understand the functionality of application. It reduces complexity.

**Test Condition** should be followed static rules to test an application. It makes system bug free.

**@. Test Procedure and Test Suite ----** coc

|  |  |
| --- | --- |
| **Test Procedure** | **Test Suite** |
| It is a combination of Test Cases to test an application | It is a group of Test Cases to test an application. |
| Here order of execution is fixed. | Here order of execution may not important. |
| Test procedure contains End-to-End Test Cases. | Test Suite contains all new features and Regression Test Cases. |

**@. Wt can be delivered in the sprint and h the work gets done**

Product Owner discusses the goal of the sprint. Here will select the items from product backlog and include in sprint backlog. Based on sprint backlog, team needs to decide how it will work towards developing a usable increment.

**N:** Suppose if we take the user story for current sprint and developer and tester not able to complete that particular story that story points will go to backlog and these story points will add in next sprint.

**Random Testing** is one of the BBT techniques here the application is tested with different random data

**@. Exploratory Testing, Approach and H to handle exploratory testing in agile env’s with rapid iterations ----- explore**

**Exploratory Testing** is a type of testing in which testers explore the application without predefined TCs, depends on their experience, domain knowledge, and curiosity. Here testers check the system on fly to identify the bugs by this they may note down ideas about what to test before test execution. It is widely used in Agile models and it is all about discovery, investigation and learning. It is helpful when the project with complex business logic or incomplete / unclear / less documentation / requirements to understand the application functionality and behaviour.

**Approach:**

1. Define a charter or focus area (e.g., testing login functionality or user profile).

2. Start with basic functionality, then explore edge cases.

3. Take notes on any unusual or unexpected behavior.

4. Prioritize areas with high complexity or recent changes.

5. Use tools (e.g., session recorders) to document findings.

**Scenario:** New feature delivered in 2-day sprint: One-click checkout.

**Approach:**

- Use exploratory sessions with time-boxing (e.g., 1-hour session focused on checkout flow).

- Create a checklist based on user behavior (e.g., logged-in vs guest checkout).

- Note issues immediately, log high-priority ones quickly.

-Collaborate with devs on the spot for any critical defect.

- Share findings with the team during daily stand-up.

**@. Ad-hoc Testing**

Ad-hoc testing is an informal, unstructured software testing method where testers explore the application freely without a pre-defined test plan or detailed documentation and trying to find defects by using their understanding of the system and intuition. It can apply: When there is limited time and we need quick defect detection, after formal test execution is done and we want to double-check critical areas.

It will do testing randomly on any part of application. The main aim of this testing is to find defects by random checking. Ad-hoc testing can be achieved with the **Error Guessing Technique.**

**@. Ad-hoc and testing and Exploratory testing**

|  |  |
| --- | --- |
| **Ad-hoc testing** | **Exploratory testing** |
| Completely unstructured | Semi-structured (learn+test\_note) |
| Documentation not maintained | Often documented (notes or session reports) |
| Anyone (even-non-testers sometimes) can perform | Usually skilled testers can perform |
| I can do last-minute or during crunch time | It can do anytime, especially early in development |

**2. To perform Risk-based testing, and wt factors we would consider**

Risk-based Testing is a software testing approach. It prioritizes writing TCs for high-risk areas of the application.

**Scenario:** Ur working on an e-commerce app with modules like search, checkout, cart, and user reviews.

**Approach:**

**-** Risk Identification: Checkout has high business impact if it fails, revenue is lost.

**-** Factors Considered:

- Business Impact: Which features generate revenue?

- Complexity: Which modules are hard to test or prone to bugs?

- Past Defects: History of bugs in certain areas.

- Usage Frequency: Search and Checkout are used more than Reviews.

**-** Execution:

- Test checkout flow thoroughly (multiple cards, addresses).

- Run Edge cases on coupon application

- Less time on rarely used features like product reviews.

**@. Is testing depends on TCs or testing types**

If any Build came first will conduct sanity test to know whether the build is acceptable or not (Let us assume there are chances that in a day we can expect there are 5 to 10 builds from developer side each time we will not test manually all those sanity TCs. We will automate the TCs that make we to accept the build or not)

**N:** **Template** means process document to prepare TC’s or TS’s

**N:** In service-based companies’ client / client-side person will work as scrum master and in product-based companies we can expect all the employees from company itself.

**@. Types of software testing / Functional and Non-functional testing**

There are two main types of software testing: Functional and Non-functional testing

|  |  |
| --- | --- |
| **Functional Testing** | **Non-functional Testing** |
| It is a form of BBT as the name suggests it focus on testing the functional requirements in the application rather than internal implementation. | Testing the non-functional requirements of the application. These includes performance, security, scalability and usability, and wont crash under heavy load these are explicitly requested by client. |
| Under the fun testing will come Smoke, Sanity, Regression, E2E, Acceptance testing... | Under the NF testing will come Usability, Load, Stress, Performance…. |
| Functional test data will be as per the requirement | Non-functional test data like testing with 5, 10, 20 users…… |
| Here will check the customer requirements are meet or not | Here will check the stability and response time of the appln |

**Ex’s of Non-Functional Testing:** Performance testing, Load testing, Security testing, Usability testing, Compatibility testing, installation testing, reliability testing. Non-functional testing comes after Functional testing.

**@. Y any company has to do bug release --** If a bug is high priority bug the operation will get stop in production time.

**@. Security testing**

Security testing verifies how secure the software is from threads and risks. For this tries to break a software’s security checks, to gain access to confidential data. It is crucial for web-based applications or any applications that involve money and goal is to ensure that the sensitive data remains secure.

**N: Reliability testing** is carried out in order to confirm does the software displays reliable output for longer period of time irrespective of the operating system and browser.

**@. Complains testing**

It is a non-functional test that’s actually validates does the designed software meets all the specified standards.

**@. Migration testing**

To make sure that the migrating data (nothing but data loss) from one version to other version should be minimal so that the production will not getting effected.

**N:** Wheneverinvalid or random data was given to the application and check what kind of exceptions are coming like software getting crashed or memory leakage happening is called **Fuzz testing**.

**@. H to perform Debugging in manual testing**

For example, a company having two kind of servers one is production server (production people will use) and other is QA server. Testers using QA server and all the requirements will go to QA server. What we need to do login to application, go to directory where the logs are present (logs are in human understanding readable language) and see the logs what API is called, what API is breaking with what error. Share those logs to developer is called debugging.

**N: Error Seeding** is a process of adding known bugs in a software to check the quality of the tester how can he manage

**N: To create sprint:** Scrum master will take task from product backlog. If product backlog is over, he will take task from bug bucket of p0 priority bugs and assign to developer we re-test again......

**N:** Peer Review is also called as Technical Review. It will conduct for TCs review and approval process.

**@. If lot of changes made in a build by developer and less time to finish wt u do / Wt do in impact analysis meeting**

I will conduct the **impact analysis meeting** with the developer and the product manager to ask what changes u done and which module going to impact due to these changes. And I will test that particular thing only.

**N:** Impact analysis meeting helps to decide ‘How much regression testing should be done’.

**@. If click on end button in zoom, it's not working wt is your approach**

Check in network calls whether API is called or not to find its UI issue or not. If API is working check the application logs.

**@. Wn should stop testing**

* All the bug fixes have done
* One complete cycle of all the Test Cases have been executed (i.e. certain percentage of TCs have passed)
* All the critical, major, moderate defects are resolved.
* Based on automation code coverage value.
* Testing budget is out of funds.
* Based on Mean Time Between Failure (MTBF) (MTBF is a time interval between two inherent failures) is quite large then we can stop the testing phase. It is decided by Stakeholder’s.
* Once the testing deadline is met with no high priority issues left in the system.

**@. Test Bed and Test Harness**

**Test Bed** is a test environment used for testing an application. A Test Bed configuration can consist of hardware, software and network configuration requirement of the application under testing process.

**Test Harness** is a collection of software, test scripts and test data that is often used in unit testing, integration testing, and system testing to ensure software functionality and reliability.

**N: Defect cascading** is happening when a defect is not caught by the testing team and it leads the discovery of other defects. To prevent Defect Cascading: Fix the original defect properly.

**N: Recovery testing** is to test how Quickly the application recover from the failure or from system crash or from the hardware failure to original state.

**@. Scrum and Agile ---** Scrum is a framework through which we build the software product by following agile principles

**N: Story points will calculate** based on complexity of Test cases it means how much time it will take for execution.

**N:** Product owner is responsible for preparing User story. He will define the features of the software in the form of user story.

**N:** Product manager is responsible for writing the epics and stories these will get from the customer

**Continuous Integration** is a software development process in which the changes are made in software that are integrated into the main code. Here will be having one main branch in which all the code will be checked it and simultaneously other developers who are regularly creating new code or working on new features this code will put and integrate into the main code whenever this patch will be ready it will build, test, deploy and monitor.

**Continuous Delivery** is nothing but we are delivering the changes which are happening in the CI.

**Continuous Deployment:** Whenever the CI changes are there, those are deployed into the target environment

**@. Epic, Task and User story**

An Epic, Task, and User Story are common terms used in Agile and project management frameworks, tools like Jira.

**Epic:** Epics are used to organize and track high-level objectives or initiatives. Here a large body of work that can be broken down into smaller tasks or user stories. It often represents a big goal or significant feature in a project.

**User Story:** A specific requirement (feature or functionality) from the end user's perspective. It captures all the user’s needs.

Typically, it follows the structure: "As a [**user role**], I want [**what they want**] so that [**why they want it**]."

**Task:** Actionable work or activity required to complete a user story, bug fix, or any other work item. It Tracks granular pieces of work that contribute to achieving a specific goal (Ex: within a story or directly that related to an epic).

Scope: Usually smaller and completed within a single sprint.

**Simply,** Epics are breakdown into multiple user stories and user stories further breakdown into multiple tasks then the team can work on to implement the story.

**@. Scrum and sprint**

Scrum is an Agile framework used to manage and develop complex projects in software development. It focuses on collaboration, iterative progress, and adaptability. Scrum teams work in short, time-boxed cycles called Sprints to deliver incremental improvements to a product. Simply, scrum is the overall framework that guides the Agile process.

A Sprint is a fixed time period (typically 1 to 4 weeks) within the Scrum framework here the team works on a defined set of tasks to achieve a specific goal. Each **sprint** begins with a planning and meeting and it results in a potentially shippable product increment. Simply, Sprints are the building blocks of Scrum, where actual work happens in iterative cycles.

**@. Scrum Events / Scrum process**

**Product Backlog** is a prioritized list of features or tasks maintained by the Product Owner.

**Sprint planning** is the detailed discussion of what work to be performed in the sprint. Here **Scrum** **Master** is responsible for making this and participants can understand its purpose. And also, the team selects items from the Product Backlog to work on during the Sprint.

**Sprint Execution** – The development team works on the tasks during the Sprint (1-4 weeks).

**Daily Scrum –** A short daily meeting to track progress and discuss blockers.

**Sprint review** is conducted at the end of the sprint. Here team will demonstrate the completed work to stakeholders. And discuss about implementation of month-long sprint.

**Daily scrum** is a time-box to 15 min to discuss development team’s activities for the next 24hrs and it also includes discussions regarding the what work performed in last 24hrs.

**Sprint Retrospective** is nothing but a meeting and that can be facilitated by the **Scrum** **Master**. Here team will discussion on just-concluded sprint and also determines what could be change in the next sprint to make sprint is more productive. It often helpful to management to get feedback from the team about the progress work.

**Sprint estimation** is the process of estimating how much effort, time, and budget is required to complete a set of tasks within a sprint. In Scrum, the entire team estimates during the Sprint Planning Meeting.

**Common estimation techniques:**

**Story points:** Teams use story points to estimate the size of a user story. For example, a task with a lower story point might be less complex and require less effort than a task with a higher story point.

**Sprint Backlog** is a subset of the Product Backlog, it containing tasks and user stories selected for a single Sprint (typically 1–4 weeks). Those are not done at the time of sprint execution.

**@. Common mistakes that lead to major issues**

Not following the exact process, ignoring small issues, underestimating, and improper resource allocation.

**@. Software Testing Lifecycle (STLC)**

Similar to software development, software testing also has its lifecycle. During the testing tester goes through the following activities.

Understanding the requirements: Before testing software or a feature, tester must first understand the what is supposed to do. If they don’t know how the software is supposed to work, they can’t test it effectively.

Test planning:

Test design (Test Case development):

Test environment setup:

Generate the Test data: Thought it is impossible to do exhaustive testing of the software, the tester tries to use realistic test data to give them the confidence that the software will survive the real world if it passes the tests.

Test Execution:

Test Closure: At the end of the test execution there can be two possible outcomes. First, tester finds a bug in the software under test. In this case they create a test record/bug report. Second, software works as expected. Both these events indicate the end of the test cycle.

**@. Types of testing**



**@. SDLC models / methodologies**

**1. Waterfall Model** is a sequential process where each phase (Requirements capturing, Design, development, Testing, Deployment, Maintenance) must be completed before moving to the next.

**Key Features:**

* Testing begins only after the development is complete.
* Each phase (Requirement, Design, Development, Testing) must be completed before moving to the next.

**Adv:**

* Simple and easy to manage.
* Works well for small projects with clear requirements.

**Dis-adv:**

* Bugs found late in the lifecycle can be costly.
* Not suitable for dynamic requirements.

**2. V-Model (Validation and Verification Model)** is a structured SDLC methodology where development and testing processes run in parallel. The "V" shape represents the association between development phases and their corresponding testing phases, ensuring validation and verification at every step.

**Key Features:**

* Correspondence between development stages and testing phases.
* Emphasizes early testing.

**Adv:**

* Reduces defect leakage.
* Suitable for projects with well-defined requirements.

**Dis-adv:**

* Less flexible for changing requirements.
* High initial planning effort required.

**3. Agile Testing** is a popular Software Development Life Cycle (SDLC) methodology that emphasizes iterative, incremental development, collaboration, and flexibility in responding to changes.

**Key Features:**

* Continuous testing and feedback.
* Collaboration between developers and testers.

**Adv:**

* Flexible for changing requirements.
* Early detection of defects.

**Dis-adv:**

* Requires skilled testers and strong communication.
* Testing in sprints may become rushed.

**4. Spiral Model** is a software development life cycle (SDLC) model designed to manage risk in large and complex projects. It integrates iterative development with systematic risk management and suitable for projects where requirements are not well defined at the start or are expected to evolve.

**Key Features:**

* Risk-driven approach.
* Repeated testing in every spiral phase.

**Adv:**

* Effective for large, complex, and high-risk projects.
* Flexible for requirement changes.

**Dis-adv:**

* Requires significant expertise in risk analysis.
* Expensive for smaller projects.

**@. Segregate the Manual and Automation Tests Cases**

|  |  |
| --- | --- |
| **Criterial for Manual testing** | **Criterial for Automation testing** |
| Use manual testing for Ad-hoc and Exploratory scenarios, Usability Testing, Tests with frequent changes | Use automation testing for repetitive and time-consuming Test Cases like: Regression, Data-driven tests, Performance testing |

**@. Test Data management**

1. Create reusable, well defined test data sets

2. Ensure data covers positive, negative and edge cases

**@. Dependency handling in Test Cases**

Identify dependencies between Test Cases (User creation is required before testing login).

Use pre-conditions and link Test Cases accordingly to ensure proper execution order.

**@. Test Case versioning**

**1.** Maintain versions of the Test Cases to track changes over time as the application evolves. Benefits:

* Allow rollback to previous versions if required
* Ensure traceability when features are updated or deprecated

**2.** Version control tools GitHub, Test Rail… for managing Test Case updates.

**N:** **Cross-Environment Testing** is nothing but create Test Cases to validate functionality across different environments like (Dev, QA, Production)

**@. Debug process while automation script fails**

When automation script fails debugging process involves identifying the root cause and fixing it

1. Check the error logs and identify the line of code where the script failed.

2. Verify the test data end environment setup

3. Re-run the script with debug mode enabled to identify the issue

4. Fix the issue and rerun the script to ensure it passes

5. Update the Test case and report the issue to the development team if necessary

**Reports in testing**

**Test Summary Report (TSR):** It summarizes the overall testing activities at the end of a test cycle. (Test objectives, Test scope, Test execution summary, Defects summary, Test environment details, Recommendations)

Ex: "Out of 500 test cases, 450 passed, 30 failed, and 20 were blocked."

**Test Execution Report:** It provide details on test case execution during a specific period. (Number of test cases executed, Pass/fail status, Pending test cases, Execution trends)

Ex: "90% of test cases executed, with a 95% pass rate."

**@. Defect Report (Bug Report):** It documents detected defects and tracks their resolution status.

(Defect ID, Summary, Steps to reproduce, Severity & priority, Status (Open, In Progress, Fixed, Closed), Assigned to)

**Ex:** "Bug #101: Login button not working in Chrome, assigned to Developer X."

**Test Case Report:** It tracks the status of individual test cases. (Test case ID, Description, Execution status (Pass/Fail/Blocked), Defect mapping (if failed))

**Ex:** "Test Case TC\_001: Passed in all browsers except Safari."

**RTM:** It ensures that all requirements have been covered by test cases.

(Requirement ID, Test case ID(s), Execution status)

**Ex:** "Requirement RQ\_05 has been verified by test cases TC\_101 and TC\_102."

**Test Coverage Report:** It measures how much of the application has been tested.

(Code coverage percentage, Functional coverage, Risk-based coverage)

**Ex:** "80% of critical functionalities tested, 70% code coverage achieved."

**UAT Report:** It summarizes testing conducted by end-users before final release. (Features tested, Issues reported, User feedback, Final approval status)

**Ex:** "Users reported 3 UI issues but approved the release."

**Automation Test Report:** It shows results of automated test scripts. (Number of scripts executed, Pass/fail ratio, Execution logs, Error screenshots)

**Ex:** "Out of 200 automated test cases, 180 passed, 15 failed, 5 skipped."

**@. To log defect in Jira**

**Step 1:** Log in to JIRA

Open JIRA in your browser.

Enter your username and password.

**Step 2:** Navigate to the Project

Select the Project where we want to log the defect.

Click on the Create button in the top navigation bar.

**Step 3:** Create a New Issue: Click on the “Create” button on the top navigation bar.

**Step 4:** Select Issue Type

In the Create Issue window:

Select the appropriate Project.

In the “Issue Type” dropdown, select "Bug" as the issue type.

**Step 5:** Fill in the Bug Details

Summary: A short, descriptive title of the defect.

Description: A detailed explanation of the defect, including:

Steps to reproduce:

Expected vs. actual results

Screenshots, logs, or error messages (if applicable)

Priority: Set the urgency level (e.g., Critical, High, Medium, Low).

Environment: Specify the affected platform, browser, OS, or application version.

Assignee: Assign the bug to the appropriate developer (or leave unassigned).

Attachments: Upload screenshots, log files, or any other supporting documents.

Labels (Optional): Add labels for easy tracking, Specify the affected component (e.g., UI, Backend, API).

**Step 6:** Submit the Defect

Click Create to log the defect in JIRA.

**Step 7:** Track and Update the Bug

Monitor progress under the "Issues" section.

Add comments or updates as needed.

Change the status (e.g., "In Progress," "Fixed," "Closed") when appropriate.

**@. Link bugs with user story in Jira**

Linking bugs with user stories is essential for maintaining traceability and ensuring that all issues are properly addressed within the development cycle. Most project management and issue tracking tools (e.g., Jira, Azure DevOps, Trello, Rally, ClickUp) allow to link bugs directly to user stories. In Jira we can use the "**Link Issue**" feature to associate a bug with a user story. When logging a bug, Include the User Story ID in the Bug Report

**@. Types of Closure Reports in Software Testing**

A closure report is a formal document that summarize the completion of testing activities, including defect resolution and test execution status.

**1. Test Closure Report:** Summarizes overall test activities and results at the end of the testing phase.

Contents:

Summary of test execution (test cases passed/failed)

Defect summary and resolution status

Outstanding defects (if any)

Test coverage report

Lessons learned and recommendations

Approval and sign-off

**2. Defect Closure Report:** Documents all reported defects and their final status.

Contents:

Total defects reported

Defects fixed, deferred, or rejected

Severity and priority distribution

Root cause analysis

**@. To ensure quality of software applications**

Software quality can be ensured by following best practices like testing, quality management, and risk management.

**Testing:** Usability testing, Compatibility testing, Performance testing, Regression testing, Security testing, Unit testing.

**Quality management:** Test plan and Quality management plan (Defines standards and sets roles and responsibilities).

**Risk management:** Identify the risk areas and find solutions to clear them.

**N:** Gap Analysis testing technique is used to identify the missing requirement.

@. Testing pyramid and its significance in software testing

**1. Unit Tests (Base Layer):** Tests individual components or functions in isolation.

**Tools:** JUnit, NUnit, pytest, Jasmine, etc.

**2. Integration/Service Tests (Middle Layer):** Tests interactions between components or services, such as APIs, DBs, etc.

**Tools:** Postman, RestAssured, TestNG, Spring Test, etc.

**3. UI Tests (Top Layer):** Tests the application through its user interface (end-to-end testing).

**Tools:** Selenium, Cypress, Playwright, etc.

**@. Y software testing is necessary**

**Detecting Bugs Early:** Testing helps to identify bugs and issues early in the development cycle, reducing the cost and time of fixing them later.

**Ensures Quality:** Testing verifies that the software works correctly, providing confidence in the product's stability and performance.

**Security Assurance:** Security testing ensures the software is free from vulnerabilities, especially critical in banking, healthcare, and e-commerce apps.

**Customer Satisfaction:** Delivering a bug-free, smooth user experience increases customer trust and satisfaction.

**Validates Requirements:** Testing confirms that the application meets both functional (what it should do) and non- functional (how it should do it) requirements.

**Reduces Maintenance Costs:** Well-tested software is less likely to fail, reducing future maintenance and support costs. **Improves Performance:** Testing can identify performance issues (like speed, responsiveness) that can be optimized before release.

@. H to handle unclear requirements as a QA tester

* Reach out to the Business Analyst (BA), product owner, or client ask for missing user stories or acceptance criteria, clarification on flows or edge cases. **Ex:** Can u confirm what should happen when a user enters a wrong OTP 3 times.
* Refer available documents clearly.
* Collaborate with developers or product team.
* Use Exploratory Testing.
* Log assumptions in TCs.
* Push for requirement reviews (Raise it during daily stand-ups or sprint planning)

Defect detection ratio (DDR) = (no. of defects rejected / total no. of defect raise)\*100

Defect leakage ratio (DLR) = (no. of defects missed / total defect of software)\*100

@. To choose which TC to automate

Stable - The functionality is not changing frequently

High-priority - Core business flows like Login, checkout, and search

Repetitive - Tests that are run in every sprint or Regression

Frequently executed - Smoke and Regression TCs

**N:** Avoid automating dynamic UI elements or one-time TCs

@. W if a TC fail during execution

Analyse the failure, check for environment or data issues, log a bug if it’s a genuine defect, Retest once fixed.

@. To ensure complete test coverage

Use RTM, review requirements and TCs, perform peer reviews, validate edge cases and alternate flows.

@. To ensure a defect report is effective

1. Clear Title: Use a descriptive title summarizing the defect.

2. Detailed Steps to Reproduce: Provide precise steps, test data, and environment details.

3. Expected vs. Actual Result: Clearly state what was expected and what actually happened.

4. Attachments: Include screenshots, logs, or videos to provide evidence.

5. Severity and Priority: Assign appropriate levels based on impact and urgency.

6. Environment Information: Specify browser, device, or OS details where the defect was observed.

@. H would you test a login page

Testing a login page involves a combination of functional, security, and UI testing. Key test cases include:

1. Functional Testing:

Valid credentials: Verify login with correct username and password.

Invalid credentials: Check error messages for wrong username/password.

Blank fields: Ensure validation messages appear when fields are left empty.

2. BVA: Test username and password with minimum and maximum character limits.

3. Negative Testing:

Test with SQL injections, special characters, or script tags.

Check for error messages when submitting without input.

4. Security Testing:

Verify password encryption and secure data transmission (HTTPS).

Ensure the application prevents brute-force attacks by locking accounts after multiple failed attempts.

5. Usability Testing: Validate field alignment, placeholder text, and ease of navigation.

6. Cross-Browser Testing: Verify the login functionality on different browsers and devices.

@. To handle incomplete or ambiguous requirements?

1. Clarify with Stakeholders: Collaborate with business analysts, product owners, or clients to get clarity.

2. Refer to Similar Features: Use knowledge from similar modules or past projects as a reference.

3. Document Assumptions: Clearly outline assumptions about expected behaviour and share them for approval.

4. Create a Risk-Based Plan: Focus on critical functionalities and perform exploratory testing for the unclear areas.

5. Communicate Effectively: Keep all stakeholders informed about the challenges and testing strategy.

@. H to test a payment gateway

1. Functional Testing:

Verify successful transactions using valid card details.

Test failed transactions with invalid, expired, or insufficient funds cards.

2. Security Testing:

Ensure sensitive data (card number, CVV) is encrypted.

Validate secure communication via HTTPS.

3. Boundary Value Testing: Test amount limits (minimum and maximum transaction values).

4. Integration Testing: Verify integration with third-party payment processors like PayPal or Stripe.

5. Negative Testing: Test transaction interruptions (e.g., network failure, session timeout).

6. Performance Testing: Simulate high transaction loads to ensure stability under peak usage.

@. H u define a QA strategy for a large or complex application

**Scenario:** Suppose ur testing a banking appln that includes user login, fund transfer, loan processing, and reporting modules.

**Approach:**

I define a QA strategy by breaking it into the following components:

- Understand Requirements: Collaborate with stakeholders to understand business needs.

- Test Scope & Objectives: Identify what needs to be testedfunctional areas, integrations, performance, security, etc.

- Risk Assessment: Identify modules that are complex or business-critical (e.g., fund transfer).

- Resource Planning: Assign experienced testers to critical modules; junior testers to low-risk areas.

- Test Types: Include functional, regression, integration, UI/UX, API, and UAT.

- Test Environment: Create multiple environments for testing Dev, QA, UAT.

- Automation Strategy: Automate stable regression cases, especially for login, account summary, etc.

- Entry & Exit Criteria: Define when testing starts (Ex: test data ready, build deployed) and when it ends (Ex: 95% pass rate, critical bugs closed).

@. Biggest challenges in manual testing today, and H do you tackle them

**Challenges:** Time-Consuming Regression Testing, Test Coverage Gaps, and Flaky Requirements.

**To tackle them:**

- Prioritize test cases using risk-based approach.

- Collaborate closely with BAs and Devs to clarify requirements early.

- Use mind maps and checklists for exploratory testing.

- Begin test planning while development is ongoing ("Shift Left").

- Advocate automation for repetitive flows like login or search.

@. To mentor junior testers in test planning and execution

Scenario: On a new project, 3 freshers joined the QA team.

Approach:

- Walk them through the application end-to-end.

- Explain test design techniques using real examples (e.g., boundary testing on date of birth).

- Help them write test cases and review them together.

- Pair test them for first few test cycles.

- Encourage them to ask questions and share their learnings in team meetings.

- Assign ownership of small modules to build their confidence.

@. To measure and improve the effectiveness of ur Test Cases?

Metrics Used:

- % of test cases that catch bugs (defect yield)

- % of test cases reused across cycles

- Mapping test cases to requirements (traceability matrix)

Improvements:

- Regular reviews of test cases after UAT.

- Update test cases post-production issues.

- Remove redundant or outdated test cases.

- Add edge case scenarios based on bug history.

@. H do u ensure manual testing adds value in a team moving toward automation?

Scenario: Team is adopting Selenium automation but release is near.

Approach:

- Focus on exploratory testing to catch UI/UX issues automation might miss.

- Cover new features that are not automated yet.

- Provide detailed feedback and edge case coverage to developers.

- Use manual testing for areas where automation is flaky or unstable (e.g., Captcha handling).

- Be a bridge between business and automation team ensure test coverage matches business expectations.

@. Approach to testing legacy systems with poor documentation?

Scenario: Testing an old HR system with no test cases or documentation.

Approach:

- Explore the application module by module.

- Interview developers or senior users to understand functionality.

- Perform reverse engineering observe existing functionality to derive expected behaviour.

- Document every observation as informal TCs.

- Start building a regression suite gradually.

- Track issues and link them back to discovered requirements.

@. To ensure test coverage for cross-platform or cross-browser scenarios

**Scenario:** Application used in Chrome, Firefox, Safari; also on Android and iOS.

**Approach:**

- Identify top-used platforms using analytics (e.g., 80% on Chrome, 10% on Safari).

- Prioritize functional and UI tests on those platforms.

- Create a test matrix (Browsers vs Features).

- Use tools like BrowserStack for parallel testing.

- Document layout, font, and functionality mismatches.

- Run regression on different screen sizes (mobile, tablet, desktop).

@. Describe a time when ur manual testing caught a high-impact defect missed by others.

**Scenario:** On a banking portal, I tested fund transfer during a planned UAT.

**Defect:** The amount transferred was showing incorrectly in the UI after rounding off (99.99 was showing as 100.00).

**Impact:** Could lead to compliance issues.

**Action Taken:** I reported it with screenshots and steps. Dev’s fixed it with high priority before release. Stakeholders appreciated the attention to detail.

@. To participate in sprint planning and backlog grooming as a QA lead?

Scenario: During sprint planning for a new feature Loyalty points.

Lead Role:

- Review user stories for testability.

- Ask questions: What happens if a user returns the product?

- Estimate testing effort and capacity.

- Identify test dependencies (API, data).

- Split stories if too large.

-During grooming, I help clarify acceptance criteria and prepare test scenarios in advance.

@. Wt kind of metrics we track and report to stakeholders?

**Metrics Shared:**

- Test Coverage (% of requirements tested)

- Defect Density (defects per module)

- Severity Distribution (High/Medium/Low)

- Test Execution Status (Passed/Failed/Blocked)

- Bug Turnaround Time

- Automation Progress (% automated vs manual)

**How Shared:** Weekly Test Report with charts and Daily updates in stand-up.

End#3

**Appium**

Pre-req

NodeJS (inside npm will come)

Android Studio -(acts as one ide, having sdk packgaes, and to generate vertual devices)

Install Appium Server (through cmd: npm install -g appium) (to start 'appium server' (and know version also) - appium, stop cmd - Ctrl+Shift+C) (cmds: appium driver list, appium driver install uiautomator2, appium driver update uiautomator2

Appium server GUI (insted of using cmd prompt to start 'Appium Server' by using 'appium' cmd)

Appium inspector (Or APK inspector - to inspect elements)

for android we are using 'uiautomator2' driver

to check appium driver list: appium driver list --installed

install appium-doctor: npm install appium-doctor -g

Real device: as per customer req will do some testing on real devices

Vertual Devices (not real devices):

Emulator (Android vertual device)

Simulator (IOS vertual device)

AVD name: AndroidEmulator

install app through cmd: adb.exe install appiumTutorial.apk

To know the connected devices in cmd: adb devices

for capabilities refer 'capabilities documentation' below link

Dependencys

java-client (io.appium....), selenium-java, TestNG

default Appium taking port: 4723

to know qndroid version - adb shell getprop ro.build.version.release

know appPackage and appActivity

C:\Users\Admin>adb shell

ice:/ $ dumpsys window displays | grep -e 'mCurrentFocus'

mCurrentFocus=Window{a38a8a7 u0 cz.hipercalc/app.hipercalc.CalculatorActivity}

ice:/ $

ChromeDrive download : https://googlechromelabs.github.io/chrome-for-testing/

tgo inspect mobile chrome browser:

opnen desktop chrome browser -> chrome://inspect/#devices -> click 'inspect'

**@. To write Test cases for Mobile Testing**

1. Functionality: Validate app features (Ex: Login, navigation)

2. UI/UX: check layout, fronts, colours and responsiveness

3. Performance: Access app load times and resource usage

4. Interruptions: Test behaviour on calls, notifications

5. Device compatibility: Test across different OS versions and devices

.

**Docker**

Docker provides containerized test environments for consistency and scalability.

docker version

docker run ubuntu - cmd tomcreate container for downloaded image - if the image is not downloaded but we are trying to execute this command. it will download image and create container in single step

docker ps - to show running containers

docker ps -a - to show all running and not running containers

docker logs continer\_id - to generate logs

docker run -it ubuntu - to create container and interact with continer before we start container and to start container

for 1 image we can create multiple containers

docker stop cont\_id - to stop container

docker start cont\_id -again to start container

docker rm cont\_id/name - remove container

docker system prune -f - to delete networks / remove all container and evrything in docker in sigle shot

3 images used for our automation

selenium/hub ----- acts as hub

selenium/node-firefox - (belongs to linux-firefox) ---- acts as node

selenium/node-chrome - (belongs to linix-chrome) ---- acts as node

create containers for above 3 images and these containers should be part of one single network (same network) (i.e. nodes and hub connecting eachother)

@@@@@@@@@@ 2nd tried (at the time of 1st trying this approach its get failed so tried below one as 1st and then tried this one as 2nd)

@@@@@@@@@@its working@@@@@@@@@@FInally 2nd tried this container for exceution. Its working fine but slow execution. ---- (for this execution changed all WebDriver's into RemoteWebDriver's)

Running Docker Containers by using below 4 commands.

docker network create grid ----('grid' is just name) ---command to create network

docker run -d -p 4442-4444:4442-4444 --net grid --name selenium-hub selenium/hub ----command to create hub

docker run -d --net grid -e SE\_EVENT\_BUS\_HOST=selenium-hub -e SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 -e SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443 selenium/node-chrome ----command to create node1

docker run -d --net grid -e SE\_EVENT\_BUS\_HOST=selenium-hub -e SE\_EVENT\_BUS\_PUBLISH\_PORT=4442 -e SE\_EVENT\_BUS\_SUBSCRIBE\_PORT=4443 selenium/node-firefox ----command to create node2

docker run -d -p 4444:4444 -v /dev/shm:/dev/shm selenium/standalone-chrome:latest

net grid - network name

name selenium-hub - node name

selenium/hub, selenium/node-chrome, selenium/node-firefox - image names

4442-4444 - in between these 3 range any one of the port will allocate (4442, 4443, 4444) dynamically

-d - represents this process will happen in the backend system

grid console path: http://localhost:4444/ui/ --paste it in browser then we can see the grid console

@@@@@@@@@@@ 1st tried

@@@@@@@@@@@its working@@@@@@@@@FInally 1st tried this container for exceution. Its working fine but slow execution. ---- (for this execution changed all WebDriver's into RemoteWebDriver's)

docker pull selenium/standalone-chrome:latest

docker run -d -p 4444:4444 -p 7900:7900 --shm-size="2g" selenium/standalone-chrome:latest

Error Hints:

1) go to Services --> Restart/strat 'Docker Desktop Sevice'

2) Execute only through dockertextng.xml file (RC on dockertextng.xml file --> run as TextNG text --> E)

3) first execute individual TC classes throug .xml file (not all at a time) for stability of individual classes. if fail execution try 3-4 times then it will pass in more cases

3b) then execute group of TC classes at a time. if fail execution try 3-4 times then it will pass in more cases

3c) in above case mostly test fail due to server issue so keep on try 3-4 times the test may pass

docker run -d -p 4444:4444 --name selenium-hub selenium/hub

docker run -d --link selenium-hub:hub selenium/node-chrome

**Jenkins**

BalakrishnaAratipamula - Sr786bhanu786#1

1) if u want to see execution on real browser download below war file (for practicing purpose)

Download Jenkins 2.479.2 LTS for:

Generic Java package (.war)

\*\*\*above approach is require to start jenkins through cmd manually

2) or u want headless approach go with jenkins.exe installer (for real time use)

for .exe link: see below above war file download -> click on windows -> jenkins.msi file will download -> install that file

\*\*\*above approach is not require to start jenkins through cmd manually

\*\*\*jenkins - GitHub integration

1)

start Jenkins throughn cmd go to jenkins.war location -> opem cmd prompt -> java -jar jenkins.war -> E -> go to browser -> http://localhost:8080 -> E -> we can see jenkins dashboard

2)

install maven plugin in jenkins

Manage jenkins -> plugins -> available plugins -> search for 'maven integration plugin' -> install -> check 'installed plugins' for installation done or not

specify java, git and maven locations: Manage Jenkins -> tools -> give below paths

give any userdefined name ------ C:\Program Files\Java\jdk-21

give any userdefined name ------ C:\Program Files\Git\bin\git.exe

give any userdefined name ------ D:\apache-maven-3.9.6

apply and save

3)

to start job: New Item -> Enter an Item name (give any name) -> click on 'maven project' -> ok

Parameters:

Description --> give some meaningful description --> click on 'Git' radio btn -->

Repository URL : https://github.com/BalakrishnaAratipamula/SIMPLII\_AUTOMATION.git

--> credentials (if alredy created give those by clicking dropdown otherwise generate by click on add -> jenkins -> usename (github usename), password (github generated token), id (give any meaning full ID for ex PAT\_Bala)), Desription is same as ID -> Add -->

Branch : as per git (\*/maser or \*/main) --> under 'Build'

Root POM : AUTOMATION/OpencartV121/pom.xml (solution for how to set project path in jenkins in case of group of projects in github repository) -->

Goals and Options : test --> Apply --> Save

4)

to execute build

Dashboard --> click on Item\_name --> click on 'Build Now'

https://github.com/BalakrishnaAratipamula/SIMPLII\_AUTOMATION.git

GitHub generated Token

ghp\_00qAmYycVLJSUQmQykVeqYs5FesL7r1mO9vK

**SQL**

**1. What is SQL and when SQL was appeared**

SQL stands for Structured Query Language it is a declarative language used to communicate the DBMS. SQL was developed by IBM in 1974 later acquired by oracle. SQL is ANSI (American National Standards Institute) language for accessing Databases.SQL syntax is not care sensitive.

**2. What is SQL Process**

When we are executing SQL commands for any DBMs, the system determines the best way to carry out our request and SQL engine identifies how to interpret the task. There are various components included in this process. These components are Query Dispatcher, Optimization engines, Classic Query Engine and SQL query engine etc. Classic query engine handles all non-SQL queries and SQL query engine handle only SQL queries won't handle logical files.

**3. Is SQL supports Programming**

No, SQL doesn’t have any Conditional and Loop statements. Using SQL Commands, we can access the Databases only.

**4. What is MS Access**

MS Access is entry-level Database management software it was launched in 1992 by Microsoft Corporation as part of MS Office. It is not only an inexpensive and also powerful database for small-scale projects. MS Access comes with the professional edition of MS Office package and it is a user-friendly Data Base Management System.

**5. What is Oracle**

Oracle is a RDBMS developed by Oracle Corporation and launched in 1977. Oracle supports all major Operating Systems includes MS Windows, NetWare and most UNIX flavours.

**6. What is MS SQL Server**

MS SQL Server is a RDBMS, developed by Microsoft. Its primary query languages are T-SQL and ANSI SQL.

**7. What is Sybase**

Sybase is a name of computer software company and their primary product is Sybase RDBMS, it is a RDBMS based upon structured query language.

**8. What is MySQL Q: Teradata**  
MySQL is open-source RDBMS, developed by Swedish company. It is used for developing web applications

* MySQL Supports many different platforms including MS Windows, Linux distributions and Mac OS.
* MySQL has free and paid versions depending on its usage and features.
* MySQL comes with a very fast, multi-threaded and multi-user SQL database server.

**9. What is DB2** --------- DATABASE2 is a relational database product developed by IBM. in 1983  
**10. What is DB/400** -- DB/400 is a relational database product developed by IBM. It is one of the flavours of IBM DB2  
  
**11. What are Arithmetic operators in SQL**

|  |  |
| --- | --- |
| **Operator** | **Description** |
| + (Addition) | Add values |
| - (Subtraction) | Subtracts the Right-side value from Left side value |
| \* (Multiplication) | Multiplies values on either side of the operator |
| / (Division) | Divides left hand operand by right hand operand and returns co-efficient |
| % (Modulus) | Divides left hand operand by right hand operand and returns remainder |

**12. What are Comparison operators in SQL ----- or ----- Which Operators used in SELECT statements**  
 Let us assume x = 1 and y= 2  
**Operator    Example**  
=              (x = y) is False  
!=            (x!=y) is True.   
<>           (x <> y) is true.   
>             (x > y) is False  
<             (x < y) is True          
>=           (x >= y) is False  
<=           (x <= y) is True  
!<            (x! < y) is False  
!>            (x! > y) is True.

BETWEEN: Between an inclusive range LIKE Search for a pattern  
**N:** Comparison Operators are always return Boolean Results  
 **13. What are Logical operators in SQL**    
**Operator    Description   
AND**          Returns TRUE if both component conditions are TRUE **&&** Returns FALSE if either is FALSE.  
**OR**          Returns TRUE if either component condition is TRUE **&&** Returns FALSE if both are FALSE.

**NOT**           Returns TRUE if the condition is FALSE **&&** Returns FALSE if it is TRUE

**14. What is Data Relationship and What are they**

Database Relationship is nothing but a connection between the tables in a DB. There are 4 types of relationships:

* One to One Relationship, One to Many Relationship, Many to One Relationship and Many to Many Relationship

**15. What are the Important Data Types in SQL**

Char(x), varchar(x), int, smallInt, float(p), numeric(p,s), decimal(p,s), Date, Time, bit(x) and Real

**16. How to Select / Activate / focus particular existing Database in SQL** ------ **SQL>** USE db\_name;  
**17. How to view all existing Databases list** ------ **SQL>** SHOW DATABASES;

**18. How to fetch data from a Database Table**  
Using ‘SELECT’ Statement we can fetch (view table info) data from a Database Table  
**SQL>** SELECT column1, column2…..columnN FROM table\_name; **---OR---** SELECT \* FROM table\_name;

**19. Give a usage for BETWEEN ... AND**

SELECT emp\_name FROM table\_name WHERE age BETWEEN value1 AND value2 -- (the values can be numbers / dates)

**20. What is the use of CASCADE CONSTRAINTS** --- <https://www.techonthenet.com/sql_server/foreign_keys/foreign_delete.php>

When this **clause** is used with the **DROP** command i.e., a parent table can be dropped even when a child table exists.

**21. Why does the following command give a compilation error** --- **DROP TABLE &table\_name;**  
Variable names should start with an alphabet but here the table name starts with an '&' symbol.

**22. Which system tables contain information on privileges granted and privileges obtained**  
USER\_TAB\_PRIVS\_MADE **&&&&** USER\_TAB\_PRIVS\_RECD

**23. What is the advantage when specifying with GRANT option in the GRANT command**

Grant option gives the user permissions to grant the privileges to other users. Privileges**:** select, insert, update, delete

**24. What does the following query do SELECT SAL + NVL(COMM,0) FROM EMP;**

This displays the total salary of all employees. The null values in the commission column will be replaced by 0 and added to salary.

**25. Which system table contains info on constraints on all the tables created obtained** ------ USER\_CONSTRAINTS

**26. State true or false. EXISTS, SOME, ANY are operators in SQL** ------- True

**27. Which command executes the contents of a specified file** ------ START or @

**28. Which command displays the SQL command in the SQL buffer and then executes it** ------- RUN

**29. Which command is used to get back the privileges offered by the GRANT command** ------- REVOKE

**30. What operator performs pattern matching** ------ LIKE operator

**31. What is the use of DESC in SQL**

DESC has two purposes. It is used to describe a schema as well as to retrieve rows from table in DESCending order.  
**Ex:** SELECT \* FROM emp\_order BY emp\_name DESC; will display the output sorted on emp\_name in descending order.

**32. What command is used to create a table by copying the structure of another table**

To copy only the structure, the WHERE clause of the AS SELECT command should contain a FALSE statement with CREATE TABLE command can be use as: CREATE TABLE new\_t\_name AS SELECT \* FROM existing\_t\_name WHERE 1=2;

If the WHERE condition is true, then all the rows or rows satisfying the condition will be copied to new table.

**33. What are the wildcards used for pattern matching** \_ for single char substitution **&…&** % for multichar substitution

**34. What is an SQL injection**

When form data contains an SQL escape sequence and injects a new SQL query to be run is called **SQL injection.**

**35. Difference between TRUNCATE & DELETE**

|  |  |
| --- | --- |
| **TRUNCATE** | **DELETE** |
| TRUNCATE allows deleting entire table and TRUNCATED records cannot rolled back | DELETE allows the **f**iltered deletion so Deleted records can rolled back |
| Database triggers do not fire on TRUNCATE | Database triggers **f**ire on DELETE |

**36. What is the use of the DROP option in the ALTER TABLE command** ----- To drop constraints specified on the table

**37. What operator tests column for the absence of data** --------- IS NULL operator

**38. What are the privileges that can be granted on a table by a user to others**

select**,** Insert, update, delete, alter, references, index, execute, all.

**39. Which function is used to find the largest integer less than or equal to a specific value** ------- FLOOR

**40. Which subset of SQL commands is used to manipulate Oracle Database structures, including tables** ---- DDL

**41. What is the output of the following query SELECT TRUNC (1234.5678, -2) FROM DUAL;** ------- 1200

**42. What is the sub-query --** Sub-query is a query which return values are used in filtering conditions of the main query

**43. What is a join Explain the different types of joins**

The join clause is used to combine or retrieves records from 2 or more tables in a Database. Type of joins are:

* INNER JOIN - An inner join is used to return the matched records from both tables
* LEFT(OUTER) JOIN: Returns all records from the left table and the matched records from the right table
* RIGHT(OUTER) JOIN: Returns all records from the right table, and the matched records from the left table
* FULL(OUTER) JOIN: Returns all records when there is a match in either left and right table

****   

**44. What is correlated sub-query** ------ Correlated sub-query is a sub-query which reference to the main query.

**45. Explain CONNECT BY PRIOR** ------ Retrieves rows in hierarchical order for Ex, Select e\_no, e\_name from emp\_table;

**Hierarchical Queries:** If a table contains hierarchical data, then we can select rows in a hierarchical order by using the hierarchical query clause.

**46. Difference between INSTR and SUBSTR**

The **INSTR** function returns a character position in a pattern of string

**Ex:** INSTR(‘Dot-Net-Funda’,’-’,2) ----- O/P: **8**(i.e., 2nd occurrence of ‘-‘)

The **SUBSTR** function returns a specific portion of string

**Ex:** SUBSTR(‘DotNetFunda’,6) ----- O/P: DotNet

**47. Explain about INTERSECT, MINUS, UNION and UNION ALL**  
INTERSECT - returns all distinct rows selected by both queries.  
MINUS - returns all distinct rows selected by the first query but not by the second query.  
UNION - returns all distinct rows selected by either query  
UNION ALL - returns all rows selected by either query, including all duplicates.

**48. What is ROWID**

**ROWID** is a pseudo-column attached to each row of a table and that uniquely defines a single row in a database table. The pseudo-column is used to refer ROWID in the WHERE clauses of a query as we would refer to a column stored in our database. the difference is we cannot insert, update or delete ROWID values.

**Pseudo-column:** A **pseudo**-**column** behaves like a table **column** but values are not actually stored in the table. We can select values from pseudo-columns but we cannot insert, update or delete their values. A pseudo-column is also similar to a function without arguments.

**49. What is the fastest way of accessing a row in a table** ------- By using ROWID CONSTRAINTS

**50. What is an integrity constraint** ------- Integrity constraint is a rule that restricts values to a column in a table

**51. What is referential integrity constraint**

Maintaining data integrity through a set of rules, that restrict the values of one or more columns of the tables based on the values of primary key or unique key of the referenced table.

**52. Difference between primary key and unique key**

This concept can use to make the relation between tables.

|  |  |
| --- | --- |
| **Primary key** | **Unique key** |
| Primary key doesn’t accept duplicate values and null vals | Unique key not accept duplicate values but accept null vals |
| Only one primary key present in a table | Multiple Unique keys present in a table |

**API Testing**

**@. API**

API stands for Application Programming Interface. It is a set of rules, protocols, and tools that allow different software applications or components are communicate or interact with each other. APIs define how requests and responses should be formatted, enabling seamless integration between systems.

**@. API Testing**

API testing involves testing application programming interfaces directly to validate interactions between components as part of integration testing to determine if they meet expectations for functionality, reliability, performance and security.

**@. Rest API**

REST API follows REST principles using HTTP. CRUD operations map to GET, POST, PUT, DELETE.

**N:** CRUD stands for create, read, update, delete. Four basic operations that define how software application interact with data. Create-post, Read-get, Update-put, patch, delete-delete.

**@. REST Assured**

RestAssured is a java library used for testing and validating RESTful APIs. It simplifies the process of testing APIs by providing a domain-specific language (DSL) for writing tests.

**N:** To setup RestAssured in a project we have to add ‘rest-assured-4.3.3’ dependency to pom.xml

**@. API and Unit Testing**

Unit testing focus on individual components of the software whereas API testing focus on the entire system API, ensuring that endpoints function correctly and meet the requirements.

**@. Common types of API testing**

Common types include Functional testing, Load testing, Security testing, runtime error detection, UI testing, penetration testing and fuzz testing.

**@. Important operations will do in API testing What are HTTP Requests / Main HTTP methods used in Restful APIs**

**GET:** It will get the record/data from the server by passing URL

**POST:** It will go and create a new resource(record) in the server. POST require to pass Request body/ Request Payload. Use of Request Payload is when we create a post request, we need to specify what kind of record going to create in the server. There are two kind of POST requests 1st one is going to create a new record In database and 2nd one is going to validate some data in the database.

**PUT:** If we want to update the record, we need to use PUT request. PUT require to pass Request body/ Request Payload. Use of Request Payload is when we update a PUT request, we need to specify what kind of record going to update in the server.

**DELETE:** It will delete the record/data from server by passing URL

Along these **PATCH**, **OPTIONS** and **HEAD**

**N:** Every request will get response in the JSON format. RESTapi supports multiple formats (HTML, XML, JSON) but SOAP supports only XML

Once we get the response there are certain things we need to focus those are: Response Body (Body), HTTP status code (200, 201 are successful status codes ‘OK’ other than these codes our request is not processed properly because of some reasons), how much time it is taken (Time), how much data we got (size), along with this every response will have some headers (Header), Cookies

Add validation points under Test tab

**N:** API Testing tools: Postman, SoapUI, REST Assure, Swagger and Katalon studio.

**@. PUT and PATCH** Ref: APIPetStoreAutomation\src\test\java\api\test\Z3\_PatchRequestEx

**PUT** fully replaces entire resource (i.e. full resource update). **PATCH** updates specific field to existing resource.

**@. Perform GET request using RestAssured**

Response response = RestAssured.get(“https://api.example.com/resource”);

System.out.println(response.getStatusCode());

**@. To send GET request and validate the status code / Validate the status code of a response**

given().

when().

get(“https://api.example.com/resource”).

then().

assertThat().statusCode(200);

**@. Write POST request code to create new user / Send POST request with a JSON body using RestAssured**

given().

contextType(“application/json”).

body(“{ \”key\” : \”value\”}”).

when().

post(“https://api.example.com/resource”).

then().

statusCode(201);

**@. Status codes**

Status codes are the standard HTTP response codes. A server returns after processing an API request. They tell us whether the request was successful, failed, or needs further action.

Status code 200 means – "OK" - Request was successful

Status code 201 means – "Created" - Request is successful and a new resource was created.

Status code 204 means – "No Content" - Success but no response body

Status code 400 means – "Bad Request" - Server was unable to process a request.

Status code 401 means – "Unauthorized" - Server was unable to identify the user because the request lacked valid authentication credentials.

Status code 404 means – "Page Not Found" error - We requested was not found on the server

Status code 500 means – "internal server error" - Server is unable to fulfil a requirement due to an unexpected condition

**@. Validate response content type using RestAssured**

given().

when().

get(“https://api.example.com/resource”).

then().

assertThat().contentType(“application/json”);

**@. Validate response contains a particular JSON field**

given().

when().

get(“https://api.example.com/resource”).

then().

.body(“$”, “hasKey(“data”)”); (checks if response JSON has a “data” key)

**@. Validate JSON response data / field / response body using RestAssured**

given().

when().

get(“https://api.example.com/resource”).

then().

.assertThat().body(“key”, euqlasTo(“value”));

**@. To log request and response details in RestAssured**

given().

Log().all().

when().

get(“https://api.example.com/resource”).

then().

Log().all();

**@. To log request and response details in RestAssured**

given().

Log().all().

when().

get(“https://api.example.com/resource”).

then().

Log().all();

**@. To Handle basic authentication in RestAssured**

given().

auth().

preemptive().

Basic(“username”, ”password”).

when().

get(“https://api.example.com/secure-resource”).

then().

statusCode(200);

@. Authentication in Rest APIs (Bearer, Basic)

**Bearer:** Uses token in header.

**Basic:** Base64 encoded username::password in header

**@. To handle OAuth2 authentication in RestAssured**

given().

auth().

OAuth2(“our\_access\_token”). //OAuth 2.0 tokens with authorization: Bearer <token> headers

when().

get(“https://api.example.com/secure-resource”).

then().

statusCode(200);

**@. Authentication and Authorization**

Authentication the process of verifying ‘who a user is’. It is to ensure the user is genuine (identity check). (Ex: Login with user name and password).

Authorization the process of verifying ‘what access the user has’ (Ex: Can the user open the admin page?).

**@. Request specification in RestAssured**

Request specification is a reusable set of request configurations like base URI, headers, and authentication that can be applied to multiple requests.

**@. Create and use a Request specification in RestAssured**

RequestSpecification requestSpec = new RequestSpecification().

setBaseUri(“https://api.example.com”).build();

given().

spec(requestSpec).

when().

get(“/resource”).

then().

statusCode(200);

**@. Response specification in RestAssured**

Response specification is a reusable set of response verifications like status code, header and body content that can be applied to multiple responses.

**@. Create and use a Response specification in RestAssured**

ResponseSpecification responseSpec = new ResponseSpecification().

expectStatusCode(200).build();

given().

when().

get(“https://api.example.com/secure-resource”).

then().

spec(responseSpec).

**@. Headers**

In API testing, headers are key–value pairs sent along with the HTTP request (or returned in the HTTP response) that carry important metadata about the request/response. Ex for common Headers: **1)** Content-type: application/json **2)** Authorization: Bearer token123

**@. To add headers to a request in RestAssured**

given().

Header(“Content-Type”, “application/json”)

when().

get(“https://api.example.com/ resource”).

then().

statusCode(200).

@. Path and Query parameters

**Path Parameters:** Variables included as part of the URL path. Used for identifying a specific resource.

**Position in URL:** Before the ?, typically embedded in the URL path.

**Ex:** https://api.example.com/users/123 (Here, 123 is a path parameter indicating a specific user)

**Query Parameters:** k-v pairs appended to the end of a URL. Used for filtering, sorting, or providing additional data

**Position in URL:** After the ? and separated by &.

**Ex:** https://api.example.com/users?role=admin&active=true (Here, role and active are query para’s used to filter users)

**@. Query parameter and Path parameter**

Query parameter will filter the data (?type=admin) whereas Path parameter will get data from the server based on path we specify (/users/{id}).

**@. To add/handle query parameter to/in a request in RestAssured**

given().

queryParam(“Param1”, “value1”) Ex: queryParam(“Page”, 2)

.queryParam(“Param2”, “value2”)

when().

get(“https://api.example.com/ resource”).

then().

statusCode(200);

**@. To send cookies with a request in RestAssured**

given().

cookie(“session-id”, “abc123”)

when().

get(“https://api.example.com/ resource”).

then().

statusCode(200).

**@. Validate response header using RestAssured**

given().

when().

get(“https://api.example.com/ resource”).

then().

assertThat().

header(“Contect-Type”, “application/json”)

**@. Validate a JSON array in the response using RestAssured**

given().

when().

get(“https://api.example.com/ resource”).

then().

body(“data”, hasSize(5));

**@. Extract a value from the JSON response using RestAssured**

Response response =

given().

get(“https://api.example.com/ resource”).

String value = response.jsonPath().getString(“key”);

**@. Validate that a key is present in the JSON response using RestAssured**

given().

when().

get(“https://api.example.com/ resource”).

then().

body(“$”, hasKey(“key”));

**@. Handle nested JSON response in RestAssured**

Parse JSON using jsonPath() and navigate to nested ele’s i.e. response.jsonPath().getString(“data[0].attributes.name”);

(or)

given().

when().get(“https://api.example.com/ resource”).

then().body(“parent.child”, equalTo(“value”));

**@. Validate the response time of an API request using RestAssured**

given().

when().

get(“https://api.example.com/ resource”).

then().

time(lessThan(2000L)); (check if response time is less than 2 sec)

**@. To parse XML response using REST Assured**

given().

when().

get(“https://api.example.com/ resource”).

then().

body(“response.element”, equalTo(“value”));

**@. To validate that a list in a JSON response contains specific values using RestAssured**

given().

when().

get(“https://api.example.com/ resource”).

then().

body(“data”, hasItems(“value1”, “value2”));

**@. H to configure RestAssured to use relaxed HTTPS validation** --- RestAssured.useRelaxedHTTPSValidation();

To reuse request and response specifications across multiple tests in RestAssured

The specifications in a setup method annotated with @BeforeClass (JUnit) or @BeforeAll (TestNG)

**@. File upload in RestAssured**

given().

multiPart(new File(“/path/to/file”)).

when().

post(“https://api.example.com/upload”).

then().

statusCode(200);

**@. To perform multipart from data requests in RestAssured**

given().

multiPart(“formField”, “value”).

multiPart(“fileField”, new File(“/path/to/file”)).

when().

post(“https://api.example.com/resource”).

then().

statusCode(200);

For finding paths ---- json path finder

API documentation --- https://reqres.in/ (diff type of requests will avl username and pass copy from here for POST)

**@. POJO Class**

Plane Old Java Object: Simple class with private fields, public getter / setters, and a no-argument constructor. It is used for sending request body to our API test we have used to POJO and Each request body has specific POJO class.

**@. To write Test cases for API**

1. Understand API documentation: Study endpoints, methods, parameters and response formats

2. Identify Test Scenarios:

Positive: Valid inputs

Negative: Invalid inputs

Edge cases: Unusual inputs or boundary values

3. Define Test Cases:

Test Case ID: Unique identifier

Endpoint: API URL

Method: GET, POST, PUT, DELETE

Headers: Authorization tokens, Context-type

Request body: JSON/XML data

Expected response: Status code, response body

**Ex:** Test GET/users: Verify 200 OK status and correct user details

@. To pass headers in API calls/request using collections

• Use Map<String, String> to store headers

• Easily reusable across requests (auth, content- type)

• Works well with REST Assured .headers() method

@. Payload

In API testing, a payload refers to the actual data we send to the server in the body of an HTTP request (usually in POST, PUT, or PATCH methods). Ex

@. To build dynamic JSON payloads using collections

• Use Map<String, Object> to simulate JSON object, add keys and values dynamically

• Useful for user creation, updates, login payloads

• Work well with POST and PUT requests

• Allows flexible key-value insertions based on Test Case

@. To represent a JSON array of objects in java

• Use List<Map<String, Object>> for multiple user- like objects

• Simulates payloads like [{...}, {...}]

• Useful for batch POST or validation and Ideal for bulk creation APIs or multi-record validation

List Map<String, Object>> users = new ArrayList<>():users.add(user);

@. To extract a list or multiple values from API response

• Use List<String> to collect values using .jsonPath().getList() for list values

• Handy for validating response IDs, emails, names

• Supports direct assertions or further processing

List<String> ids = response.jsonPath().getList("users.id"):

@. To validate uniqueness in API responses

Convert list to Set and compare size i.e. use Set<String> to store values and compare size of Set and List to ensure no duplicates.



@. To manage multiple responses in a test flow

Use Map<String, Response>. It helps to store and reuse responses from different endpoints.

@. Y collections important in API automation framework

* Enable dynamic request generation
* Help in building DD tests
* Useful for parsing and validating complex JSON responses
* Support clean and reusable code

**@. Postman used for** --- Postman is a GUI tool to test APIs, automate requests, and validate responses.

**@. To write RestAssured script** ---- Use given().when().get().then().assertThat().statusCode(200); syntax

@. To validate response body and headers

Use body() for content checks and header() for header validation in RestAssured.

@. If API returns 500 error, H to debug

Check request payload, headers, and try the same in postman, or review backend logs.

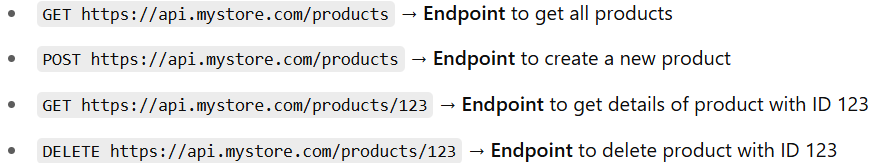
@. To design a reusable test utility for API testing

Build a utility layer using REST Assured, encapsulating common methods for GET, POST, PUT, DELETE. Parameterize endpoints, support dynamic headers/auth, and make it easy to chain requests. This helps reduce duplication and improve maintainability.

@. Endpoint

In API testing, an endpoint refers to a specific URL at which an API can be accessed by a client application. Ex below img.

W to test at an endpoint: When doing API testing, we send requests to these endpoints and check:

* Response status codes (200, 404, 500, etc.)
* Response data (is the returned data correct and in the right format?)
* Headers (content type, authorization, etc.)
* Error handling (does it handle invalid inputs properly?)
* Performance (response time)
* Security (authentication, authorization)

@. Validations put in API testing ---- Status code, Response time, Schema validation, Field presence, Business logic.

@. To send form parameters in RestAssured

given().

formParam(“Param1”, “value1”) Ex: queryParam(“Page”, 2)

. formParam (“Param2”, “value2”)

when().

post(“https://api.example.com/ resource”).

then().

statusCode(200);

@. To check response JSON path returns non-null value

given()

when().get(“https://reqres.in/api/users/2”)

then().body(“data.email”, notNullValue());

@. Structure of HTTP request and response

Request – Methods (GET, POST, PUT, etc), URL, Headers, and (optionally) body

Response – Status code, Headers, and body (often JSON, or XML)

@. To automate API testing in java

Use RestAssured library combined with TestNG for validations. By this create requests and validate response like status code, and headers.

@. To Authenticate APIs

Depending on the API, we can use Basic Auth, Bearer Tokens (QAuth 2.0), API keys, or custom headers. In ResrAssured it can be set easily using .auth() methods.

@. To validate API response

Check the status code, response time, content type, headers, and body fields using Assertion methods.

@. JSON, to validate JSON using RestAssured

JSON (JavaScript Object Notation) is a light weight data format with key-value pairs for API requests and responses. In RestAssured, we validate JSON using body() method with matches like equalTo() or by parsing and asserting values.

@. Serialization and Deserialization

Serialization is converting a java object into JSON or XML. Deserialization is a opposite process of Serialization. Here will use Jackson library.

@. Some examples of API testing in project / API testing experience

I test user authentication, courses list endpoints, payment APIs, and verify database updates after hitting specific end points. I also validate negative scenarios and security aspects.

We automate E2E API testing by sending different type of requests (GET, POST, PUT, DELETE) using RestAssured. We validate responses by checking status codes, headers, and body fields. We also handle authentication tokensdynamically and use POJOs for serialization/deserialization to ensure clean code structure.

@. To log request and response

By using log().all() for requests and .then().log().all() for responses to get full visibility.

@. To extract value form JSON response ---- String value = response.jsonPath().getString(“data.id”);

@. To test file upload

given()

.multiPart(“file”, new File(“path/to/file”))

.when()

.post(“/upload”).then().statusCode(200);

@. To use Postman for testing

We use Postman for quick manual API validation and environment management. Once the end points are understood, we automate tests using RestAssured with Jackson for scalable, maintainable testing.

@. HTTP and HTTPS

Imagine HTTP is like sending a postcard-anyone handling it can read your message. HTTPS is like sending a locked briefcase-only the recipient has the key.

Technically, HTTPS encrypts data using SSL/TLS certificates, while HTTP sends everything as plain text. As testers, we always check for 'https://' and the padlock icon, especially on login pages.

@. Describe how a client and server interact? Think of it like ordering food:

Client (you): Asks for a burger by clicking a 'Submit Order' button.

Server (kitchen): Receives the request, makes the burger, and sends it back.

In testing, I verify both sides-like checking if the button works (client UI) and if the order appears in the database (server API).

@. To verify a site uses HTTPS for forms

• Manual Check: Submit a form and look for 'https://' in the URL bar and the padlock icon.

• Aut Test: In Sele, we assert the curr URL starts with 'https': assertTrue(driver.getCurrentUrl().startsWith("https://"));

@. First thing we check in API testing

I start with the status codes-like a health report: 200 OK means 'good job, API!' 401 Unauthorized means 'you forgot credentials.' For example, if our login API returns 200, I then validate the response body contains the user's token.

@. To test a new API endpoint?"

I follow the 'happy path, sad path, bad path' approach:

Happy Path: Valid request 200 OK+ correct response.

Sad Path: Invalid input (e.g., wrong email format) → 400 Bad Request.

Bad Path: Malicious input (e.g., SQL injection) → 403 Forbidden.

@. Object Mapper? Why do you use it?

Object Mapper is like a translator between JSON/XML (API responses) and Java objects. For example, when an API returns this JSON: {"name":"John","age":30}

An Object Mapper (like Jackson) converts it to a Java Person class automatically:

Person person = objectMapper.readValue(json, Person.class);

// Now person.getName()→ "John"

We use it to avoid manual parsing-saving time and reducing errors.

@. To format a Date field to 'yyyy-MM-dd' in JSON?

With @JsonFormat:

public class Event {

}

@JsonFormat(pattern="yyyy-MM-dd") private Date eventDate;

Or globally:

objectMapper.setDateFormat(new SimpleDateFormat("yyyy-MM-dd")); This ensures consistency across all API responses.

@. Test an API that has no documentation

Use tools like Postman or browser dev tools to inspect API traffic. Make educated guesses based on endpoints. Use trial and error to explore behaviour.

End#4