**JAVA**

**JDK**

JDK stands for Java Development Kit.

JDK includes JRE and development tools.

**JRE**

JRE stands for Java Runtime Environment.  
Using JRE, we can only execute already developed applications. We cannot develop new applications or modify existing applications.

**JVM**

JVM stands for Java Virtual Machine. JVM drives the java code. Using JVM, we can run java byte code by converting them into machine language.

Compilation and execution of a Java program is two step process. During compilation phase Java compiler compiles the source code and generates *bytecode*. This intermediate *bytecode* is saved in form of a .class file. In second phase, Java virtual machine (JVM) also called Java interpreter takes the .class as input and generates output by executing the *bytecode*. Java is an object oriented programming language; therefore, a program in Java is made of one or more classes. No matter how trivial a Java program is, it must be written in form of a class.

In java we have two categories of data type: 1) Primitive data types 2) Non-primitive data types.

Eight primitive data types: boolean, char, byte, short, int, long, float and double

Arrays and Strings are non-primitive data types.

**byte**, **short**, **int** and **long** data types are used for storing whole numbers.

**float** and **double** are used for fractional numbers.

**char** is used for storing characters(letters).

**boolean** data type is used for variables that holds either true or false.

|  |  |  |
| --- | --- | --- |
| **Data Type** | **Size** | **Description** |
| byte | 1 byte | Stores whole numbers from -128 to 127 (1 byte =8 bits) |
| short | 2 bytes | Stores whole numbers from -32,768 to 32,767 |
| int | 4 bytes | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| long | 8 bytes | Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807 |
| float | 4 bytes | Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits |
| double | 8 bytes | Stores fractional numbers. Sufficient for storing 15 decimal digits |
| boolean | 1 bit | Stores true or false values |
| char | 2 bytes | Stores a single character/letter or ASCII values |

Stack & Heap:

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM .

Variables allocated on the stack are stored directly to the memory and access to this memory is very fast, and it's allocation is dealt with when the program is compiled. When a function or a method calls another function which in turns calls another function etc., the execution of all those functions remains suspended until the very last function returns its value. The stack is always reserved in a LIFO order, the most recently reserved block is always the next block to be freed.

It's a special region of your computer's memory that stores temporary variables created by each function (including the main() function). The stack is a "LIFO" (last in, first out) data structure, that is managed and optimized by the CPU quite closely. Every time a function declares a new variable, it is "pushed" onto the stack. Then every time a function exits, **all** of the variables pushed onto the stack by that function, are freed (that is to say, they are deleted). Once a stack variable is freed, that region of memory becomes available for other stack variables.

A key to understanding the stack is the notion that when a function exits, all of its variables are popped off of the stack (and hence lost forever). Thus stack variables are local in nature. This is related to a concept we saw earlier known as variable scope, or local vs global variables

Variables allocated on the heap have their memory allocated at run time and accessing this memory is a bit slower, but the heap size is only limited by the size of virtual memory . Element of the heap have no dependencies with each other and can always be accessed randomly at any time. You can allocate a block at any time and free it at any time.

**String**:

Strings are immutable and final in Java. Immutable simply means unmodifiable or unchangeable.

An immutable object is an **object whose internal state remains constant after it has been entirely created**.  
This means that once the object has been assigned to a variable, we can neither update the reference nor mutate the internal state by any means. String pool (String intern pool) is a special storage are a in [Method Area](https://www.programcreek.com/2013/04/jvm-run-time-data-areas/). **When a string is created and if the string already exists in the pool, the reference of the existing string will be returned, instead of creating a new object.**

There are two ways to create a String in Java

1. Using String literal :  
   String s="Welcome";
2. Using new keyword  
   String s=new String("Welcome");

In the heap memory, JVM allocates some memory specially meant for string literals.   
This part of the heap memory is called **String Constant Pool**.  
Whenever we create a string object using string literal, that object is stored in the **string constant pool** and whenever you create a string object using new keyword, such object is stored in the heap memory.

For example, when you create string objects like below, they will be stored in the String Constant Pool.

?

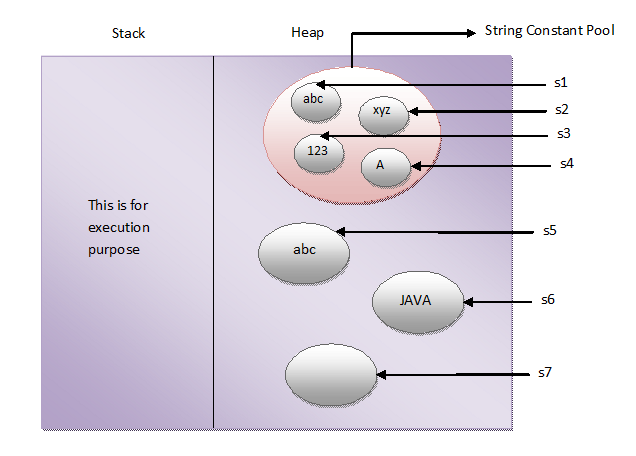
|  |  |
| --- | --- |
|  | String s1 = "abc";    String s2 = "xyz";    String s3 = "123";    String s4 = "A"; |

And when you create string objects using new keyword like below, they will be stored in the heap memory.

?

|  |  |
| --- | --- |
|  | String s5 = new String("abc");    char[] c = {'J', 'A', 'V', 'A'};    String s6 = new String(c);    String s7 = new String(newStringBuffer()); |

This is how String Constant Pool looks like in the memory.P**ool space is allocated to an object depending upon it’s content**. There will be no two objects in the pool having the same content.

****

**When you create a string object using string literal, JVM first checks the content of the object to be created. If there exist an object in the pool with the same content, then it returns the reference of that object. It doesn’t create new object. If the content is different from the existing objects then only it creates new object.**

**But, when you create string objects using new keyword, a new object is created whether the content is same or not.**

Since *Strings* are immutable in Java, the JVM optimizes the amount of memory allocated for them by storing only **one copy of each literal *String* in the pool**. This process is called interning:

|  |  |
| --- | --- |
|  | String s1 = "Hello World"; |
|  | String s2 = "Hello World"; |
|  |  |
|  | assertThat(s1 == s2).isTrue(); |

Public class StringExamples

{

    Public static void main(String[] args)

    {

        //Creating string objects using literals

        String s1 = "abc";

        String s2 = "abc";

        System.out.println(s1 == s2);        //Output : true

        //Creating string objects using new operator

        String s3 = new String("abc");

        String s4 = new String("abc");

        System.out.println(s3 == s4);        //Output : false

    }

}

* == is a reference comparison, i.e. both objects point to the same memory location
* .equals() evaluates to the comparison of values in the objects

**Variables:**

**1.Local variables:**

These variables are declared inside method of the class. Their scope is limited to the method.  
You can’t change their values and access them outside of the method.  
These variable are created when the the function is called and destroyed after exiting from the block or when the call returns from the function.

The scope of these variables exists only within the block in which the variable is declared. i.e. we can access these variable only within that block.

**2.Instance Variables**

Instance variables are declared in a class, but outside a method.  
As instance variables are declared in a class. These variables are created when an object of the class is created, destroyed when the object is destroyed.

Unlike local variables, we may use access specifiers for instance variables. If we do not specify any access specifier then the default access specifier will be used.

**3.Static variables**

**S**tatic variables are declared with the static keyword in a class, but outside a method

There would only be one copy of each class variable per class, regardless of how many objects are created from it.

**Access Modifiers** − default, public , protected, private

**Non-access Modifiers** − final, abstract, strictfp

**Continue statement** is mostly used inside loops. Whenever it is encountered inside a loop, control directly jumps to the beginning of the loop for next iteration, skipping the execution of statements inside loop’s body for the current iteration

The remaining statements in the loop are skipped. The execution starts from the top of the loop again. We can use continue statement to skip current iteration and continue the next iteration inside loops.

**Break Statement:**Use break statement to come out of the loop instantly. Whenever a break statement is encountered inside a loop, the control directly comes out of loop and the loop gets terminated for rest of the iterations.

**i++ and ++i;**

**i++ and ++i means i=i+1; so the value of i gets incremented. Only difference is assignment of value.**

**Example 1.**

**i=3;**

**a=i++; a=3; i=4;**

**Example 2.**

**i=3;**

**a==++i; a=4;i=4**

**Package**

**Class**

**Type Casting.**

Assigning a value of one type to a variable of another type is known as **Type Casting**.

 Type casting is classified into two types.

**Widening (Implicit)**

the two types are compatible  
the target type is larger than the source type.

assigning a smaller type value to a variable of larger type

int i =100;

long l = i; //no explicit type casting required

float f = l;

**Narrowing Casting(Explicitly done)**

When you are assigning a larger type value to a variable of smaller type, then you need to perform explicit type casting.

Double d=100.04;

long l=(long)d;//explicit type casting required

int i=(int)l;

Type casting in Java is to cast one type, a class or interface, into another type i.e. another class or interface  
you need type casting to get access to fields and methods declared on target type or class. You can not access them with any other type.  
We have two classes, Base, and Derived. Derived class extends Base i.e. Base is a Super class and Derived is a Subclass. So their type hierarchy looks like the following tree :

Parent

 |

Child

Parent p = **new** Child(); *//reference variable of Base class points object of Derived class*

Casting a subtype to a super type is called upcasting.

Upcasting can be done implicitly.

Another way:

Child c =new Child();

Parent p=(Parent)c;

**Parameter** is a variable defined by a method that receives value when the method is called. Parameter are always local to the method they dont have scope outside the method. While **argument** is a value that is passed to a method when it is called.

UpCast ->From lower level class type to upper level class type

Downcast ->From upper level class type to lover level class type

**Enum:**

In applications, you often need to work with a set of constant values. For example, representing a contract status with the “permanent”, “temp”, and “intern” values, or directions with the “north”, “south”, “east”, and “west” values.

A *Java Enum* is a special Java type used to define collections of constants.

public enum Level {

HIGH,

MEDIUM,

LOW

}

Level level = Level.HIGH;

**Call By Value & Call By Reference:**

1. **call-by-value :** In this approach copy of an argument value is pass to a method. Changes made to the argument value inside the method will have no effect on the arguments.
2. **call-by-reference :** In this reference of an argument is pass to a method. Any changes made inside the method will affect the agrument value.

**NOTE :**There is only call by value in java, not call by reference.

**Constructor** a special type of method that is used to initialize the object.  
constructor is invoked at the time of object creation.   
It constructs the values i.e. provides data for the object that is why it is known as constructor.  
Every class has a constructor. If we do not explicitly write a constructor for a class the java compiler builds a default constructor for that class. This constructor is known as default constructor. You would not find it in your source code(the java file) as it would be inserted into the code during compilation and exists in .class file.

Types of Constructor:

Default   
Parameterized / overloaded

There are basically two rules defined for the constructor.

* Constructor name must be same as its class name.
* Constructor must have no explicit return type.
* A constructor in Java cannot be abstract, final, static and Synchronized.
* Constructor is called only once at the time of Object creation while method(s) can be called any numbers of time.
* A constructor is called automatically when a new instance of an object is created.
* Access modifiers can be used in constructor declaration to control its access.

Synchronization:

In multi-threaded environment *synchronization of Java object or synchronization of Java class becomes extremely important*. Synchronization in Java is possible by usingJava keywords ***"synchronized"*** and ***"volatile”***.   
  
1)The synchronized keyword in Java provides locking, which ensures mutually exclusive access to the shared resource and prevents data race.  
  
2) synchronized keyword also prevent reordering of code statement by the compiler which can cause a subtle concurrent issue if we don't use synchronized or volatile keyword.  
  
3) synchronized keyword involve locking and unlocking. before entering into **synchronized method or block** thread needs to acquire the lock, at this point it reads data from main memory than cache and when it release the lock, it flushes write operation into main memory which eliminates memory inconsistency errors.

**Access Specifier / Access Modifier**

Visible to the package, the **default**. No modifiers are needed.

Visible to the class only (private).

Visible to the world (public).

Visible to the package and all subclasses (protected).

Static:

A *static* member is a member of a class that isn’t associated with an instance of a class. Instead, the member belongs to the class itself.

In the Java programming language, **the keyword**static**indicates that the particular member belongs to a type itself, rather than to an instance of that type.**This means that only one instance of that static member is created which is shared across all instances of the class**.**

The static keyword in java is used for memory management mainly. We can apply java static keyword with variables, methods, blocks and nested class.

Memory for static variable is created only once in the program at the time of loading of class. These variables are preceded by static keyword. Static variable can access with class reference.

**if a field is declared static, then exactly a single copy of that field is created and shared among all instances of that class**. It doesn’t matter how many times we initialize a class; there will always be only one copy of static field belonging to it. The value of this static field will be shared across all object of either same of any different class.

Static field: A field that’s declared with the static keyword, like this:

private static int ballCount;

**Static method**: A method declared with the static keyword.   
Like static fields, static methods are associated with the class itself, not with any particular object created from the class. As a result, you don’t have to create an object from a class before you can use static methods defined by the class.

The best-known static method is main, which is called by the Java runtime to start an application. The main method must be static, which means that applications run in a static context by default.

One of the basic rules of working with static methods is that you can’t access a non-static method or field from a static method because the static method doesn’t have an instance of the class to use to reference instance methods or fields.

**OOPS Concepts: Abstraction, Encapsulation, Inheritance, Polymorphism**

**Inheritance**:

Inheritance can be defined as the process where one class acquires the properties (methods and fields) of another. With the use of inheritance the information is made manageable in a hierarchical order.

The class which inherits the properties of other is known as subclass (derived class, child class) and the class whose properties are inherited is known as superclass (base class, parent class).

ClassEmployee {

.....

.....

}

ClassDeveloperextendsEmployee{

.....

.....

}

**The super and this keyword**

**this:**

* The keyword ‘this’ is used to represent current object of the class. It is used to access various elements of the class like instance variables and methods. For eg: “this.var1” and “this.method1()”
* The keyword ‘this’ when used along with parenthesis as in ‘this()’, is used to invoke the constructor of the class
* ‘this’ keyword is used to differentiate the instance variable from the local variable, when both use the same name.
* ‘this’ keyword can be passed as an argument to a method or a constructor and can even be used as a return parameter after a method execution like “return this;”

‘this’ used to refer the current instance variable

**class** Box {

**double** height, width;

Box(**double** height, **double** width) {

**this**.height = height; //this.height represents the instance variable ‘height’ of the Box class

**this**.width = width; //height,weight represents the value of the local variable given in the parameter

}

**void** display() {

System.***out***.println("Width:"+width+"\t"+"Height:"+height);

}

**Public static void** main(String args[])

Box b = **new** Box(3.0,5.0);

b.display();

}

}

‘this()’ used to invoke constructor of the current class

**class** Box {

**double** height, width;

Box() {

System.***out***.println("Object created…");

System.***out***.println("Width:"+width+"\t"+"Height:"+height);

}

Box(**double** h, **double** w) {

**this**();

height = h;

width = w;

}

**Public static void** main(String args[])

{

Box b = **new** Box(3.0,5.0);

}

}

The super keyword is similar to this keyword. Following are the scenarios where the super keyword is used.

It is used to differentiate the members of superclass from the members of subclass, if they have same names.

It is used to invoke the superclass constructor from subclass.

Compare 'this' vs 'super' keyword.

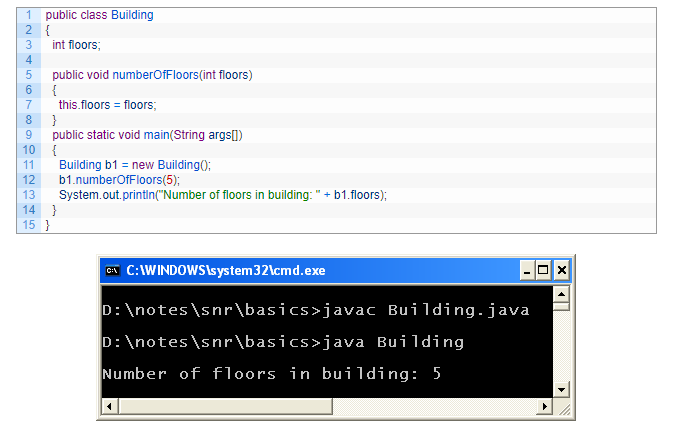
|  |  |  |
| --- | --- | --- |
| **'this' keyword** | | **'super' keyword** |
| It represents the current instance of a class. | | It represents the current instance of a parent class. |
| It is used to call default constructor of the same class. | | It is used to call default constructor of the parent class. |
| It is used to access methods of the current class. | | It is used to access methods of the base class. |
| It is used for pointing the current class instance. | | It is used for pointing the super class instance. |
| Class SuperClass  {      Int i;    //Field        SuperClass(int j)      {          System.out.println("Super Class Constructor");      }        Void methodOfSuperClass()     //method      {          System.out.println("From method of super class");      }  }    Class SubClass extends SuperClass  {      SubClass()      {          super(10);          //Calling statement to super class constructor      }        Void methodOfSubClass()      {          System.out.println(super.i);  //super class field is accessed          super.methodOfSuperClass();  // super class method is called          System.out.println("From method of sub class");      }  } | | |

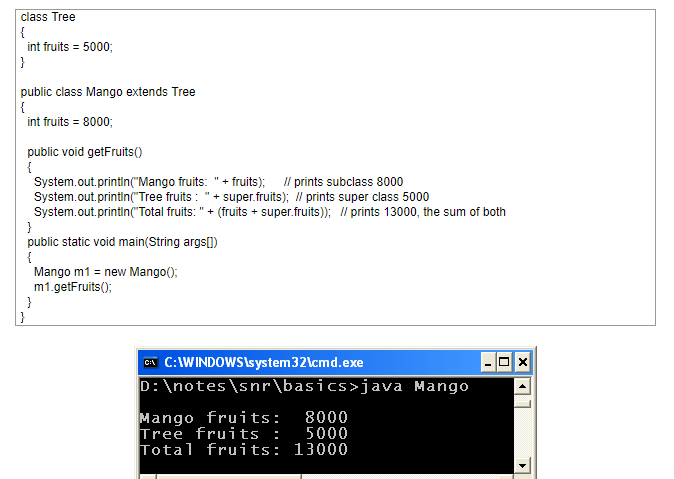
super class constructor is called by **super()** calling statement.You can’t use super() calling statement outside the constructor. By default, super() calling statement is the first statement in any constructor. this keyword is used to access other members of the same class. Using this keyword, you can access methods, fields and constructors of the same class within the class. this refers to current instance of the class.

|  |  |
| --- | --- |
|  | Class AnyClass  {      Int i;       AnyClass()      {        System.out.println("First Constructor");      }      AnyClass(int j)      {          this();    //calling statement to First Constructor          System.out.println("Second Constructor");      }        VoidmethodOne()      {          System.out.println("From method one");      }        VoidmethodTwo()      {          System.out.println(this.i);  //Accessing same class field          this.methodOne();      //Accessing same class method      }  } |

this() is the calling statement to same class constructor. It must be used within constructor only. If it is used, it must be the first statement in the constructor.

super() should be **first** statement inside any constructor. It can be used **only inside constructor** and nowhere else. super() is used to refer **only parent class’s(super class’s) constructor**.





Java - Data Structures

The Stack :

The Stack class implements a last-in-first-out (LIFO) stack of elements.  
The last element you added to the stack is the first one to come back off.

SET & ITERATOR:

Set setA = new HashSet();

setA.add("element 0");

setA.add("element 1");

setA.add("element 2");

//access via Iterator

Iterator iterator = setA.iterator();

while(iterator.hasNext(){

String element = (String) iterator.next();

}

//access via new for-loop

for(Object object : setA) {

String element = (String) object;

}

Generic Sets

Set<MyObject> set = new HashSet<MyObject>();

Platform independence means that you can run the same Java Program in any Operating System. For example, you can write java program in Windows and run it in Mac OS.

Java Virtual Machine (JVM) is the heart of java programming language. JVM is responsible for converting byte code into machine readable code. JVM is not platform independent, thats why you have different JVM for different operating systems.

Java Development Kit (JDK) is for development purpose and JVM is a part of it to execute the java programs.

JDK provides all the tools, executables and binaries required to compile, debug and execute a Java Program. The execution part is handled by JVM to provide machine independence.

When we have more than one method with same name in a single class but the arguments are different, then it is called as method overloading.

Overriding concept comes in picture with inheritance when we have two methods with same signature, one in parent class and another in child class. We can use @Override annotation in the child class overridden method to make sure if parent class method is changed, so as child class.

we can have multiple methods with name “main” in a single class. However if we run the class, java runtime environment will look for main method with syntax as public static void main(String args[])

Acess modifier:

Java Access Modifiers – public keyword

If class member is “public” then it can be accessed from anywhere. The member variable or method is accessed globally. This is simplest way to provide access to class members, however we should take care in using this keyword with class variables otherwise anybody can change the values. Usually class variables are kept as private and getter-setter methods are provided to work with them.

Java Access Modifiers – private keyword

If class member is “private” then it will be accessible only inside the same class. This is the most restricted access and the class member will not be visible to the outer world. Usually we keep class variables as private and methods that are intended to be used only inside the class as private.

Java Access Modifiers – protected keyword

If class member is “protected” then it will be accessible only to the classes in the same package and to the subclasses. This modifier is less restricted from private but more restricted from public access. Usually we use this keyword to make sure the class variables are accessible only to the subclasses.

Java Access Modifiers – default access

If class member doesn’t have any access modifier specified, then it’s treated with default access. The access rules are similar as classes and the class member with default access will be accessible to the classes in the same package only. This access is more restricted than public and protected but less restricted than private.

**Encapsulation : (access modifiers)**

Process of wrapping of *data* and *methods* in a single unit is called encapsulation.

The whole idea behind **encapsulation is to hide the implementation details from users.** If a data member is private it means it can only be accessed within the same class. No outside class can access private data member (variable) of other class.

 it is a protective shield that prevents the data from being accessed by the code outside this shield.

How to implement encapsulation in java:  
1) Make the instance variables private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class.  
2) Have getter and setter methods in the class to set and get the values of the fields.

Difference between Encapsulation and Abstraction

Public class EncapTest{

Private int age;

Public int getAge(){

return age;

}

}

**POLYMORPHISM:**

polymorphism is the ability by which, we **can create functions or reference variables which behaves differently in different context**.  
Polymorphism is an object-oriented programming concept that refers to the ability of a variable, function or object to take on multiple forms

Polymorphism in Java has two types:

Compile time polymorphism (Static binding) Eg: Method Overloading

Runtime polymorphism (Dynamic binding). Eg: Method Overriding

**Overloading**

*(Same method name with different signature)*

When a Java program contains more than one method with the same name but different properties (signature), then it is called method overloading. In Java, static polymorphism is achieved through method overloading. Method overloading means there are several methods present in a class having the same name but different types/order/number of parameters(signature).

At compile time, Java knows which method to invoke by checking the method signatures.  So, this is called **compile time polymorphism** or **static binding**.

**Method overloading doesnt depend upon return type.**

Depends on below

Number of parameters passed

Data type of actual parameters

Sequence of data type of actual parameters

**Overriding**

**Same method name and signature, one on parent class another on child class.**

Overriding occurs when there are two methods of same name and properties, but one is in child class and one is in parent class.

1. If you try to make child class private then it will give compile time error.
2. **you can not override private methods in parent class.** If subclass tries to override the private method of parent class then program will throw compile time error as shown below.

class Animal {

public void move() {

System.out.println("Animals can move");

}

}

class Dog extends Animal {

public void move() {

System.out.println("Dogs can walk and run");

}

}

public class TestDog {

public static void main(String args[]) {

Animal a = new Animal(); // Animal reference and object

Animal b = new Dog(); // Animal reference but Dog object

a.move(); // runs the method in Animal class

b.move(); // runs the method in Dog class

}

}

Animals can move

Dogs can walk and run

In the above example, you can see that even though **b** is a type of Animal it runs the move method in the Dog class. The reason for this is: In compile time, the check is made on the reference type. However, in the runtime, JVM figures out the object type and would run the method that belongs to that particular object.

[Live Demo](http://tpcg.io/VHj8iU)

class Animal {

public void move() {

System.out.println("Animals can move");

}

}

class Dog extends Animal {

public void move() {

System.out.println("Dogs can walk and run");

}

public void bark() {

System.out.println("Dogs can bark");

}

}

public class TestDog {

public static void main(String args[]) {

Animal a = new Animal(); // Animal reference and object

Animal b = new Dog(); // Animal reference but Dog object

a.move(); // runs the method in Animal class

b.move(); // runs the method in Dog class

b.bark();

}

}

This program will throw a compile time error since b's reference type Animal doesn't have a method by the name of bark.

**Inheritance**

It is the mechanism in java by which one class is allowed to inherit the features(fields and methods) of another class.

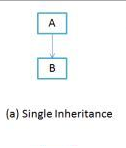
Super Class: The class whose features are inherited is known as super class(or a base class or a parent class).

Sub Class: The class that inherits the other class is known as sub class(or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the superclass fields and methods.

The keyword used for inheritance is **extends**.

Types of Inheritance:

**Single Inheritance**



**Class** Employee{

**float** salary=40000;

}

**class** Programmer **extends** Employee{

**int** bonus=10000;

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

   Programmer p=**new** Programmer();

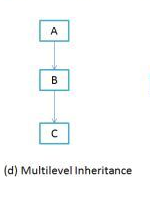
  System.out.println("Programmer salary is:"+p.salary);

System.out.println("Bonus of Programmer is:"+p.bonus);  }

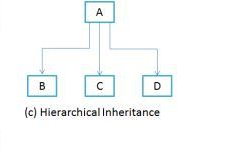
}

**Multilevel Inheritance**

In this inheritance, a derived class is created from another derived class.



**Hierarchical Inheritance**



one class is inherited by many**sub classes**. In below example class B,C and D **inherits** the same class A.

A is **parent class (or base class)** of B,C& D

[Multiple Inheritance](https://www.geeksforgeeks.org/java-and-multiple-inheritance/)**(Through Interfaces) :**In Multiple inheritance ,one class can have more than one superclass and inherit features from all parent classes. Please note that Java does **not** support [multiple inheritance](https://www.geeksforgeeks.org/java-and-multiple-inheritance/) with classes. In java, we can achieve multiple inheritance only through [Interfaces](http://quiz.geeksforgeeks.org/interfaces-in-java/).

**Abstraction** :  
**Abstraction is a process of hiding the implementation details from the user**. Оnly the functionality will be provided to the user.

Abstract Class-Incomplete class

 In more simple terms, abstraction is to hide information that is not relevant or rather show only relevant information and to simplify it by comparing it to something similar in the real world.

**Abstract method & Class**

**A class that is declared using “abstract” keyword is known as abstract class**. It can have abstract methods(methods without body) as well as concrete methods (regular methods with body).   
A normal class(non-abstract class) cannot have abstract methods.

A class derived from the abstract class must implement all those methods that are declared as abstract in the parent class.

Abstract class cannot be instantiated which means you cannot create the object of it. To use this class, you need to create another class that extends this class and provides the implementation of abstract methods, then you can use the object of that child class to call non-abstract methods of parent class as well as implemented methods(those that were abstract in parent but implemented in child class).

We cant create an object of abstract class because these classes are incomplete, they have abstract methods that have no body so if java allows you to create object of this class then if someone calls the abstract method using that object then What would happen?There would be no actual implementation of the method to invoke.  
Also because an object is concrete. An abstract class is like a template, so you have to extend it and build on it before you can use it.

1) Abstract method has no body.  
2) Always end the declaration with a **semicolon**(;).  
3) It must be overridden. An abstract class must be extended and in a same way abstract method must be overridden.  
4) A class has to be declared abstract to have abstract methods.

A method without body is known as abstract method.   
If a class has an abstract method, class should be declared abstract as well.

Public abstract int myMethod(int n1,int n2);

**abstract** **class** Shape{

**abstract** **void** draw();

}

**class** Rectangle **extends** Shape{

**void** draw(){  
System.out.println("drawing rectangle");  
 }

}

**class** Circle **extends** Shape{

**void** draw(){  
System.out.println("drawing circle");  
}

}

**class** TestAbstraction1{

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

Shape s=**new** Circle();

s.draw();

}

}

An abstract class is a class that is declared with [abstract keyword.](https://www.geeksforgeeks.org/abstract-keyword-in-java/)

An abstract method is a method that is declared without an implementation.

An abstract class may or may not have all abstract methods. Some of them can be concrete methods

A method defined abstract must always be redefined in the subclass, thus making [overriding](http://contribute.geeksforgeeks.org/overriding-in-java/) compulsory OR either make subclass itself abstract.

Any class that contains one or more abstract methods must also be declared with abstract keyword.

There can be no object of an abstract class.That is, an abstract class can not be directly instantiated with the [*new operator*](https://www.geeksforgeeks.org/new-operator-java/).

**Interface**: To achieve total abstraction

An interface can have methods and variables, but the methods declared in interface are by default abstract (only method declaration, no body).   
The interface in java is **a mechanism to achieve abstraction**. There can be only abstract methods in the java interface not method body. It is used to achieve abstraction and multiple inheritance in Java.

interface<interface\_name>{

// declare constant fields

// declare methods that abstract

// by default.

}

To declare an interface, use **interface** keyword. It is used to provide total abstraction. That means all the methods in interface are declared with empty body and are public and all fields are public, static and final by default. A class that implement interface must implement all the methods declared in the interface. To implement interface use **implements** keyword.

**interface** printable{

**void** print();

}

**class** A6 **implements** printable{

**public** **void** print(){  
System.out.println("Hello");  
}

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

A6 obj = **new** A6();

obj.print();

  }

}

Why do we use interface ?

It is used to achieve total abstraction.

Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance .

It is also used to achieve loose coupling.  
**Loose coupling :**In simple words, loose coupling means they are mostly independent. If the only knowledge that class A has about class B, is what class B has exposed through its interface, then class A and class B are said to be loosely coupled.

Interfaces are used to implement abstraction. So the question arises why use interfaces when we have abstract classes?

The reason is, abstract classes may contain non-final variables, whereas variables in interface are public,static and final.

An interface is different from a class in several ways, including −

You cannot instantiate an interface.

An interface does not contain any constructors.

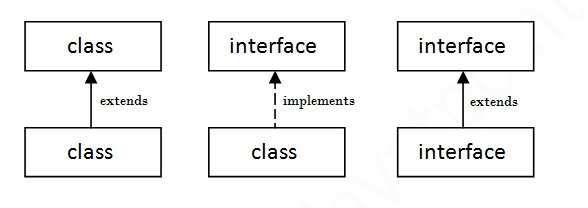
All of the methods in an interface are abstract.

An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.

An interface is not extended by a class; it is implemented by a class.

An interface can extend multiple interfaces.

Abstract class VS Interface



Modifiers in Java :

1. Access control modifier
2. Non Access Modifier

Java language has four access modifier to control access levels for classes, variable methods and constructor.

* **Default :** Default has scope only inside the same package
* **Public :** Public has scope that is visible everywhere
* **Protected :** Protected has scope within the package and all sub classes
* **Private :** Private has scope only within the classes

Non-access modifiers do not change the accessibility of variables and methods, but they do provide them special properties. Non-access modifiers are of 5 types,

1. Final
2. Static
3. Transient
4. Synchronized
5. Volatile

**Final Variable**

Final keyword in java is used to restrict usage of variable, class and method.

When a variable is declared as final, then its value cannot be changed. The variable acts like a constant.

Syntax:

final int a = 5;

**Final Method**  
When a method is declared as final, then that method cannot be overridden.

Class StudyTonight

{

Final void learn()

{

System.out.println("learning something new");

}

}

Class Student extends StudyTonight

{

Void learn()

{

System.out.println("learning something interesting");

}

Public static void main(String args[]){

Student object=new Student();

object.learn();

}

}

This will give a compile time error because the method is declared as final and thus, it cannot be overridden.

**Note:** A final method can be inherited/used in the subclass, but it cannot be overriden.

A class can also be declared as final. A class declared as final cannot be inherited. **String** class in java.lang package is a example of final class.

**Throw, Throws and Throwable.**

The throw keyword in Java is used to explicitly throw an exception from a method or any block of code. We can throw either checked or unchecked exception. The throw keyword is mainly used to throw custom exceptions.  
Syntax:

Throw:

* throw is a keyword in java
* throw is used to throw exception, from method or executable block.

Public int getAge(){

if(age <0)

throw new RuntimeException("Age cannot be negative");

return age;

}

**Throws:**

Throws is a keyword in Java which is used in the signature of method to indicate that this method might throw one of the listed type exceptions. The caller to these methods has to handle the exception using a try-catch block.

* throws is also keyword in java
* throws keyword is used in conjunction with method signature.
* throws signifies, the kind of exception, the method can throw.
* **type method\_name(parameters) throws exception\_list**
* exception\_list is a comma separated list of all the exceptions which a method might throw.

Public int getAge()throws RuntimeErrorException{

if(age <0)

throw new RuntimeException("Age cannot be negative");

return salary;

}

Eg: InterruptedException -very often used in thread operations. Thread.Sleep(2000)  
FileNotFoundException, IOException

**Throwable**

* Throwable is super class of all exceptions and errors.
* All kind of exceptions and error are kind of throwable type.

**Exceptions**:

An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program’s instructions.

All exception and errors types are sub classes of class **Throwable**, which is base class of hierarchy.Exception class is used for exceptional conditions that user programs should catch. NullPointerException is an example of such an exception.Anotherbranch, **Error** are used by the Java run-time system([JVM](https://www.geeksforgeeks.org/jvm-works-jvm-architecture/)) to indicate errors having to do with the run-time environment itself(JRE). StackOverflowError is an example of such an error.

Java exception handling is managed via five keywords: **try**, **catch**, **finally ,throw**, **throws**.

Program statements that may raise exceptions are contained within a try block. If an exception occurs within the try block, code can catch this exception (using catch block) and handle it in some rational manner. System-generated exceptions are automatically thrown by the Java run-time system. To manually throw an exception, use the keyword [throw](https://www.geeksforgeeks.org/throw-throws-java/). Any exception that is thrown out of a method must be specified as such by a [throws](https://www.geeksforgeeks.org/throw-throws-java/) clause. Any code that absolutely must be executed after a try block completes is put in a finally block.

The finally keyword is used in association with a [try/catch block](https://www.geeksforgeeks.org/flow-control-in-try-catch-finally-in-java/) and guarantees that a section of code will be executed, even if an exception is thrown. The finally block will be executed after the try and catch blocks.

 finally is meant to execute whether exception occurs or not or whether corresponding catch block found or not.  
use of finally block is **resource deallocation**. Means all the resources such as Network Connections, Database Connections, which we opened in try block are needed to be closed, so that we won’t lose our resources as opened. So those resources are needed to be closed in finally block.

 The finally block is a key tool for preventing **resourceleaks**. When closing a file or otherwise recovering resources, place the code in a finally block to ensure that resource is always recovered.

**finalize method**

The finalize() method is called the finalizer.

Finalizers get invoked when JVM figures out that this particular instance should be garbage collected. Such a finalizer may perform any operations, including bringing the object back to life.

The main purpose of a finalizer is, however, to release resources used by objects before they’re removed from the memory. A finalizer can work as the primary mechanism for clean-up operations, or as a safety net when other methods fail.

In a program, if there is a chance of rising an exception then compiler always warn us about it and compulsorily we should handle that checked exception, Otherwise we will get compile time error saying **unreported exception XXX must be caught or declared to be thrown**. To prevent this compile time error we can handle the exception in two ways:

1. By using [try catch](https://www.geeksforgeeks.org/flow-control-in-try-catch-finally-in-java/)
2. By using **throws** keyword

**Checked Vs Unchecked Exceptions**

**Checked:** are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

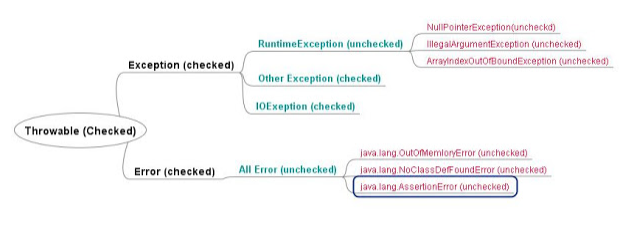
For example, consider the following Java program that opens file at location “C:\test\a.txt” and prints first three lines of it. The program doesn’t compile, because the function main() uses FileReader() and FileReader() throws a checked exception FileNotFoundException. It also uses readLine() and close() methods, and these methods also throw checked exception IOException.

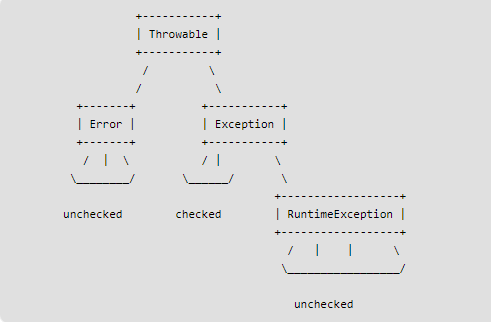
Examples:

FileNotFoundException, IOException, InterruptedException,ParseException

**Unchecked** are the exceptions that are not checked at compiled time. In Java exceptions under Error and RuntimeException classes are unchecked exceptions, everything else under throwable is checked.

Un Checked :NullPointerException, ArrayIndexOutofBoundException





**Checked Exception in Java** is all those Exception which requires being catched and handled during compile time. If Compiler doesn’t see try or catch block handling a Checked Exception, it throws Compilation error. Now Which Exception is checked Exception and Why Checked Exception are introduced in first place? All the Exception which are direct sub Class of Exception but not inherit RuntimeException are Checked Exception.

**Unchecked Exception in Java** is those Exceptions whose handling is not verified during Compile time. Unchecked Exceptions mostly arise due to programming errors like accessing method of a null object, accessing element outside an array bonding or invoking method with illegal arguments. In Java, Unchecked Exception is direct sub Class of RuntimeException.

Custom Exception  
  
Reading from Keyboard:

**Scanner scan = new Scanner (System.in);**

Integer : **int n = scan.nextInt();**

|  |  |
| --- | --- |
| nextInt() | Integer |
| nextFloat() | Float |
| nextDouble() | Double |
| nextLong() | Long |
| nextShort() | Short |
| next() | Single word |
| nextLine() | Line of Strings |
| nextBoolean() |  |

Array:

1. Int numbers[ ] ; / int [ ] numbers;

numbers =new Int[ 20];

1. Int numbers[ ] =new int [20];
2. int[] numbers = new int[]{ 1,2,3,4,5,6,7,8,9,10 }; // Declaring array literal

Hashing is a technique to make things more efficient by effectively narrowing down the search at the outset.

**Java Collections :  (**List, Queue and Set**)**

**Collections in java** is a framework that provides an architecture to store and manipulate the group of objects.  
All the operations that you perform on a data such as searching, sorting, insertion, manipulation, deletion etc. can be performed by Java Collections.

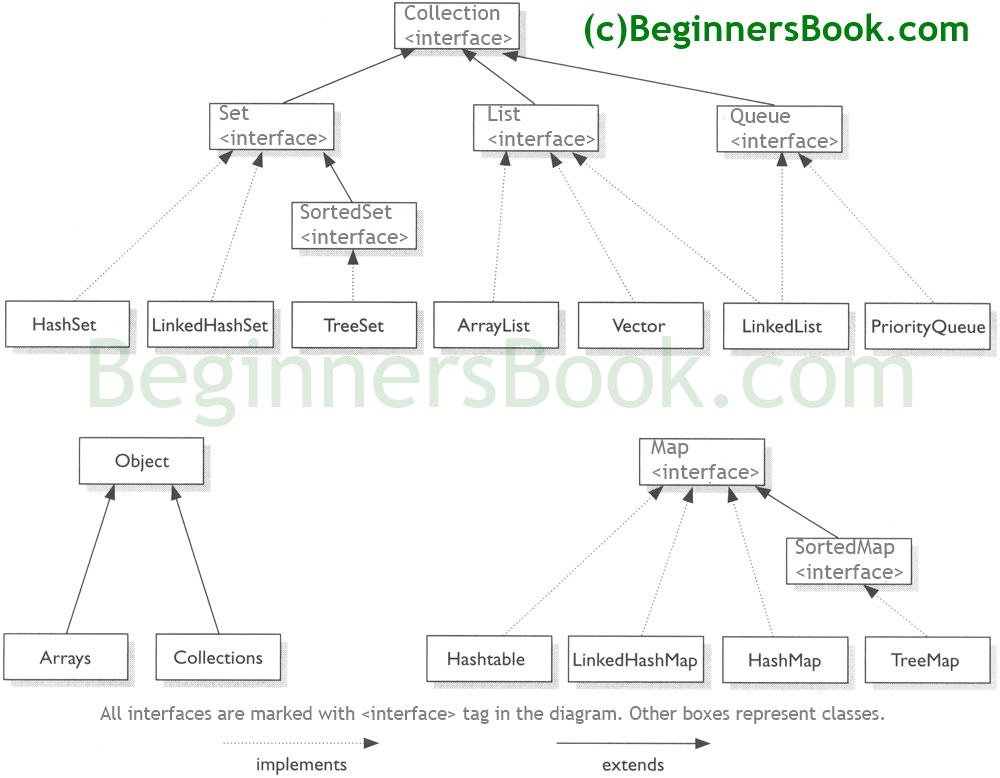
The Collection interface (**java.util.Collection**) and Map interface (**java.util.Map**) are the two main “root” interfaces of Java collection classes. The Map interface is not a subtype of the [Collection interface](https://www.geeksforgeeks.org/collections-in-java-2/). There are three components that extend the collection interface i.e **List, Set and Queue**

All the 3 are interfaces.

**List** : Array List, Linked List, Vectors

**Sets** : HashSets, Linked HashSet, TreeSet

**Queue**: Priority Queue, Blocking Queue



List :

A List is an ordered (maintains insertion order) Collection of elements which may contain duplicates. It is an interface that extents the Collection interface. Lists are further classified into the following:

1. ArrayList
2. LinkedList
3. Vectors

**Array list:** ArrayList is the implementation of List Interface where the elements can be dynamically added or removed from the list. Also, the size of the list is increased dynamically if the elements are added more than the initial size.

List li = new ArrayList();

OR

ArrayList li = new ArrayList ();

List is an Interface (Super type of ArrayList, LinkedList). when you write List li = new ArrayList(); you are creating a subclass object with super class reference, thus you can access the methods of only super class but in case of overridden methods the subclasses method would be invoked.

List l = new ArrayList()  
you will only be able to invoke methods and fields declared inside list interface.  
later you can use the same reference to create a LinkedList like   
l = new LinkedList();   
it allows you to switch between different implementations of the List interface with ease.

ArrayList al = new ArrayList()  
you will be able to invoke methods and fields from both the list and the arraylist.

 Public static void main(String args[]){

 ArrayList al=new ArrayList();  // creating array list

 al.add("Jack");                // adding elements

 al.add("Tyler");

 Iterator itr=al.iterator();

 while(itr.hasNext()){

 System.out.println(itr.next());

 }

 }

// create an array list

ArrayList al = new ArrayList();

System.out.println("Initial size of al: " + al.size());

// add elements to the array list

al.add("C");

al.add("A");

al.add("E");

al.add("B");

al.add("D");

al.add("F");

al.add(1, "A2");

System.out.println("Size of al after additions: " + al.size());

// display the array list

System.out.println("Contents of al: " + al);

// Remove elements from the array list

al.remove("F");

al.remove(2);

System.out.println("Size of al after deletions: " + al.size());

System.out.println("Contents of al: " + al);

}

Output

Initial size of al: 0

Size of al after additions: 7

Contents of al: [C, A2, A, E, B, D, F]

Size of al after deletions: 5

Contents of al: [C, A2, E, B, D]

**Java ArrayList set**() Method example

If there is a need to update the list element based on the index then **set** method of **ArrayList** class can be used. The method **set**(int index, Element E) updates the element of specified index with the given element E.

public static void main(String args[]) {

ArrayList<Integer> arraylist = new ArrayList<Integer>();

arraylist.add(1);

arraylist.add(2);

arraylist.add(3);

arraylist.add(4);

arraylist.add(5);

arraylist.add(6);

arraylist.add(7);

System.out.println("ArrayList before update: "+arraylist);

//Updating 1st element

arraylist.set(0, 11);

//Updating 2nd element

arraylist.set(1, 22);

//Updating 3rd element

arraylist.set(2, 33);

//Updating 4th element

arraylist.set(3, 44);

//Updating 5th element

arraylist.set(4, 55);

System.out.println("ArrayList after Update: "+arraylist);

}

Output

ArrayList before update: [1, 2, 3, 4, 5, 6, 7]

ArrayList after Update: [11, 22, 33, 44, 55, 6, 7]

An ArrayList class can **act as a list** only because it implements List only.

LinkedList class can **act as a list and queue** both because it implements List and Deque interfaces. ArrayList search operation is pretty fast compared to the LinkedList search operation. get(int index) in ArrayList gives the performance of O(1) while LinkedList performance is O(n). ArrayList maintains **index** based system for its elements as it uses array data structure implicitly which makes it faster for searching an element in the list. On the other side LinkedList implements **doubly linked list** which requires the traversal through all the elements for searching an element.  
LinkedList doesn't have an array but a double-ended queue of mutually-connected elements instead. The first element points to the second one, which points to the third one, and so forth. Since this is a doubly-linked list, each element also points to its predecessor. The fifth element, for example, points both to the fourth element and the sixth element.

*An ArrayList is a*resizable array*that grows as additional elements are added. A LinkedList is a doubly-linked list/queue implementation.*

Linked List, In order to store element B, it's not enough to just store its value as you would with an ArrayList.

A pointer to the previous and the next element is also needed in order for the linked list to be traversable. The entire list structure thus consists of mutually connected nodes. Each node contains its element and two pointers: a link to the previous node and the link to the next node. The first node has no previous node and the last node has no next node.

Set:

Unordered collection of objects, in which duplicate values cannot be stored.

Set is NOT an ordered collection, it’s elements does NOT have a particular order. Java Set does **NOT** provide a control over the position where you can insert an element. You cannot access elements by their index and also search elements in the list.

public static void main(String[] args)

    {

        Set<String> hs = new HashSet<String>();

        hs.add("Geeks");

        hs.add("For");

hs.add("Geeks");

        hs.add("Example");

        hs.add("Set");

        System.out.println(hs);

    }

SET VS LIST

List is a type of ordered collection that maintains the elements in **insertion order** while Set is a type of unordered collection so elements are not maintained any order.

List allows duplicates while Set doesn't allow **duplicate elements** . All the elements of a Set should be unique if you try to insert the duplicate element in Set it would replace the existing value.

List permits any number of **null values** in its collection while Set permits only one null value in its collection.

New methods are defined inside **List interface** . But, no new methods are defined inside Set interface, so we have to use Collection interface methods only with **Set subclasses** .

List can be inserted in in both **forward** direction and **backward** direction using Listiterator while Set can be traversed only in forward direction with the help of iterator

By default you can put any Object into a Set, but from Java 5, Java Generics makes it possible to limit the types of object you can insert into a Set

MAP:

A Map is an object that maps keys to values.

A Map doesn't allow duplicate keys, but you can have duplicate values.

HashMap

Linked HashMap

HashTree

TreeMap

The Java Map interface, java.util.Map, represents a mapping between a key and a value. More specifically, a Java Map can store pairs of keys and values. Each key is linked to a specific value. Once stored in a Map, you can later look up the value using just the key.

The Java Map interface is not a subtype of the Collection interface. Therefore it behaves a bit different from the rest of the collection types.

Not a child interface of collection.

* java.util.HashMap
* java.util.Hashtable
* java.util.EnumMap
* java.util.**IdentityHashMap**
* java.util.**LinkedHashMap**
* java.util.Properties
* java.util.**TreeMap**
* java.util.WeakHashMap

The most commonly used Map implementations are HashMap and TreeMap.

HashMap maps a key and a value. It does not guarantee any order of the elements stored internally in the map.

TreeMap also maps a key and a value. Furthermore it guarantees the order in which keys or values are iterated - which is the sort order of the keys or values.  
The HashMap implementation is typically the fastest of the two Map implementations, so whenever you don't need to sort the elements in the Map you can just use a HashMap. Otherwise use a TreeMap.

Map mapA = new HashMap();

Map mapB = new TreeMap();

mapA.put("key1", "element 1");

mapA.put("key2", "element 2");

mapA.put("key3", "element 3");

* Whenever we put a (key,value) pair into the hash map, it evaluates the key object using it's **hasCode()**method to get an index ( hash code) in the array.It then puts the (key,value) pair in the List corresponding to that index.
* Whenever we try to get a value for a given key object, it again evaluates the index (hash code) and go to the List corresponding to that index and search for the key object using the **equals()**method in that list. If it finds the key object, it returns the corresponding value. Otherwise it returns null.
* Creating a generic Map means that you can define the type of the key and the type of the value of object stored in the Map. The declaration and the instantiation of a generic Map is only different to other type of collection such as List and Set is that we to define two types. One type for the key and the other type for the value.
* The syntax for creating a generic Map is as follow:
* Map<String, MyObject> map = new HashMap<String, MyObject>();
* Where K is the type of map key and V is the type of the value stored in the map. If you want a map to hold a value of reference to String object and using an Integer as the key, you will write the declaration and instantiation like the snippet below.
* Map<Integer, String> map = new HashMap<Integer, String>();
* To make it simpler, you can use the diamond operator too.
* Map<Integer, String> map = new HashMap<>();

When you want to add some elements to the map you can use the same put() method. But you don’t have to worry that you put a wrong type of object into the map. Because the Java compiler will check it to see if you are storing a correct type. Generic will catch the bug that should not happen at runtime because the code is already validated at compile time.

**HashMap**

* HashMap has complexity of O(1) for insertion and lookup.
* HashMap allows one null key and multiple null values.
* HashMap does not maintain any order.
* Fastest

**TreeMap**

* TreeMap has complexity of O(logN) for insertion and lookup.
* TreeMap does not allow null key but allow multiple null values.
* TreeMap maintains order. It stores keys in sorted and ascending order.

**LinkedHashMap**

* LinkedHashMap has complexity of O(1) for insertion and lookup.
* LinkedHashMap allows one null key and multiple null values.
* LinkedHashMap maintains order in which key-value pairs are inserted.

Cursors in Java:

If we want to get objects one by one from the collection then we should go for cursor.

There are 3 types of cursors available in java. They are:

1. Enumeration

2. Iterator

3. ListIterator

**Collection :** Root interface with basic methods like add(), remove(),

contains(), isEmpty(), addAll(), ... etc.

[**Set**](https://www.geeksforgeeks.org/set-in-java/) **:** Doesn't allow duplicates. Example implementations of Set

interface are HashSet (Hashing based) and TreeSet (balanced

BST based). Note that TreeSet implements **SortedSet**.

[**List**](https://www.geeksforgeeks.org/list-interface-java-examples/) **:** Can contain duplicates and elements are ordered. Example

implementations are LinkedList (linked list based) and

[ArrayList](https://www.geeksforgeeks.org/array-vs-arraylist-in-java/) (dynamic array based)

[**Queue**](https://www.geeksforgeeks.org/queue-interface-java/) **:** Typically order elements in FIFO order except exceptions

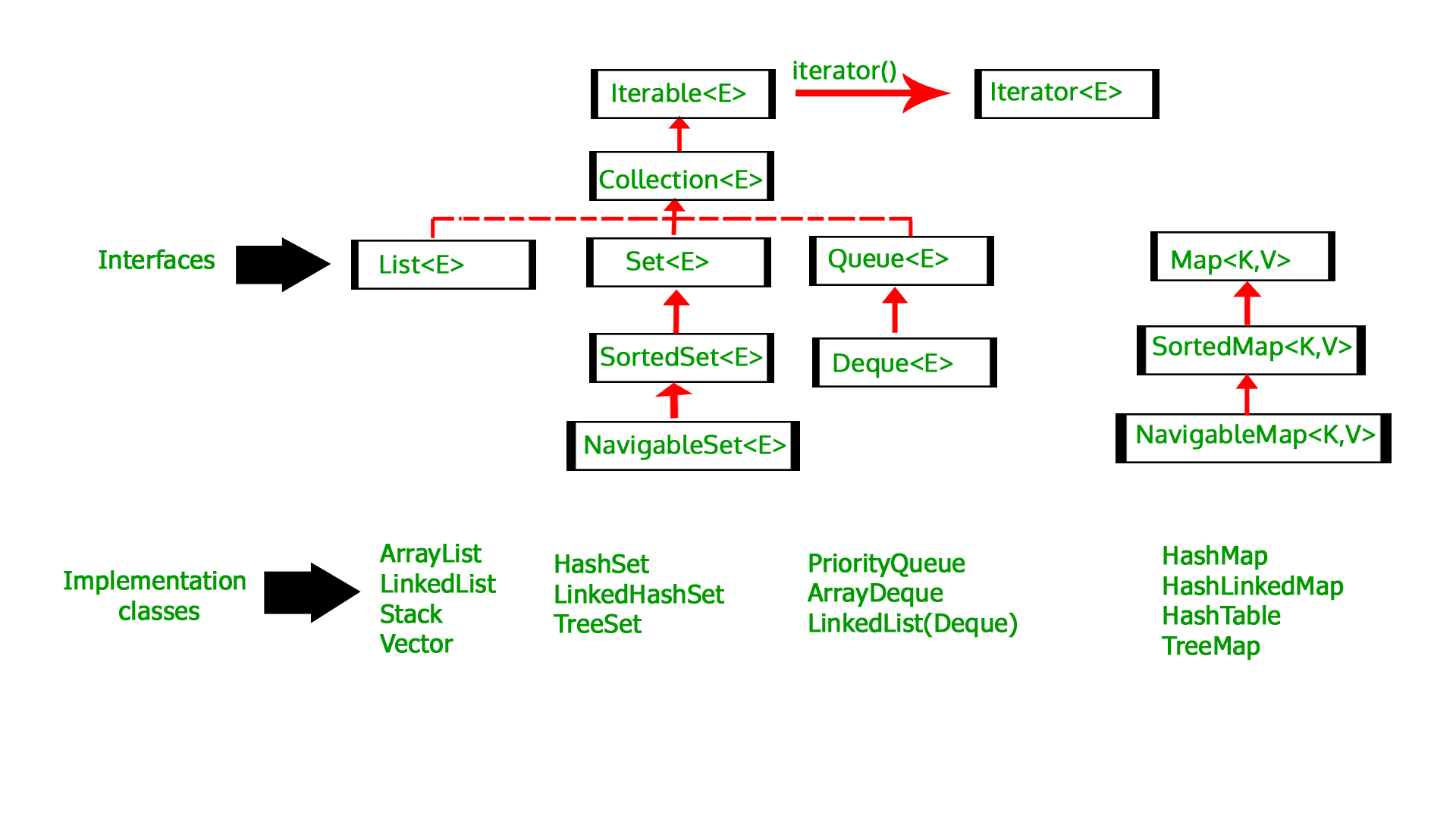
like PriorityQueue.

[**Deque**](https://www.geeksforgeeks.org/deque-interface-java-example/) **:** Elements can be inserted and removed at both ends. Allows

both LIFO and FIFO.

The difference between Set and Map interface is that in Set we

have only keys, whereas in Map, we have key, value pairs.



**Iterator interface** : Iterator is an interface that iterates the elements. It is used to traverse the list and modify the elements. Iterator interface has three methods which are mentioned below:

1. **public Boolean hasNext()** – This method returns true if iterator has more elements.
2. **public object next()** – It returns the element and moves the cursor pointer to the next element.
3. **public void remove()** – This method removes the last elements returned by the iterator.

Iterator used along with List:

ArrayList<String> list = new ArrayList<String>();

        list.add("A");

        list.add("B");

        list.add("C");

        list.add("D");

        list.add("E");

        // Iterator to traverse the list

        Iterator iterator = list.iterator();

        System.out.println("List elements : ");

        while (iterator.**hasNext**())

            System.out.print(iterator.**next**() + " ");

        System.out.println();

    }

Singleton Class

The Singleton's purpose is to control object creation, limiting the number of objects to only one. Since there is only one Singleton instance, any instance fields of a Singleton will occur only once per class, just like static fields. Singletons often control access to resources, such as database connections or sockets.

A singleton class is a class that can have only one object (an instance of the class) at a time.

To design a singleton class:

1. Make constructor as private.
2. Write a static method that has return type object of this singleton class.

class Dealer

{

    // static variable single\_instance of type Singleton

    private static Dealer single\_instance = null;

    // variable of type String

    public String s;

    // private constructor restricted to this class itself

    private Dealer()

    {

        s = "Hello I am a string part of Singleton class";

    }

    // static method to create instance of Singleton class

    public static Dealer getInstance()

    {

        if (single\_instance == null)

            single\_instance = new Singleton();

        return single\_instance;

    }

}

Difference in normal and singleton class in terms of instantiation is that, For normal class we use constructor, whereas for singleton class we use getInstance() method

You can notice in the above code that getInstance method ensures that only one instance of the class is created. The constructor should not be accessible from the outside of the class to ensure the only way of instantiating the class would be only through the getInstance method.

The getInstance method is used also to provide a global point of access to the object and it can be used in the following manner:

**Singleton**.getInstance().doSomething();

## Applicability & Examples

According to the definition the singleton pattern should be used when there must be exactly one instance of a class, and when it must be accessible to clients from a global access point. Here are some real situations where the singleton is used:

Singletons can be used while working with databases. They can be used to create a connection pool to access the database while reusing the same connection for all the clients. For example,

### Example 1 - Logger Classes

The Singleton pattern is used in the design of logger classes. This classes are ussualy implemented as a singletons, and provides a global logging access point in all the application components without being necessary to create an object each time a logging operations is performed.

### Example 2 - Configuration Classes

The Singleton pattern is used to design the classes which provides the configuration settings for an application. By implementing configuration classes as Singleton not only that we provide a global access point, but we also keep the instance we use as a cache object. When the class is instantiated( or when a value is read ) the singleton will keep the values in its internal structure. If the values are read from the database or from files this avoids the reloading the values each time the configuration parameters are used.

### Example 3 - Accesing resources in shared mode

It can be used in the design of an application that needs to work with the serial port. Let's say that there are many classes in the application, working in an multi-threading environment, which needs to operate actions on the serial port. In this case a singleton with synchronized methods could be used to be used to manage all the operations on the serial port.

### Example 4 - Factories implemented as Singletons

Let's assume that we design an application with a factory to generate new objects(Acount, Customer, Site, Address objects) with their ids, in an multithreading environment. If the factory is instantiated twice in 2 different threads then is possible to have 2 overlapping ids for 2 different objects. If we implement the Factory as a singleton we avoid this problem. Combining Abstract Factory or Factory Method and Singleton design patterns is a common practice.

## Specific problems and implementation

### Thread-safe implementation for multi-threading use.

A robust singleton implementation should work in any conditions. This is why we need to ensure it works when multiple threads uses it. As seen in the previous examples singletons can be used specifically in multi-threaded application to make sure the reads/writes are synchronized.

Upcasting:

Casting a subtype to a super type.

Java Generics:

The **Java Generics** programming is introduced in J2SE 5 to deal with **type-safe objects.**  
Before generics, we can store any type of objects in collection i.e. non-generic. Now generics, forces the java programmer to store specific type of objects.

***The goal of implementing Generics is finding bugs in compile-time, other than in run-time****.* Finding bugs in compile-time can save time for debugging java program, because compile-time bugs are much easier to find and fix. Generic types only exist in compile-time. This fact is the most important thing to remember for learning Java Generics.

Using Java Generics, a class can be parameterized with a type argument. For instance, consider class A, in the example below.

Class Employee{

...

}

You can make it a generic type by declaring it as A (pronounced A of T). Now, you can use the type variable T, in the body of class Employee, as if it were a proper type.

Class Employee<T> {

T myFieldOfTypeT;

...

}

To instantiate Employee, you need to provide the arguments for the type parameters, which is the actual type you want to use:

Employee<Long> e = new Employee<Long>();

Now e is an instance of the Employee class, which is a proper parameterized generic class. Here we need to clarify a few definitions:

* A **generic type** is a class that is parameterized with type arguments, for example, ArrayList<T>.
* A **parameterized type** is a class where the type parameter is instantiated with a fixed argument, for example, ArrayList<Long>.
* A **raw type** is a generic type that is not parameterized with anything, like new ArrayList().

You will often find a ‘?’ Symbol inside a generic parameter. It is a wildcard and stands for an arbitrary type. For example, an ArrayList<?> is an array list of any type. It means that it can represent an ArrayList of Strings, or an ArrayList of Integers, or whatever type. However, at the runtime, it will have some fixed type.

**1) Type-safety :** We can hold only a single type of objects in generics. It doesn’t allow to store other objects.

**2) Type casting is not required:** There is no need to typecast the object.

Before Generics, we need to type cast.

List list = **new** ArrayList();

list.add("hello");

String s = (String) list.get(0);//typecasting

After Generics, we don't need to typecast the object.

List<String> list = **new** ArrayList<String>();

list.add("hello");

String s = list.get(0);

By adding the diamond operator <> / angle brackets containing the type, we narrow the specialization of this list only to Integer type i.e. we specify the type that will be held inside the list. The compiler can enforce the type at compile time.

Generics helps us to create a single class, which can be useful to operate on multiple data types. A class, interface or a method that operates on a parameterized type is called generics class, interface or method. Generics adds type safty. Remember that generics only works on objects, not primitive types.

Generic in Java is added to provide compile time type-safety of code and removing risk of ClassCastException at [runtime](http://javarevisited.blogspot.sg/2012/03/what-is-static-and-dynamic-binding-in.html) which was quite frequent error in Java code, for those who doesn’t know what is type-safety at compile time, it’s just a check by compiler that correct Type is used in correct place and there should not be any ClassCastException.  
  
Java Generic type is only a compile-time concept. During run-time, all types information is erased, and this is call Type Erasure. [Here](https://www.programcreek.com/2011/12/java-type-erasure/) is an interesting example to show how to avoid the common mistakes of Type Erasure.

### Java static block

Java static block is the group of statements that gets executed when the class is loaded into memory by [Java ClassLoader](https://www.journaldev.com/349/java-classloader).

### Can we execute a program without main() method?

Ans) No, one of the ways was the static block, but it was possible till JDK 1.6. Since JDK 1.7, it is not possible to execute a java class without the main method.

**class** A3{

**static**{

  System.out.println("static block is invoked");

  System.exit(0);

  }

}

Algorithms:

Quick Sort:

The basic idea behind implementing quick sort is to divide the array in small parts and then sort the individual parts. The quick sort algorithm uses partitioning as its major routine. The strategy used is Divide and Conquer.

The steps are: 1) Pick an element from the array, this element is called as pivot element. 2) Divide the unsorted array of elements in two arrays with values less than the pivot come in the first sub array, while all elements with values greater than the pivot come in the second sub-array (equal values can go either way). This step is called the partition operation. 3) Recursively repeat the step 2(until the sub-arrays are sorted) to the sub-array of elements with smaller values and separately to the sub-array of elements with greater values

http://www.algolist.net/Algorithms/Sorting/Quicksort

Bubble Sort:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order.



The **bubble sort** makes multiple passes through a list. It compares adjacent items and exchanges those that are out of order. Each pass through the list places the next largest value in its proper place. In essence, each item “bubbles” up to the location where it belongs.

[Figure 1](http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBubbleSort.html#fig-bubblepass) shows the first pass of a bubble sort. The shaded items are being compared to see if they are out of order. If there are n items in the list, then there are n−1n−1 pairs of items that need to be compared on the first pass. It is important to note that once the largest value in the list is part of a pair, it will continually be moved along until the pass is complete.

At the start of the second pass, the largest value is now in place. There are n−1n−1 items left to sort, meaning that there will be n−2n−2 pairs. Since each pass places the next largest value in place, the total number of passes necessary will be n−1n−1. After completing the n−1n−1 passes, the smallest item must be in the correct position with no further processing required. [ActiveCode 1](http://interactivepython.org/runestone/static/pythonds/SortSearch/TheBubbleSort.html#lst-bubble) shows the complete bubbleSortfunction. It takes the list as a parameter, and modifies it by exchanging items as necessary.

Merge Sort

Linked List:

Insertion:

SELENIUM:

Version : 3.141.59

What is Selenium 1 and Selenium 2?

Selenium RC =1

Selenium RC + Web Driver =2

### WebDriver driver = new WebDriver();

**Wrong**- Webdriver is an Interface. We cannot instantiate interface.

### FirefoxDriver driver = new FirefoxDriver();?

**Correct**:

Yes, it is perfectly correct. FirefoxDriver is an implementing class of WebDriver interface and the above statement will launch the Firefox browser.

WebDriver is an interface and the implementing classes are Firefoxdriver, chromedriver etc.

why do we need to create reference variable of type WebDriver?

WebDriver driver = new FirefoxDriver();?

Having a reference variable of type WebDriver allows us to assign the driver object to different browser specific drivers. Thus allowing multi-browser testing by assigning the driver object to any of the desired browser.

Architecture:

**Selenium webdriver architecture mainly divided into three parts**

1. Language level bindings
2. Selenium Webdriver API
3. Drivers
4. **Language binding:** To support multiple languages, selenium has developed language bindings. If you want to use the browser driver in Java, use the Java bindings for Selenium Webdriver. If you want to use the browser driver in C#, Ruby or Python, use the binding for that language. All language binding can be downloaded from selenium official [website](http://www.seleniumhq.org/download/).
5. **Selenium Webdriver API:** It is an API which makes possible to communication between programming languages and browsers. It follows object oriented concepts. It has multiple classes and interfaces.
6. **Browser drivers:**  A browser drivers helps in communication with browser without revealing the internal logic of browser’s functionality. The browser driver is the same regardless of the language used for automation.

When the automation script is executed, the following steps are done internally:

1. A HTTP request is created and sent to browser driver for each selenium instruction or commands.
2. A browser driver receives the HTTP request through HTTP server.
3. HTTP server decides all steps to perform instructions which are executed on browser.
4. Execution status is sent back to HTTP server which is sent back to automation script.

Let us now understand the Selenium Web Driver Architecture. Selenium WebDriver API enables interaction between browsers and browser drivers. This architecture consists of four layers namely the Selenium Client Library, JSON Wire Protocol, Browser Drivers and Browsers.

* Selenium Client Library consists of languages like Java, Ruby, Python, C# and so on. After the test cases are triggered, entire Selenium code will be converted to Json format.
* JSON stands for Javascript Object Notation. It takes up the task of transferring information from the server to the client. JSON Wire Protocol is primarily responsible for transfer of data between HTTP servers. Generated Json is made available to browser drivers through http Protocol.
* Each browser has a specific browser driver. Browser drivers interact with its respective browsers and execute the commands by interpreting Json which they received from the browser. As soon as the browser driver gets any instructions, they run them on the browser. Then the response is given back in the form of HTTP response.

Let’s consider the following block of code −

WebDriver driver = new ChromeDriver();

driver.get ([“https://www.tutorialspoint.com/index.htm“](https://www.tutorialspoint.com/index.htm));

Once we run this block of code, the entire code will be converted with the help of JSON Wire Protocol over HTTP as a URL. The converted URL will be fed to the ChromeDriver.

The browser driver utilizes HTTP server to get the request from HTTP. As the browser driver gets the URL, it passes the request to its browser via HTTP. It will trigger the event of executing the Selenium instructions on the browser.

Now if the request is that of POST, it will trigger an action on the browser. If it’s a GET request, then the response will be produced at the browser end. Finally it will be passed over HTTP to the browser driver. The browser driver will in turn send it to the UI via JSON Wire Protocol.

This sums up the overall explanation of the Selenium WebDriver

Locators in Selenium:

Id, Name, Xpath, CSS, DOM, Link Text & partial Link Text, Tag Name, Class Name

**XPATH**:

**Single Attribute:**

The syntax for locating elements through XPath- Single Attribute can be written as:

//<HTML tag>[@attribute\_name='attribute\_value']

//input[@id=’user’]

**Multiple Attribute:**

The syntax for locating elements through XPath- Multiple Attribute can be written as

//HTML tag[@attribute\_name1='attribute\_value1'][@attribute\_name2='attribute\_value2]

# And

//<HTML tag>[@attribute\_name1='attribute\_value1' and @attribute\_name2='attribute\_value2]

OR:

//<HTML tag>[@attribute\_name1='attribute\_value1' or @attribute\_name2='attribute\_value2]

# Contains: //<HTML tag>[contains(@attribute\_name,'attribute\_value')

# starts-with()

//<HTML tag>[starts-with(@attribute\_name,'attribute\_value')]

Text()

//\*[text()='Google offered in']

Last()

last() method" selects the last element (of mentioned type) out of all input element present.

(//input[@type='text'])[last()]

Pipe operator:

//input[@id='email']|//input[@name='email']

XPATH AXES:

Eg: https://money.rediff.com/gainers/bse/daily/groupa?src=gain\_lose

* 1. Ancestor

//a[contains(text(),'Reliance Nav')]/ancestor::tr/td[2]

Get the ancestor tag and selecting sibling

* 1. Ancestor or self
  2. Attribute
  3. Child

//div[@class='SDkEP']/child::\* google

Return all child nodes

* 1. Descendant

Returns child as well as grand children

* 1. Descendant or self
  2. Following
     + Returns all nodes after the current node.
  3. Following sibling
     + Returns only the sibling of the current node.
     + //div[@class='RjPuVb']/following-sibling::\* (google)
  4. Preceeding sibling
     + //td[contains(text(),'F000ZA')]/preceeding-sibling::td[4]
  5. Namespace
  6. Parent
  7. Self

Ancestor, following, following-sibling

**CSS**:

When we don't have an option to choose Id or Name, we should prefer using CSS locators as the best alternative.  
CSS is "Cascading Style Sheets" and it is defined to display HTML in structured and colorful styles are applied to webpage.

Selectors are patterns that match against elements in a tree, and as such form one of several technologies that can be used to select nodes in an XML document.

* CSS has more Advantage than Xpath
* CSS is much faster and simpler than the Xpath.

1. **ID -: #idValue**  
   <button id="submitButton1" type="button" class="btn">Submit</button>  
    Answer: # submitButton1
2. **Class :.classvalue**
3. <button id="submitButton1" type="button" class="btn">Submit</button>  
   Answer : .btn
4. Tag: tagname  
   <input id="fname" type="text" name="firstName" class="textbox">  
   Answer: input
5. **Attribute & Value:[attribute=’value’]**  
   <input id="fname" type="text" name="firstName" class="textbox">  
   Answer: [name='firstName']
6. **Tag and ID :  tag#id**  
   <input id="fname" type="text" name="firstName" class="textbox">  
    Answer :input#fname
7. **Tag and class : tag.class**  
   <input id="fname" type="text" name="firstName" class="textbox">  
   Answer: input.textbox
8. **Tag and attribute :tag[attributeName='attributeValue']**  
   <input id="fname" type="text" name="firstName" class="textbox">  
   Answer :  input[name='firstName']

 Tag, class, and attribute : css=tag.classvalue[attribute=value]

DOM : document.getElementById("id of the element")

Entering Values in Input Boxes: **sendKeys()**

Deleting Values in Input Boxes :**clear()**

Radio Button :  
**rdoButton.click()  OR rdoButton.sendKeys(Keys.SPACE)**

Check Box :  
chkBox.click() OR chkBox.sendKeys(Keys.SPACE)

Links: lnk.click()

Submitting a Form

The **submit()**method is used to submit a form.  
This is an alternative to clicking the form's submit button.

txtUserid.sendKeys(userID);

txtPassword.sendKeys(password);

txtPassword.submit();  
//btnLogin.click();

drop down:

@FindBy(id="primaryApplicant\_ResidenceType")

WebElement drpAppResType;

Select osel=new Select(drpAppResType);  
osel.selectByIndex(1);

odrp.**selectByVisibleText()**

**odrp.selectByValue();**

Selecting Items in a Multiple SELECT elements

Same like dropdown, but write multiple odrp.SelectBy...

//span[contains(text(), 'RouteOne App')]

**Printing the Options on a Drop Down.**

@FindBy(id="primaryApplicant\_ResidenceType")   
WebElement drpAppResType;

Select odrp=**new** Select(drpAppResType);

List<WebElement> drpEle= odrp.getOptions();

**for**(**int** i=0;i<drpEle.size();i++)

{

System.***out***.println(drpEle.get(i).getText());

}

Getting all Hyperlinks on a webpage:  
List<WebElement> allLinks = driver.findElements(By.tagName("a"));

@FindBy(tagname="a")

List<WebElement> allLinks;

for (int i = 0; i<= allLinks.size(); i++){

System.out.println(allLinks.get(i).getText());

}

Keyboard & Mouse Event using Action Class in Selenium Webdriver

Import the **Actions** and **Action** classes.

Import org.openqa.selenium.interactions.Actions

Import org.openqa.selenium.interactions.Action

@FindBy(xpath="//a[@data-tab-name='tab.newBusinessApp']")   
WebElement lnkBApp;

Actions builder =new Actions(driver);

builder.moveToElement(lnkBApp).perform();

lnkBApp.click();

In above case we simply performed one action on the element, incase we want to perform sequence of events on the same element, we can achieve same by using build method.

Pressing SHIFT

 Actions builder = **new** Actions(driver);

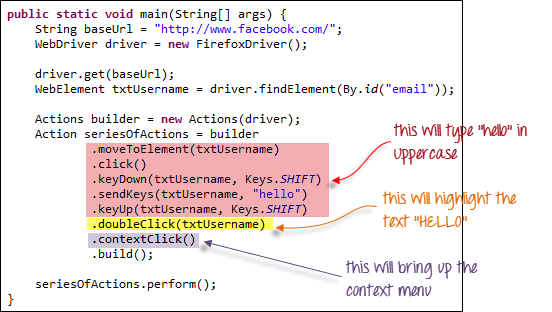
Action typeInCAPS = builder.keyDown(txtAppLname, Keys.***SHIFT***)

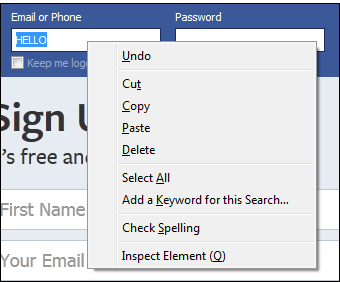
.sendKeys(txtAppLname, "amjed")

.keyUp(txtAppLname, Keys.***SHIFT***)

.build();

typeInCAPS.perform();





**clickAndHold()  
contextClick()  
doubleClick()  
dragAndDrop(source, target)  
dragAndDropBy(source, x-offset, y-offset)  
keyDown(modifier\_key)  
keyUp(modifier \_key)  
moveByOffset(x-offset, y-offset)  
moveToElement(toElement)  
release()  
sendKeys(onElement, charsequence)**

**Drag and Drop:**

//Element which needs to drag.

WebElementFrom=driver.findElement(By.xpath("//\*[@id='credit2']/a"));

//Element on which need to drop.

WebElement To=driver.findElement(By.xpath("//\*[@id='bank']/li"));

//Using Action class for drag and drop.

Actions act=new Actions(driver);

//Dragged and dropped.

act.dragAndDrop(From, To).build().perform();

}

CHILD WINDOW

// Load app

driver.get("http://www.naukri.com/");

String parent=driver.getWindowHandle();

// It will return the parent window name as a String

Set<String>child\_windows=driver.getWindowHandles();  
// This will return the number of windows opened by Webdriver and will return Set of Strings

Iterator<String> I1= child\_windows.iterator();

// Now we will iterate using Iterator

while(I1.hasNext())

{

   String child =I1.next();

// Here we will compare if parent window is not equal to child window then we  will close

if(!parent.equals(child))

{

driver.switchTo().window(child);

System.out.println(driver.switchTo().window(child\_window).getTitle());

driver.close();

}

}

// once all pop up closed now switch to parent window

driver.switchTo().window(parent);

}

}

**Finding Broken Images:**

To achieve this, we can use HTTPClient library to check status codes of the images on a page. If they don't load correctly, then it will be registered with likely a 404 but not a 200 status code

First we will try to find all images on the page by using Webdriver.  
List<WebElement>imagesList = driver.findElements(By.tagName("img"));

Now iterate through each image and verify response code with HttpStatus and it should be 200 if not, increment invalid images count. We can get the response code using below statement:

response.getStatusLine().getStatusCode()

Framework Explanantion

In our Selenium Project we are using Java language.we are using Test-driven Framework by using Page Object Model design pattern with Page Factory.

As per the Page Object Model, we have maintained a class for every web page. Each web page has a separate class and that class holds the functionality and members of that web page.

1. Finding elements using @FindBy

Using the PageFactory, these WebElements are initialized when a Page Object is created

1. Constructor
2. Methods

We have separate packages for Pages and Tests. All the web page related classes come under **Pages** package and all the tests related classes come under **Tests** package.

Three packages: page objects, Tests and Utils

**Utility Class (AKA Functions Class):**Utility class (TestUtil.java) stores and handles the functions (The code which is repetitive in nature such as waits, actions, capturing screenshots, accessing excels, sending email etc.,) which can be commonly used across the entire framework. The reason behind creating utility class is to achieve reusability.

**Properties file:**This file (config.properties) stores the information that remains static throughout the framework such as browser specific information, application URL, screenshots path etc.

All the details which change as per the environment and authorization such as Browser,URL, Login Credentials are kept in the input.properties file. Keeping these details in a separate file makes easy to maintain.

**Screenshots:** Screenshots will be captured and stored in a separate folder and also the screenshots of a failed test cases will be added in the extent reports.

**Test Data:** All the historical test data will be kept in excel sheet (controller.xlsx). By using ‘controller.xlsx’, we pass test data and handle data driven testing. We use [Apache POI](https://www.softwaretestingmaterial.com/handling-excel-files-using-apache-poi/) to handle excel sheets.

**TestNG:** Using TestNG for Assertions, Grouping and Parallel execution.

**Maven:** Using Maven for build, execution and dependency purpose. Integrating the TestNG dependency in POM.xml file and running this POM.xml file using Jenkins.

**Version Control Tool:** We use Git as a repository to store our test scripts.

**Jenkins:** By using Jenkins CI Tool, we execute test cases on daily basis and also for nightly execution based on the schedule. Test Result will be sent to the peers using Jenkins.

**Extent Reports:** For the reporting purpose, we are using Extent Reports. It generates beautiful HTML reports. We use the extent reports for maintaining logs and also to include the screenshots of failed test cases in the Extent Report.

Types of web driver APIs in Selenium (Implementing classes for WeDriver Interface)

AndroidDriver,

ChromeDriver,

EventFiringWebDriver,

FirefoxDriver,

HtmlUnitDriver,

InternetExplorerDriver,

IPhoneDriver,

IPhoneSimulatorDriver,

RemoteWebDriver.

The fastest implementation of WebDriver is the HTMLUnitDriver. It is because the HTMLUnitDriver does not execute tests in the browser.

What are the Programming Languages supported by Selenium WebDiver?

•   Java

•   C#

•   Python

•   Ruby

•   Perl

•   PHP

23. What are the Operating Systems supported by Selenium WebDriver?

•   Windows

•   Linux

•   Mac

24. What are the Open-source Frameworks supported by Selenium WebDriver?

•   JUnit

•   TestNG

•   CUCUMBER

•   JBHEAVE

What are the different exceptions you have faced in Selenium WebDriver?

1. **ElementNotVisibleException**: Although an element is present in the DOM, it is not visible (cannot be interacted with). E.g. Hidden Elements – defined in HTML using type=”hidden”.
2. **ElementNotSelectableException**: Although an element is present in the DOM, it may be disabled (cannot be clicked/selected).
3. **InvalidSelectorException**: Selector used to find an element does not return a WebElement. Say XPath expression is used which is either syntactically invalid or does not select WebElement.
4. **NoSuchElementException**: WebDriver is unable to identify the elements during run time, i.e. FindBy method can’t find the element.
5. **NoSuchFrameException**: WebDriver is switching to an invalid frame, which is not available.
6. **NoAlertPresentException**: WebDriver is switching to an invalid alert, which is not available.
7. **NoSuchWindowException**: WebDriver is switching to an invalid window, which is not available.
8. **StaleElementReferenceException**: The referenced element is no longer present on the DOM page (reference to an element is now Stale). E.g. The Element belongs to a different frame than the current one OR the user has navigated away to another page.
9. **SessionNotFoundException**: The WebDriver is performing the action immediately after ‘quitting’ the browser.
10. **TimeoutException**: The command did not complete in enough time. E.g. the element didn’t display in the specified time. Encountered when working with waits.
11. **WebDriverException**: The WebDriver is performing the action immediately after ‘closing’ the browser.

NoSuchWindow,NoSuchFrame,NoSuchElement,ElementNotVisible,Timeout,WebDriverException

How To Login Into Any Site If It Is Showing Any Authentication Pop-Up For Username And Password?

To do this we pass username and password with the URL

<http://username:password@url>

How to input text in the text box without calling the sendKeys()?

// To initialize js object

JavascriptExecutor js = (JavascriptExecutor)driver;

// To enter email id

js.executeScript("document.getElementById(‘Email’).value=amjed4u@gmail.com");

How to get a text of a web element?

By using getText() method

47. How to get an attribute value using Selenium WebDriver?

input attr1='a' attr2='b' attr3='c'>foo</input>

By using getAttribute(attr1); ans: a

How to submit a form using Selenium WebDriver?

We use “submit” method on element to submit a form

driver.findElement(By.id("form\_1")).submit();

Alternatively, you can use click method on the element which does form submission

 How to press ENTER key on text box In Selenium WebDriver?

To press ENTER key using Selenium WebDriver, We need to use Selenium Enum Keys with its constant ENTER.

driver.findElement(By.xpath("xpath")).sendKeys(Keys.ENTER);

How to pause a test execution for 5 seconds at a specific point?

By using java.lang.Thread.sleep(long milliseconds) method we could pause the execution for a specific time. To pause 5 seconds, we need to pass parameter as 5000 (5 seconds)

Thread.sleep(5000)

What is the alternative to driver.get() method to open an URL using Selenium WebDriver?

driver.navigate.to(“url”)

 What is the difference between driver.get() and driver.navigate.to(“url”)?

driver.get(): To open an URL and it will wait till the whole page gets loaded

driver.navigate.get(): To navigate to an URL and It will not wait till the whole page gets loaded

Can I navigate back and forth in a browser in Selenium WebDriver?

We use Navigate interface to do navigate back and forth in a browser. It has methods to move back, forward as well as to refresh a page.

driver.navigate().forward(); – to navigate to the next web page with reference to the browser’s history

driver.navigate().back(); – takes back to the previous webpage with reference to the browser’s history

driver.navigate().refresh(); – to refresh the current web page thereby reloading all the web elements

driver.navigate().to(“url”); – to launch a new web browser window and navigate to the specified URL

 How to fetch the current page URL in Selenium?

To fetch the current page URL, we use getCurrentURL()

driver.getCurrentUrl();

To maximize browser window in selenium we use maximize() method. This method maximizes the current window if it is not already maximized

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

How to delete cookies in Selenium?

To delete cookies we use deleteAllCookies() method

driver.manage().deleteAllCookies();

61. What are the ways to refresh a browser using Selenium WebDriver?

There are multiple ways to refresh a page in selenium

•   Using driver.navigate().refresh() command as mentioned in the question 45

•   Using driver.get(“URL”) on the current URL or using driver.getCurrentUrl()

•   Using driver.navigate().to(“URL”) on the current URL or driver.navigate().to(driver.getCurrentUrl());

•   Using sendKeys(Keys.F5) on any textbox on the webpage

What is the difference between driver.getWindowHandle() and driver.getWindowHandles() in Selenium WebDriver?

driver.getWindowHandle() – It returns a handle of the current page (a unique identifier)

driver.getWindowHandles() – It returns a set of handles of the all the pages available.

What is the difference between driver.findElement() and driver.findElements() commands?

The difference between driver.findElement() and driver.findElements() commands is-

•   findElement() returns a single WebElement (found first) based on the locator passed as parameter. Whereas findElements() returns a list of WebElements, all satisfying the locator value passed.

•   Syntax of findElement()-

•   WebElement textbox = driver.findElement(By.id(“textBoxLocator”));

•   Syntax of findElements()-

•   List <WebElement> elements = driver.findElements(By.id(“value”));

•   Another difference between the two is- if no element is found then findElement() throws NoSuchElementException whereas findElements() returns a list of 0 elements.

List<WebElement> list = driver.findElements(By.tagName(“a”));

Sop(list.size()); ==40

How to find whether an element is displayed on the web page?

WebDriver facilitates the user with the following methods to check the visibility of the web elements.  
These web elements can be buttons, drop boxes, checkboxes, radio buttons, labels etc.

 isDisplayed()

 boolean elePresent = driver.findElement(By.xpath("xpath")).isDisplayed();

  isSelected()

 boolean eleSelected= driver.findElement(By.xpath("xpath")).isSelected();

  isEnabled()

   boolean eleEnabled= driver.findElement(By.xpath("xpath")).isEnabled();

How to select a value in a dropdown?

By using Select class

WebElement mySelectElement = driver.findElement(By.name("dropdown"));

Select dropdown = new Select(mySelectElement);

dropdown.selectByVisibleText(Text);

dropdown.selectByIndex(Index);

dropdown.selectByValue(Value);

How to mouse hover on a web element using WebDriver?

By using Actions class

WebElement ele = driver.findElement(By.xpath("xpath"));

Actions action = new Actions(driver);

action.moveToElement(ele).build().perform();

 How to handle hidden elements in Selenium WebDriver?

It is one of the most important selenium interview questions.

We can handle hidden elements by using javaScript executor

(JavascriptExecutor(driver)).executeScript("document.getElementsByClassName(ElementLocator).click();");

**What is aXpath?**

Xpath is a XML path language, which describes a way to traverse through XML document to locate nodes.   
Same is used to locate elements in HTML.

**How handle pop up in web application using Selenium Webdriver?**

Alert alert = driver.switchTo().alert();

To cancel the message :

alert.dismiss();

To accept the message :

alert.accept();

To get string from alert message :

String msg = alert.getText();

To send string to alert box :

alert.sendKeys(“Hi”);

**Name the fastest web driver**

WebDriver driver = new HtmlUnitDriver(); //Headless browser

**TestNG**

**List annotations in TestNG**

**@BeforeSuite:** The annotated method will be run only once before all tests in this suite have run.

**@**AfterSuite : The annotated method will be run only once after all tests in this suite have run.

**@BeforeTest:** Method will run before any test method related to the classes which are inside the <test> tag as per the <testng.xml> file.  
**@AfterTest:** AfterTest method runs after any test method inside the <test> tag is run. (in testNG.xml file). It runs after the execution of classes takes place..  
**@BeforeGroups:** The list of groups that this configuration method will run before.

**@AfterGroups:** The list of groups that this configuration method will run after.

**@BeforeClass:** The method will be run before the first test method in the current class is invoked.

**@AfterClass:** The method will be run after all the test methods in the current class have been run.

**@BeforeMethod:**The method will be run before each test method.

**@AfterMethod:** The method will be run after each test method*.*

**@Test:** The method is a part of a test case.

|  |  |
| --- | --- |
| **@Factory** | You use it to execute any specific group of test cases with different values. It returns an array of test class objects as the <Object[ ]>. |
| **@Listeners** | You can use them with the test classes for the logging function. |
| **@Parameters** | You can use this annotation for passing the parameters to the test methods. |
| **@DataProvider** | It marks a method as a data source for the test. Every @DataProvider annotated method must always return the value as <Object[ ][ ]>. You can use it in any of the @Test annotated methods. |

Test(enabled=true,priority=2,dataProvider="dealerships",dataProviderClass=DataProviderClass.class)

in beforeSuite

in beforeTest

in beforeClass

in beforeMethod

in test case 1

in afterMethod

in beforeMethod

in test case 2

in afterMethod

in afterClass

in afterTest

in afterSuite

* First of all, beforeSuite() method is executed only once. Lastly, the afterSuite() method executes only once.
* Even the methods beforeTest(), beforeClass(), afterClass(), and afterTest() methods are executed only once.
* beforeMethod() method executes for each test case but before executing the test case.
* afterMethod() method executes for each test case but after executing the test case.

If you are not using any priority in your test method then TestNG assign by default priority=0 to the Test Method.

If there is same priority assign to test methods then execution order will be alphabetical.

No priority or 0 will execute first based on alphabetical order.

Sample TestNG XML

<suite name=*"Suite"*>

<test name=*"ROne"*>

<classes>

<class name=*"net.routeone.tests.DealerUserTest"* />

</classes>

</test>

Parallel Execution

<suite name="TestSuite" thread-count="3" parallel="methods" >

<test name="testGuru">

<classes>

<class name=" *net.routeone.tests.DealerUserTest*">

</class>

</classes>

</test>

</suite>

Executing multiple times:

@Test(threadPoolSize = 3, invocationCount = 6, timeOut = 1000)

     public void testMethod()

     {

}

Excluding in TestNG.XML

<suite name="Suite">

  <test name="SmokeTest">

    <classes>

        <class name="com.test.firstpackage.FirstClassInFirstPackage">

            <methods>

            <include name="firstTestCase"></include>

            <include name="secondTestCase"></include>

            <exclude name="thirdTestCase"></exclude>

            </methods>

        </class>

    </classes>

  </test>

</suite>

**Depends on Methods:**

public class DependentTestExamples

{

     @Test(dependsOnMethods = { "C", "B" })

     public void A() {

         System.out.println("Test method A");

     }

     @Test

     public void C() {

         System.out.println("Test method C");

     }

     @Test

     public void B() {

         System.out.println("Test method B");

     }

}

Test method B

Test method C

Test method A

PASSED: B

PASSED: C

PASSED: A

Group in TestNG:

**public** **class** groupExamples {  
  
 @Test(groups="Regression")   
**public** **void** **testCaseOne**() {  
 System.**out**.println("Im in testCaseOne - And in Regression Group");   
}

@Test(groups="Regression")   
**public** **void** **testCaseTwo**(){   
System.**out**.println("Im in testCaseTwo - And in Regression Group");  
 }

@Test(groups="Smoke Test")   
**public** **void** **testCaseThree**(){  
 System.**out**.println("Im in testCaseThree - And in Smoke Test Group");   
}

@Test(groups="Regression")   
**public** **void** **testCaseFour**(){  
 System.**out**.println("Im in testCaseFour - And in Regression Group");   
}   
}

We will execute the group “Regression” which will execute the test methods which are defined with group as “Regression”

<suite name="Sample Suite">  
 <test name="testing">  
 <groups>  
 <run>  
 <include name="Regression"/>  
 </run>  
 </groups>

<classes>  
<class name="com.example.group.groupExamples" />  
 </classes>

</test>  
</suite>

**How would you make sure that a page is loaded using Selenium and Webdriver**

In Selenium, you can use the below lines of code to check for the successful loading of a web page. The best approach is by selecting an element from the page & stand by till it becomes clickable.

WebDriverWait wait=new WebDriverWait(driver,100);

wait.until(ExpectedConditions.anyElement(By.id(id)));

**How to launch a batch file in a Selenium Webdriver project?**

It's usual in a test suite to run a batch file or an executable file for setting up the pre-requisites before starting the automation. You can use the below Java code for this purpose.

Process batch =Runtime.getRuntime.exec("path of the batch file");

batch.waitFor();

**How to run the Selenium Webdriver test from the command line?**

java-classpath".;selenium-server-standalone-2.33.0.jar"SampleClass

Q: What are the different exceptions you face in Selenium Webdriver?

Ans.

WebDriverException,

NoAlertPresentException,

NoSuchWindowException,

NoSuchElementException,

TimeoutException.

ElementNotVisibleException

**Q: How would you automatically click a screenshot whenever any exception occurs?**

**Ans.**For this you will have to use <EventFiringWebDriver> class and needs to implement the <onException> method of the <WebDriverEventListener> interface.

File scrFile =((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

**How would you select any particular text using Selenium Webdriver?**

**Ans.**It seems an easy one at first but you need to do a little more to achieve this.

driver.get("/");

WebElement item =driver.findElement(By.xpath("//p[contains(text(),'Selenium webdriverquesions')]"));

Actions builder =new Actions(driver);

builder.doubleClick(item).build().perform();

**How would you fill a text field without calling the sendKeys()?**

Java code given below.

JavascriptExecutor JS =(JavascriptExecutor)driver;

JS.executeScript("document.getElementById(User').value='admin@testmail.com'");

JS.executeScript("document.getElementById('Pass').value='######'");

**How can you check the state of a checkbox/radio button?**

**Ans.**We can call the isSelected() method to test the status of these elements.

**Example Code:**

boolean test =driver.findElement(By.xpath("checkbox/radio button XPath")).isSelected();

**What is the process to start the IE/Chrome browser?**

**Ans.**If you want to start a browser then, just set the system properties as mentioned below.

// For the IE web browser.

System.setProperty("webdriver.ie.driver"," iedriver.exe file path");

WebDriver driver =new InternetExplorerDriver();

// For the Chrome web browser.

System.setProperty("webdriver.chrome.driver","chrome.exe file path");

WebDriver driver =new ChromeDriver();

**: How would you simulate the right click operation in WebDriver?**

**Ans.**You can make use of the Actions class features.

Actions builer=new Actions(driver);// Here, driver is the object of WebDriver class.

builder.moveToElement(element).perform();

builder.contextClick().perform();

**What is the FirefoxDriver, class or an interface? And which interface does it implement?**

**Ans.**FirefoxDriver is a Java class, and it implements the <WebDriver> interface. It contains the implementations of all the methods available in the <WebDriver> interface.

**What is the main difference between the close() and quit() methods?**

**Ans.**

**close() -**it closes the currently active browser window.

**quit()-**it will close all of the opened browser windows

**What is the best way to check for the highlighted text on a web page?**

**Ans.**Use the below code to verify the highlighted text for an element on the web page.

Stringclr=driver.findElement(By.xpath("//a[text()='TechBeamers']")).getCssValue("color");

Stringbkclr=driver.findElement(By.xpath("//a[text()='TechBeamers']")).getCssValue("background-color");

System.out.println(clr);

System.out.println(bkclr);

10. What is assertion in Selenium?

Assertion is used as a verification point in Selenium. It verifies that the state of the application is as expected. The three types of assertion are:

“assert”

“verify”

“waitFor

**How do you simulate browser back and forward ?**

driver.navigate().back();

driver.navigate().forward();

What are the different Selenium components?

The suite package constitutes of the following sets of tools :

Selenium Integrated Development Environment (IDE) – It is a Firefox plug-in which is easily installable along with other plug-ins. Since it has a simple framework, Selenium IDE should be used as prototyping tool. For advanced use cases, Selenium RC or WebDriver would be suitable.

Selenium Remote Control (RC) – It is a testing framework that allows the developers to write the code in any programming language. Usually it supports a wide spectrum of languages like Java, PHP, Python, Perl, C#, etc.

Selenium WebDriver – This tool has more advanced features than that of RC and IDE. It applies better approach to automate the browser activities. It does not typically rely upon JavaScript, it interacts with browser.

Selenium Grid – It works along with Selenium RC and runs tests on different nodes using different browsers simultaneously.

How to handle the AJAX popup window?

By using getWindowHandles() and obj.switchTo.window(windowid), a programmer manages pop-ups using explicit wait and driver.swtchTo.window(“name”) commands for your requirements.

How can you send text input to a focused element?

This can be done by simulating key presses on the focused element. One way is to perform “actions” on the web driver object:

new Actions(webDriver).sendKeys(“some text”).perform();

An alternative way is to switch to the active element first, and send keys to it directly:

driver.switchTo().activeElement().sendKeys(“some text”);

Why do we use headless drivers? How can you visually investigate test failure when using headless drivers?

Headless drivers are typically used in continuous integration (CI) setups. Headless drivers, such as PhantomJS, provide all standard web browser functionalities, but run in the command-line. These drivers are based on command-line tools and don’t produce screen output, making them ideal for completely automated setups.

To be able to visually investigate test failures, the developer needs to implement mechanisms to capture screenshots, otherwise rely on command line output.

How are absolute XPaths different from relative XPaths? Why are relative XPaths typically preferred over absolute XPaths in automated tests?

**Absolute** XPaths, in terms of web pages, start with the root element:

html/head/body/table/tbody/tr/th

**Relative** XPaths, on the other hand, usually start with “//”:

//table/tbody/tr/th

Even though both these XPaths probably refer to the same element on a certain web page, the former one is more likely to break with any change made to the page. For example, moving the table to inside of a div element will stop the absolute XPath from being able to locate the th element. On the other hand, the relative XPath will still continue to work.

What is a hybrid framework in Selenium?

A hybrid framework is a combination of keyword- and data-driven frameworks.

What is StaleElementException ?

It is the exception thrown when the invoked element is no longer attached to the DOM for any reason.

Question 6: Which web driver implementation is fastest?  
Answer: HTMLUnitDriver. Simple reason is HTMLUnitDriver does not execute tests on browser but plain http request – response which is far quick than launching a browser and executing tests. But then you may like to execute tests on a real browser than something running behind the scenes.

How to capture screen shot in Webdriver?

File file= ((TakesScreenshot)driver).getScreenshotAs(OutputType.FILE);

FileUtils.copyFile(file, new File("c:\\name.png"));

**Step 1)** Convert web driver object to TakeScreenshot

TakesScreenshot scrShot =((TakesScreenshot)webdriver);

**Step 2)**Call getScreenshotAs method to create image file

File SrcFile=scrShot.getScreenshotAs(OutputType.FILE);

### ****What is the use of JavaScriptExecutor?****

**JavaScriptExecutor** is an interface which provides a mechanism to execute Javascript through the Selenium WebDriver. It provides “**executescript**” and “**executeAsyncScript**” methods, to run JavaScript in the context of the currently selected frame or window. An example of that is:

|  |  |
| --- | --- |
|  | JavascriptExecutor js = (JavascriptExecutor) driver;  js.executeScript(window.scrollBy(0,500)); |

### ****12. How to scroll down a page using JavaScript in Selenium?****

We can scroll down a page by using window.scrollBy() function. Example:

|  |  |
| --- | --- |
|  | ((JavascriptExecutor) driver).executeScript("window.scrollBy(0,500)"); |

### ****How to scroll down to a particular element?****

To scroll down to a particular element on a web page, we can use the function **scrollIntoView()**. Example:

|  |  |
| --- | --- |
|  | ((JavascriptExecutor) driver).executeScript("arguments[0].scrollIntoView();", element); |

How to execute java scripts function.

Answer:

JavascriptExecutor js = (JavascriptExecutor) driver;

String title = (String) js.executeScript("pass your java scripts");

How to capture page title using Selenium 2.0?

Answer:

String title =  driver.getTitle()

How to count total number of rows of a table using Selenium 2.0?

Answer:

List <WebElement> rows = driver.findElements(By.className("//table[@id='tableID']/tr"));

inttotalRow = rows.size();

Question 17:  How to store page source using Selenium 2.0?

Answer:

String pagesource = driver.getPageSource()

How to store current url using selenium 2.0?

Answer:

String currentURL  =driver.getCurrentUrl()

//get text from element and stored in text variable

String attributeValue = el. getAttribute("AttributeName") ;

 How to double click on element using selenium 2.0?

Answer:

WebElement el  =  driver.findElement(By.id("ElementID"));

Actions builder = new Actions(driver);

builder.doubleClick(el).build().perform();

How to perform drag and drop in selenium 2.0?

Answer:

WebElement source  =  driver.findElement(By.id("Source ElementID"));

WebElement destination  =  driver.findElement(By.id("TagetElementID"));

Actions builder = new Actions(driver);

builder.dragAndDrop(source, destination ).perform();

 How to verify pdf content using selenium 2.0?

How to capture video of running scripts in selenium 2.0?

How to verify response 200 code using selenium 2.0?

 How to verify image using selenium 2.0?

What is different between findElement and findElements

### ****How can you fetch an attribute from an element? How to retrieve typed text from a textbox?****

We can fetch the attribute of an element by using the **getAttribute()** method. Sample code:

|  |  |
| --- | --- |
|  | WebElemente Login = driver.findElement(By.name(“Login”);  String LoginClassName = eLogin.getAttribute("classname"); |

to retrieve some text from any textbox, we can use getText() method. In the below piece of code I have retrieved the text typed in the ‘Login’ element.

|  |  |
| --- | --- |
|  | WebElementeLogin = driver.findElement(By.name(“Login”);  String LoginText = Login.getText (); |

I been a developer for 2 years in the beginning of my career, and then I switched to automation testing becoz I got a chance once to work on automation scripts and I totally loved it.

That was the I decided I will be completely switching to automation testing

The first thing to check is whether it is feasible to automate

Automation Advantages:

Faster execution

More accurate

Lesser investment in human resource

Supports regression testing

Frequent executions

Supports Lights out execution

Selenium is a suit of software tools to automate web browsers.  
It is open source mainly used for functional testing.

Selenium WebDriver, IDE, Grid, Remote Control

Advantages:

Open Source, Multiple PL support, Support different browsers, Support different OS

Assert and verify

When an “assert” command fails, the test execution will be aborted. So when the Assertion fails, all the test steps after that line of code are skipped. The solution to overcoming this issue is to use a try-catch block. We use the Assertion in the try catch block. Mostly, the assert command is used when the end result of the check value should pass to continue to the next step.

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

Verify:

When a “verify” command fails, the test will continue executing and logging the failure. Mostly, the Verify command is used to check non-critical things. In such cases where we move forward even though the end result of the check value is failed.

In simple words, there wont be any halt in the test execution even though the verify condition is true or false.

Reading from Excel

## Apache POI terminologies

Apache POI excel library revolves around following four key interfaces -

1. **Workbook:** A workbook is the high-level representation of a Spreadsheet.
2. **Sheet:** A workbook may contain many sheets. The sample excel file that we looked at in the previous section has two sheets - Employee and Department
3. **Row:** As the name suggests, It represents a row in the spreadsheet.
4. **Cell:** A cell represents a column in the spreadsheet.

#### HSSF and XSSF implementations -

Apache POI library consists of two different implementations for all the above interfaces.

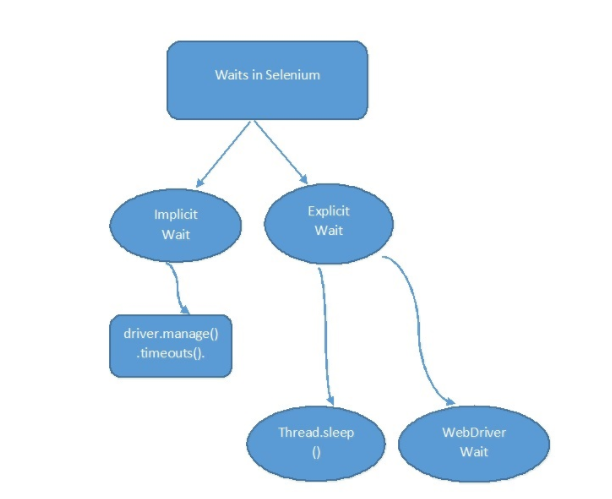
1. **HSSF (Horrible SpreadSheet Format):** HSSF implementations of POI’s high-level interfaces like HSSFWorkbook, HSSFSheet, HSSFRow and HSSFCell are used to work with excel files of the older binary file format - .xls
2. **XSSF (XML SpreadSheet Format):** XSSF implementations are used to work with the newer XML based file format - .xlsx. using this

Challenges faced in Automation.

A pop up window with a text area and an update link.

The text area have a huge XML. I have to update an ID with a unique random value on a particular XML tag and click on the update link.  
1. Switched to child window first.  
2. Saved the whole XML to a String.  
3. Generated a random 6 digit number using Random String Utils.  
4. Appended this numbers with a pre defined Alphabets.  
5. Updated the node value with the help of Document builder Factory & Transformer   
 Factory.  
6. Cleared the text area and Tried to use send keys with string values to text area, but it was not working as send keys have problem with 20,000 chars. (send keys will send letter by letter)  
7. Copied the String to Clipboard.  
8. Used send keys with Ctrl V combination to paste.  
9. Updated the popup XML with update link click.

Waits in Selenium:



Implicit, Explicit and Fluent

**Implicit**:

wait for a certain amount of time before throwing an **exception** that it cannot find the element on the page. We should note that implicit waits will be in place for the entire time the browser is open. This means that any search for elements on the page could take the time the implicit wait is set for.

**driver.manage().timeouts().implicitlyWait(10,TimeUnit.SECONDS);**

During Implicit wait if the Web Driver cannot find it immediately because of its availability, it will keep polling (around 250 milli seconds) the DOM to get the element. If the element is not available within the specified Time then, NoSuchElementException will be raised. The default setting is zero. Once we set a time, the Web Driver waits for the period of the WebDriver object instance.  
 In implicit wait the webdriver polls the DOM to check the availability of the webElement and waits till the maximum time specified before throwing NoSuchElementException.

**Explicit**:

**Using Web Driver wait:**  
It is more extendible in the means that you can set it up to wait for any condition you might like. Usually, you can use some of the prebuilt **ExpectedConditions** to wait for elements to become clickable, visible, invisible, etc.  
the explicit waits are applied to each and every webElement. In explicit wait, certain conditions are defined for which the webDriver instance waits before locating webElements or performing actions on them. Some of the most common conditions specified in explicit waits are-   
elementToBeClickable, presenceOfElementLocated etc.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | WebDriverWait wait = **new** WebDriverWait(driver, **15**);  wait.until(ExpectedConditions.presenceOfElementLocated(ElementLocator));  We can write a code such that we specify the XPath of the web element that needs to be visible on the page and then ask the WebDriver to wait for a specified time. Look at the sample piece of code below:   |  |  | | --- | --- | |  | WebDriverWait wait=newWebDriverWait(driver, 20);  Element = wait.until(ExpectedConditions.visibilityOfElementLocated(By.xpath( “<xpath”))); |   Similarly, we can write another piece of code asking the WebDriver to wait until an error appears like this:   |  |  | | --- | --- | |  | WebDriverWait wait=newWebDriverWait(driver, 20);  Element = wait.until(ExpectedConditions.alertIsPresent()); |   There can be instance when a particular element takes more than a minute to load. In that case you definitely not like to set a huge time to Implicit wait, as if you do this your browser will going to wait for the same time for every element.  To avoid that situation you can simply put a separate time on the required element only. By following this your browser implicit wait time would be short for every element and it would be large for specific element.  Thread.sleep(2000); |

**Fluent Wait:**

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

Each **FluentWait** instance defines the maximum amount of time to wait for a condition, as well as the frequency with which to check the condition. Furthermore, the user may configure the wait to ignore specific types of exceptions whilst waiting, such as **NoSuchElementExceptions** when searching for an element on the page

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

.withTimeout(30, TimeUnit.SECONDS)

.pollingEvery(5, TimeUnit.SECONDS)

.ignoring(NoSuchElementException.class);

Above code snippet will Wait 30 seconds for an element to be present on the page and check for its presence once every 5 seconds.”  
Each fluent wait instance defines the maximum amount of time to wait for a condition and we can give the frequency with which to check the condition.

### ****Ways To Handle Keyboard Keys:****

Handle Keyboard Keys using Action class.

Handle Keyboard Keys using sendkeys chord.

Handle Keyboard keys using Robot class.

### ****How to send ALT/SHIFT/CONTROL key in Selenium WebDriver?****

When we generally use ALT/SHIFT/CONTROL keys, we hold onto those keys and click other buttons to achieve the special functionality. So it is not enough just to specify **keys.ALT** or **keys.SHIFT** or **keys.CONTROL** functions.

For the purpose of holding onto these keys while subsequent keys are pressed, we need to define two more methods:**keyDown(modifier\_key)** and **keyUp(modifier\_key)**

Parameters: **Modifier\_key (keys.ALT or Keys.SHIFT or Keys.CONTROL)**Purpose: Performs a modifier key press and does not release the modifier key. Subsequent interactions may assume it’s kept pressed.

Parameters: **Modifier\_key (keys.ALT or Keys.SHIFT or Keys.CONTROL)**  
Purpose: Performs a key release.  
Hence with a combination of these two methods, we can capture the special function of a particular key.

|  |
| --- |
| Actions class:  Pressing SHIFT   Actions builder = **new** Actions(driver);  Action typeInCAPS = builder.keyDown(txtAppLname, Keys.***SHIFT***)  .sendKeys(txtAppLname, "amjed")  .keyUp(txtAppLname, Keys.***SHIFT***)  .build();  typeInCAPS.perform();  Actions builder = **new** Actions(driver);  Action typeInCAPS = builder.keyDown(txtAppLname, Keys.***SHIFT***).sendKeys(txtAppLname, "amjed").keyUp(txtAppLname, Keys.***SHIFT***).build();  typeInCAPS.perform();  Ctrl + A key press using ascii code  WebElement el = driver.FindElement(By.Id("an\_element\_id"));  char c = '\u0001'; // ASCII code 1 for Ctrl-A  el.SendKeys(Convert.ToString(c));  chord method    Actions a = **new** Actions(driver);                          //select the value which is typed in FirstName field                          a.sendKeys(txtFName, Keys.*chord*(Keys.***CONTROL***,"a")).perform();                          //Performing copy action using CTRl+C                          a.sendKeys(Keys.*chord*(Keys.***CONTROL***,"c")).perform();                          //Performing paste action using CTRL+V in LastName field                          a.sendKeys(txtLName, Keys.*chord*(Keys.***CONTROL***,"v")).perform(); |
|  |

### ****Can we enter text without using sendKeys()?****

Yes. We can enter/ send text without using **sendKeys()** method. We can do it using JavaScriptExecutor.

How do we do it?  
Using DOM method of, identification of an element, we can go to that particular document and then get the element by its ID (here login) and then send the text by value. Look at the sample code below:

|  |  |
| --- | --- |
|  | JavascriptExecutor js = (JavascriptExecutor) driver;  js.executeScript("document.getElementById(‘Login').value=Test text without sendkeys"); |

### ****Which technique should you consider using throughout the script “if there is neither frame id nor frame name”?****

If neither frame name nor frame id is available, then we can use **frame by index**.

Let’s say, that there are 3 frames in a web page and if none of them have frame name and frame id, then we can still select those frames by using frame (zero-based) index attribute. Each frame will have an index number. The first frame would be at index “0”, the second at index “1” and the third at index “2”. Once the frame has been selected, all subsequent calls on the WebDriver interface will be made to that frame.

|  |  |
| --- | --- |
| 1 | driver.switchTo().frame(0); |

### ****What is parameterization in TestNG? How to pass parameters using testng.xml?****

|  |  |
| --- | --- |
|  | publicclassParameterizedTest1{   @Test   @Parameters("myName")   Public void parameterTest(String myName) {   System.out.println("Parameterized value is : "+ myName);   }  } |

To pass parameters using testng.xml file, we need to use ‘parameters’ tag. Look at the below code for example:

|  |  |
| --- | --- |
|  | <?xmlversion="1.0"encoding="UTF-8"?>  <!DOCTYPEsuite SYSTEM "<http://testng.org/testng-1.0.dtd>" >   <suitename=”CustomSuite">    <testname=”CustomTest”>     <parametername="myName"value=”John"/>      <classes>       <classname="ParameterizedTest1"/>      </classes>    </test>   </suite> |

### ****Explain DataProviders in TestNG using an example. Can I call a single data provider method for multiple functions and classes?****

DataProvider is a TestNG feature, which enables us to write DataDriven tests. When we say, it supports DataDriven testing, then it becomes obvious that the same test method can run multiple times with different data-sets. DataProvider is in fact another way of passing parameters to the test method.

**@DataProvider** marks a method as supplying data for a test method. The annotated method must return an Object[] where each Object[] can be assigned to parameter list of the test method.

To use the DataProvider feature in your tests, you have to declare a method annotated by **@DataProvider** and then use the said method in the test method using the ‘dataProvider‘ attribute in the Test annotation.

As far as the second part of the question is concerned, Yes, the same DataProvider can be used in multiple functions and classes by declaring DataProvider in separate class and then reusing it in multiple classes.

### ****Explain what is Group Test in TestNG?****

In TestNG, methods can be categorized into groups. When a particular group is being executed, all the methods in that group will be executed.  We can execute a group by parameterizing it’s name in group attribute of **@Test** annotation. Example: @Test(groups={“xxx”})

|  |  |
| --- | --- |
|  | @Test(groups={“Car”})  publicvoiddrive(){  system.out.println(“Driving the vehicle”);  }    @Test(groups={“Car”})  publicvoidchangeGear() {  system.out.println("Change Gears”);  }    @Test(groups={“Car”})  publicvoidaccelerate(){  system.out.println(“Accelerating”);  } |

http://services.focusinfotech.com/uploading-files-through-selenium-using-autoit/

AutoIT:

File Upload:

* ControlFocus(“title”, “ text”, “ControlID”)
* ControlSetText(“title”, “text ”, “ControlID”, “Path”)
* ControlClick(“title”, “text”, “ControlID” )

Eg:

ControlFocus("Open","","Edit1")

ControlSetText("Open","","Edit1","E:\01Desk\CAS\TestData\kerala.jpg")

ControlClick("Open","","Button1")

**File file = new File(path);  
FileInputStream fis =new FileInputStream(file);  
XSSFWorkbook wb = new XSSFWorkbook(fis);   
XSSFSheet sheet = wb.getSheetAt(0);**

            Iterator<Row> rowIterator = sheet.iterator();

while(rowIterator.hasNext()) {

             Row row = rowIterator.next();

              Iterator<Cell> cellIterator = row.cellIterator();

  while(cellIterator.hasNext()) {

              Cell cell = cellIterator.next();

                    // Check the cell type and format accordingly

                    switch(cell.getCellType()) {

                    caseCell.CELL\_TYPE\_NUMERIC:

                        System.out.print(cell.getNumericCellValue() + "t");

                        break;

                    caseCell.CELL\_TYPE\_STRING:

                        System.out.print(cell.getStringCellValue() + "t");

                        break;

                    }

                }

                System.out.println("");

            }

            file.close();

Reading from xlsx directly with Filepath, SheetName/Index, Row, column

Public void readExcel()

{

File file =**new** File(".\\TestData\\new.xlsx";);

FileInputStream fis=**new** FileInputStream(file);

XSSFWorkbook wb=**new** XSSFWorkbook(fis);

String value=wb.getSheetAt(0).getRow(0).getCell(0).toString(); //or get getSheet()

System.***out***.println(value);

}

Switching into Frame:

driver.switchTo().frame(1);

Switiching back :

driver.switchTo().parentFrame();

driver.switchTo().defaultContent();

TestNG Listeners:

Listener is defined as interface that modifes the default TestNG's behavior. As the name suggests Listeners "listen" to the event defined in the selenium script and behave accordingly. It is used in selenium by implementing Listeners Interface. It allows customizing TestNG reports or logs. There are many types of TestNG listeners available.

Below are the few TestNG listeners:

1. IAnnotationTransformer ,
2. IAnnotationTransformer2 ,
3. IConfigurable ,
4. IConfigurationListener ,
5. IExecutionListener,
6. IHookable ,
7. IInvokedMethodListener ,
8. IInvokedMethodListener2 ,
9. IMethodInterceptor ,
10. IReporter,
11. ISuiteListener,
12. ITestListener

ITestListener has following methods

* **OnStart-** OnStart method is called when any Test starts.
* **onTestSuccess-** onTestSuccess method is called on the success of any Test.
* **onTestFailure-** onTestFailure method is called on the failure of any Test.
* **onTestSkipped-**onTestSkippedmethod is called on skipped of any Test.
* **onTestFailedButWithinSuccessPercentage-**method is called each time Test fails but is within success percentage.
* **onFinish-**onFinish method is called after all Tests are executed.

By using TestNG listeners 'ITestListener' or 'TestListenerAdapter' we can change the default behaviour write our own implementation when a Test fails or Skips etc.

TestNG listeners are set of class exposed into TestNG framework which we can utilize to modify default TestNG’s behavious.  
• As the name suggests Listeners “listen” to the event and behave accordingly.  
• It allows customizing TestNG reports or logs or to take a screen shot.  
• Listeners contains unimplemented methods(blank body and we can customize it)

When running TestNG tests, one could want to perform some common actions – after each test has finished successfully, after each failed test, after each skipped test, or after all the tests have finished running, no matter their result. To apply such a common behavior to a group of tests, a custom listener can be created, that implements TestNG’s ITestListener interface. There are two types of methods to implement: a set of them are related to a single test’s run (onTestStart, onTestSuccess, onTestFailure, onTestSkipped, onTestFailedButWithinSuccessPercentage), the other to the whole suite’s run (onStart, onFinish).

These interfaces are used in selenium to generate logs or customize the testing reports with the help of Step by Step Scenarios

###### Implementing the custom listener

The custom listener that will be used must implement all the ITestListener interface’s methods, as depicted below:

public class SimpleTestListener implements ITestListener{

@Override

public void onTestStart(ITestResult result) {

}

@Override

public void onTestSuccess(ITestResult result) {

}

@Override

public void onTestFailure(ITestResult result) {

}

@Override

public void onTestSkipped(ITestResult result) {

}

@Override

public void onTestFailedButWithinSuccessPercentage(ITestResult result) {

}

@Override

public void onStart(ITestContext context) {

}

@Override

public void onFinish(ITestContext context) {

}

TestNG parameters.

|  |  |
| --- | --- |
|  | <suite name="Suite">        <test name="ToolsQA">    <parameter name="sUsername" value="testuser\_1"/>    <parameter name="sPassword" value="Test@123"/>    <classes>        <class name="automationFramework.TestngParameters" />    </classes>        </test>    </suite> |

 @Test

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

  public void test(String sUsername, String sPassword) {

Desired Capabilities:

**Desired Capabilities** is a class in Selenium used to set properties of browsers to perform cross browser testing of web applications. It stores the capabilities as key-value pairs and these capabilities are used to set browser properties like browser name, browser version, path of browser driver in the system, etc. to determine the behaviour of browser at run time.

Desired capability is a series of value or key pairs that stores the web browser properties such as browser name, browser version, the path of the browser in the system etc which will help to determine the behavior of the browser at run time. Further, desired capability can be used for configuring the web driver instance of selenium webdriver.

There are different types of desired capabilities methods and some of them are as follows.

* getBrowserName()
* setBrowserName()
* getVersion()
* setVersion()
* getPlatform()
* setPlatform()
* getCapability Method
* setCapabilityMethod

Cookies in Selenium:

Each cookie is associated with a name, value, domain, path, expiry, and the status of whether it is secure or not. In order to validate a client, a server parses all of these values in a cookie.

When Testing a web application using selenium web driver, you may need to create, update or delete a cookie.

For example, when automating Online Shopping Application, you many need to automate test scenarios like place order, View Cart, Payment Information, order confirmation, etc.

If cookies are not stored, you will need to perform login action every time before you execute above listed test scenarios. This will increase your coding effort and execution time.

The solution is to store cookies in a File. Later, retrieve the values of cookie from this file and add to it your current browser session. As a result, you can skip the login steps in every Test Case because your driver session has this information in it.

The application server now treats your browser session as authenticated and directly takes you to your requested URL.

// create file named Cookies to store Login Information

File file = new File("Cookies.data");

**Get Cookies:**This statement is used to return the list of all Cookies stored in web browser.

driver.manage().getCookies();

**Get Cookies by name:** This statement is used to return the specific cookie according to its name.

manage().getCookieNamed(arg0);

*Add Cookies: This statement is used to create and add the cookie.*

*manage().addCookie(arg0)*

Delete Cookies: This statement is used to delete a specific cookie.

*manage().deleteCookie(arg0);*

Set<Cookie> cookiesForCurrentURL = driver.manage().getCookies();

**for** (Cookie cookie : cookiesForCurrentURL) {

System.out.println(" Cookie Name - " + cookie.getName()

+ " Cookie Value - " + cookie.getValue()));

}

1. ***deleteCookieNamed(String cookieName)****- To delete a particular Cookie by name.*
2. ***deleteCookie(Cookie cookie)****- To delete a particular Cookie.*
3. ***deleteAllCookies()***

GIT



Install> Get bash here

>git init  
>git remote add origin <https://github.com/amjed4u/CAS.git>

Git status

add to index> then commit

Adding to index : git add filename

Commit:: git commit –m “Message”

Git log

PULL :: git pull origin master

Creating branch : git branch firstbranch

Switching to new branch :git checkout firstbranch

MERGE :: Merging branches to Master

Git merge firstbranch

Generating SSH Key : Ssh-keygen

Viewing key cat /c/users/Amjed/.ssh/id\_rsa.pub

SSH-T [git@github.com](mailto:git@github.com)

Git push origin master

Git push –f origin master (forced push)

**commit**: adding changes to the local repository

**push**: to transfer the last commit(s) to a remote server

 Discard the local changes

using git reset --hard. or git checkout -t -f remote/branch

a) Discard local changes for a specific file

using git checkout filename

Git pulling discarding the local changes

$ git fetch --all

(2) Then reset the master:

$ git reset --hard origin/master

(3) Pull/update:

$ git pull

To ignore a file

Create a text file as .gitignore

Add the files names to ignore

Eg : \*html

\*.txt

/target

Git status will display all the unstaged files

Commit history : git log

To add all files to staging area git add.

Difference between working tree, staging area and Repository

Working tree and staging area : git diff

Staging and repository : git diff –staged

Removing a file git rm filename

Branches:

Head is pointing to master

First create a branch : git branch feature

Switching to feature branch : git checkout feature

Now the head will point to feature branch

Switching back to master : git checkout master

To list branches git branch -a

Delete branches:

Before that check which branch is pending to merge : git branch –merged

Delete : git branch –d branchName

Merges 2 types; Fast forward merge & 3 way merge

In a collaborative environment, it is common for several developers to share and work on the same source code. While some developers will be fixing bugs, others will be implementing new features, etc. With so much going on, there needs to be a system in place for managing different versions of the same code base.

Branching allows each developer to branch out from the original code base and isolate their work from others. It also helps Git to easily merge versions later on.

A Git branch is essentially an independent line of development. You can take advantage of branching when working on new features or bug fixes because it isolates your work from that of other team members.

Different branches can be merged into any one branch as long as they belong to the same repository.

You start a new branch off of the original master so that you can make changes to the project without affecting the original master branch. Whoever else is working on the project can continue with the master branch as well. Then, when you finish your changes you can merge back with the master branch and your changes will be present, along with whatever else was changed in the meantime.

To Avoid Conflicts

Keep your changes small.  
Rebase with your main branch (generally master) when it changes.  
Review pull requests faster and merge them to main branch.

* Untracked: the file exists, but is not part of git's version control
* Staged: the file has been added to git's version control but changes have not been committed
* Committed: the change has been committed

To stage a file is simply to prepare it finely for a commit. Git, with its index allows you to commit only certain parts of the changes you’ve done since the last commit.

Staging :preparing for Commit

In Git, just because a file was modified doesn't mean it will be automatically included in the next commit. Instead, you have to tell Git **explicitly** which of your modifications shall be part of the next commit. This is done by adding a change to the Staging Area or, put simply, by "staging" it.

A change can be as granular as a single changed line in a file, leading to very precise commits. If, after staging a change, you decide you don't want that change to go into the next commit, you can also "unstage" it, again.

### git config

Usage: git config –global user.name “[name]”

Usage: git config –global user.email “[email address]”

This command sets the author name and email address respectively to be used with your commits.

### git init

Usage: git init [repository name]

This command is used to start a new repository.

### git clone

Usage: git clone [url]

This command is used to obtain a repository from an existing URL.

### git add

Usage: git add [file]

This command adds a file to the staging area.

### git commit

Usage: git commit -m “[ Type in the commit message]”

This command records or snapshots the file permanently in the version history.

Usage: git commit -a

This command commits any files you’ve added with the git add command and also commits any files you’ve changed since then.

### git diff

Usage: git diff

This command shows the file differences which are not yet staged.

 Usage: git diff –staged

This command shows the differences between the files in the staging area and the latest version present.

Usage: git diff [first branch] [second branch]

This command shows the differences between the two branches mentioned.

### git reset

Usage: git reset [file]

This command unstages the file, but it preserves the file contents.

Usage: git reset [commit]

This command undoes all the commits after the specified commit and preserves the changes locally.

Usage: git reset –hard [commit]  This command discards all history and goes back to the specified commit.

### git status

Usage: git status

This command lists all the files that have to be committed.

### git rm

Usage: git rm [file]

This command deletes the file from your working directory and stages the deletion.

### git branch

Usage: git branch

This command lists all the local branches in the current repository

Usage: git branch [branch name]

This command creates a new branch.

Usage: git branch -d [branch name]

This command deletes the feature branch.

### git checkout

Usage: git checkout [branch name]

This command is used to switch from one branch to another

Usage: git checkout -b [branch name]

This command creates a new branch and also switches to it.

### git merge

Usage: git merge [branch name]

This command merges the specified branch’s history into the current branch.

### git remote

Usage: git remote add [variable name] [Remote Server Link]

This command is used to connect your local repository to the remote server.

### git push

Usage: git push [variable name] master

This command sends the committed changes of master branch to your remote repository.

Usage: git push [variable name] [branch]

This command sends the branch commits to your remote repository.

Usage: git push –all [variable name]

This command pushes all branches to your remote repository.

Usage: git push [variable name] :[branch name]

This command deletes a branch on your remote repository.

### git pull

Usage: git pull [Repository Link]

This command fetches and merges changes on the remote server to your working directory.

### git stash

Usage: git stash save

This command temporarily stores all the modified tracked files.

Usage: git stash pop

This command restores the most recently stashed files.

Usage: git stash list

This command lists all stashed changesets.

Usage: git stash drop

This command discards the most recently stashed changeset.

MANUAL

Bug &Defect :

**BUG**: A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer. A programming error that causes a program to work poorly, produce incorrect results or crash. An error in software or hardware that causes a program to malfunction. Bug is terminology of Tester.

DEFECT:  It can be simply defined as a variance between expected and actual. Defect is an error found AFTER the application goes into production. It commonly refers to several troubles with the software products, with its external behavior or with its internal features. In other words Defect is the difference between expected and actual result in the context of testing.  It is the deviation of the customer requirement.

STLC:

Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

Requirement Analysis 🡪 Test Planning 🡪 Test Case Development 🡪 Environment setup 🡪 Test Execution 🡪 Test Cycle Closure.

Requirement Analysis

Requirement Analysis is the very first step in **Software Testing Life Cycle (STLC)**. In this step Quality Assurance (QA) team understands the requirement in terms of what we will testing & figure out the testable requirements. If any conflict, missing or not understood any requirement, then QA team follow up with the various stakeholders like Business Analyst, System Architecture, Client, Technical Manager/Lead etc to better understand the detail knowledge of requirement.

From very first step QA involved in the where STLC which helps to prevent the introducing defects into Software under test. The requirements can be either Functional or Non-Functional like Performance, Security testing. Also requirement and Automation feasibility of the project can be done in this stage

Deliverables

RTM

Automation feasibility report.

Test Planning:

Test Planning is most important phase of Software testing life cycle where all testing strategy is defined. This phase also called as Test Strategy phase. In this phase typically Test Manager involved to determine the effort and cost estimates for entire project. This phase will be kicked off once the requirement gathering phase is completed & based on the requirement analysis, start preparing the Test Plan. The Result of Test Planning phase will be [Test Plan](http://www.softwaretestingclass.com/test-plan-template/)or Test strategy & Testing [Effort estimation](http://www.softwaretestingclass.com/software-estimation-techniques/) documents. Once test planning phase is completed the QA team can start with test cases development activity.

Test plan /strategy document.

Effort estimation document.

Test Case Development:

The test case development activity is started once the test planning activity is finished. This is the phase of STLC where testing team write down the detailed test cases. Along with test cases testing team also prepare the test data if any required for testing. Once the test cases are ready then these test cases are reviewed by peer members or QA lead.

Also the Requirement Traceability Matrix (RTM) is prepared. The Requirement Traceability Matrix is an industry-accepted format for tracking requirements where each test case is mapped with the requirement. Using this RTM we can track backward & forward traceability.

Test cases/scripts

Test data

Test Environment Setup:

Setting up the test environment is vital part of the STLC. Basically test environment decides on which conditions software is tested. This is independent activity and can be started parallel with Test Case Development. In process of setting up testing environment test team is not involved in it. Based on company to company may be developer or customer creates the testing environment. Mean while testing team should prepare the smoke test cases to check the readiness of the test environment setup.

Environment ready with test data set up

Smoke Test Results.

Test Execution:

Once the preparation of Test Case Development and Test Environment setup is completed then test execution phase can be kicked off. In this phase testing team start executing test cases based on prepared test planning & prepared test cases in the prior step.

Once the test case is passed then same can be marked as Passed. If any test case is failed then corresponding defect can be reported to developer team via bug tracking system & bug can be linked for corresponding test case for further analysis. Ideally every failed test case should be associated with at least single bug. Using this linking we can get the failed test case with bug associated with it. Once the bug fixed by development team then same test case can be executed based on your test planning.

If any of the test cases are blocked due to any defect then such test cases can be marked as Blocked, so we can get the report based on how many test cases passed, failed, blocked or not run etc. Once the defects are fixed, same Failed or Blocked test cases can be executed again to retest the functionality.

Completed RTM with execution status

Test cases updated with results

Defect reports

Test Cycle Closure:

Call out the testing team member meeting & evaluate cycle completion criteria based on Test coverage, Quality, Cost, Time, Critical Business Objectives, and Software. Discuss what all went good, which area needs to be improve & taking the lessons from current STLC as input to upcoming test cycles, which will help to improve bottleneck in the STLC process. Test case & bug report will analyze to find out the defect distribution by type and severity. Once complete the test cycle then test closure report & Test metrics will be prepared. Test result analysis to find out the defect distribution by type and severity.

Test Closure report

Test metrics

**Entry Criteria:**Entry Criteria gives the prerequisite items that must be completed before testing can begin.

**Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded

AGILE:

Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end.

AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. Both development and testing activities are concurrent unlike the Waterfall model

Process flow of Scrum Methodologies:

Each iteration of a scrum is known as Sprint  
Product backlog is a list where all details are entered to get end product  
During each Sprint, top items of Product backlog are selected and turned into Sprint backlog  
Team works on the defined sprint backlog  
Team checks for the daily work  
At the end of the sprint, team delivers product functionality

Burn Down Chart:

Burn down and burn up charts are two types of charts that project managers use to track and communicate the progress of their projects. A [burn down chart](http://www.clariostechnology.com/productivity/blog/whatisaburndownchart) shows how much work is remaining to be done in the project, whereas a [burn up](http://www.clariostechnology.com/productivity/blog/whatisaburnupchart) shows how much work has been completed, and the total amount of work.

Time Horizontal axis, work vertical axis.

Product backlog and Sprint backlog

The product backlog is compiled of all the things that must be done to complete the whole project. But it’s not just a simple list. An effective product backlog breaks down each of the items on the list into a series of steps that helps the development team.

The sprint backlog is like a subset of the product backlog. The sprint backlog comes from the product backlog, but it contains only that item, or those items, that can be completed during each sprint.

Testing Techiques.

Equivalence partitioning

Boundary value Analysis

 Test the login feature of a web application?

Checking input fields with positive and negative values, invalid email, valid email but incorrect password, sql injection, etc.

Possible answers to this testing interview question can be:

Sign in with valid login, Close browser and reopen and see whether you are still logged in or not.

Session management is important – how do we keep track of logged in users, is it via cookies or web sessions?

Sign in, then logout and then go back to the login page to see if you are truly logged out.

Login, then go back to the same page, do you see the login screen again?

Sign in from one browser, then open another browser to see if you need to sign in again?

Login, change password, and then logout, then see if you can login again with the old password.

Seesion and Cookies :

[cookie](http://en.wikipedia.org/wiki/HTTP_cookie): a small amount of information sent by a server to a browser, and then sent back by the browser on future page requests

Cookies are files, often including unique identifiers, that are sent by web servers to web browsers, and which may then be sent back to the server each time the browser requests a page from the server.

Cookies can be used by web servers to identity and track users as they navigate different pages on a website, and to identify users returning to a website.

Cookies may be either "persistent" cookies or "session" cookies. A persistent cookie consists of a text file sent by a web server to a web browser, which will be stored by the browser and will remain valid until its set expiry date (unless deleted by the user before the expiry date). A session cookie, on the other hand, will expire at the end of the user session, when the web browser is closed.

A cookie is a small piece of text stored on a user's computer by their browser. Common uses for cookies are authentication, storing of site preferences, shopping cart items, and server session identification.

a cookie will contain a string of text that contains information about the browser. To work, a cookie does not need to know know where you are from, it only needs to remember your browser.

A cookie is a small piece of text stored on a user's computer by their browser. Common uses for cookies are authentication, storing of site preferences, shopping cart items, and server session identification.

Cookie basics:

The first time a browser connects with a particular server, there are no cookies.

When the server responds it includes a Set-Cookie: header that defines a cookie.

Each cookie is just a name-value pair.

In the future whenever the browser connects with the same server, it includes a Cookie: header containing the name and value, which the server can use to connect related requests.

What's in a cookie?

Name and data.

Data size limited by browsers (typically < 4 KB).

A server can define multiple cookies with different names, but browsers limit the number of cookies per server (around 50).

Domain for this cookie: server, port (optional), URL prefix (optional). The cookie is only included in requests matching its domain.

Expiration date: browser can delete old cookies.

Cookies are mainly used for three purposes:

Session management

Logins, shopping carts, game scores, or anything else the server should remember

Personalization

User preferences, themes, and other settings

Tracking

Recording and analyzing user behavior

Cookies also can be used to save forms and passwords. Notice, when you start to type e-mail address, it automatically shows you options of e-mail addresses previously logged in. If you save the password, the cookies will also automatically save password and keep you logged in the website. There are various different types of cookies: Session cookie, Persistent cookie, Secure cookie, HttpOnly cookie, Third-party cookie, Supercookie and Zombie cookie.

Session :

Cookies are generally stored on the client-side machine, while sessions are stored on the server-side machine.

the session is a data structure that an application uses to store temporary data that is useful only during the time a user is interacting with the application, it is also specific to the user.

A session requires a cookie to be set on the browser to maintain the session.

Web Server, Application Server, Database Server

Web Server :

Tomcat, Jetty, IIS

A Web server can be a software (in the form of an application program) or hardware (in the form of a computer). Its basic function is to accept the HTTP (Hypertext Transfer Protocol) requests from the client side, then process and serve them back as the HTTP response along with the optional data content. The responses sent are in the form of HTML (Hypertext Markup Language) web pages or documents. The basic function of the Web server is to transfer the web content, which are accessed through the internet to the respective Clients.

Application Server:

JBoss, Web Logic, Web Sphere WAS,

While a web server handles mainly HTTP protocols, the application server deals with several different protocols, including, but not limited, to HTTP.

The Web server's main job is to display the site content and the application server is in charge of the logic, the interaction between the user and the displayed content. The application server is working in conjunction with the web server, where one displays and the other one interacts.

Scenario 1: Web server without an application server

you have an online store with only a web server and no application server. The site will provide a display where you can choose a product from. When you submit a query, the site performs a lookup and returns an HTML result back to its client. The web server sends your query directly to the database server (be patient, I will explain this one in our next nugget) and waits for a response. Once received, the web server formulates the response into an HTML file and sends it to your web browser. This back and forth communication between the server and database server happens every time a query is run.

Scenario 2: Web server with an application server

if the query you want to run has already been done previously and no data has changed since then, the server will generate the results without having to send the request to the database server. This allows a real-time query where a second client can access the same info and receive real time, reliable information without sending another duplicate query to the database server. The server basically acts as an intermediate between the database server and the web server. This allows the information pulled to be reusable while in the first scenario, since this info is embedded in a particular and "customized" HTML page, this is not a reusable process. A second client will have to request the info again and receive another HTML embedded page with the info requested -highly inefficient. Not to mention that this type of server is very flexible due to its ability to manage its own resources, including security, transaction processing, messaging and resource pooling.

Test Harness:

There are two contexts to where Test harness is used:

Automation testing

Integration Testing

In **the automation testing world,** Test harness refers to the framework and the software systems that contain the test scripts, parameters necessary) to run these scripts, gather test results, compare them and monitor the results.

**Test Scenario :**

Test Scenario gives the idea of what we have to test. Test Scenario is like a high-level test case.

Assume that we need to test the functionality of a login page of Gmail application. Test scenario for the Gmail login page functionality as follows:

**Test Scenario Example:** Verify the login functionality

Test Cases

Test cases are the set of positive and negative executable steps of a test scenario which has a set of pre-conditions, test data, expected result, post-conditions and actual results.

Test Case answers “**How to be tested**”

Assume that we need to test the functionality of a login page of Gmail application. Test cases for the above login page functionality as follows:

**Test Case 1:** Enter valid User Name and valid Password  
**Test Case 2:** Enter valid User Name and invalid Password  
**Test Case 3:** Enter invalid User Name and valid Password  
**Test Case 4:** Enter invalid User Name and invalid Password

Error Codes:

4xx Client Error  
5xx Server Error

400 Bad Request  
401 Unauthorized  
402 Payment Required  
403 Forbidden

404 Not Found

413 Request Entity Too Large

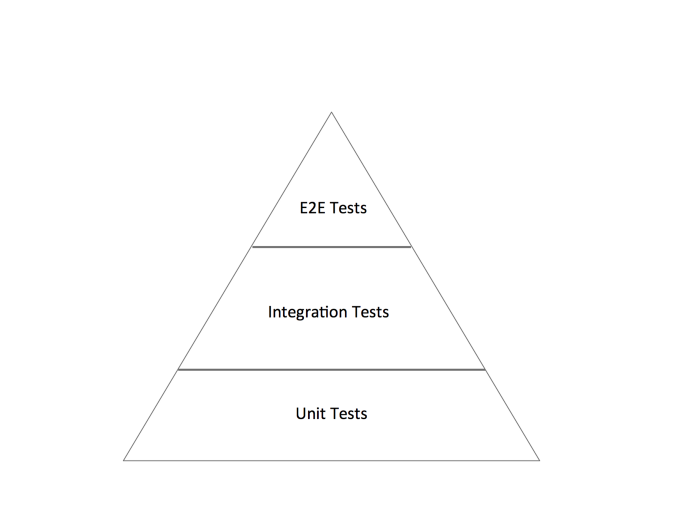
500 Internal Server Error  
502 Bad Gateway

503 Service Unavailable

504 Gateway Timeout

TestingPyramid

A testing pyramid is a pyramid of where all the different types of tests fits.



*The bulk of your tests are unit tests at the bottom of the pyramid. As you move up the pyramid, your tests gets larger, but at the same time the number of tests (the width of your pyramid) gets smaller.*

The Testing Pyramid is triangular for a reason: **there should be more tests at the bottom and fewer tests at the top**. Why?

AGILE

**Purpose:**Sprint planning sets up the entire team for success throughout the sprint. Coming into the meeting, the product owner will have a prioritized product backlog. They discuss each item with the development team, and the group collectively estimates the effort involved. The development team will then make a sprint forecast outlining how much work the team can complete from the product backlog. That body of work then becomes the sprint backlog.

### Sprint planning meeting

### Daily Scrum

### Sprint review meeting : Demo

### Sprint retrospective meeting : what went well and what didn't go well

**Burndown Chart**

As a definition of this chart we can say that the Burndown chart displays the remaining effort for a given period of time.

When they track product development using the Burndown chart, teams can use a sprint Burndown chart and a release Burndown chart. This article concentrates on the sprint Burndown chart as it is used on daily basis.

#### 2. What is Velocity?

Answer: Velocity question is generally posed to understand if you have done some real work and familiar with the term. Its definition “Velocity is the rate at which team progresses print by sprint” should be enough. You can also add saying the important feature of velocity that it can’t be compared to two different scrum teams

#### Do you know about Agile Manifesto & its Principles? Explain in brief.

Answer: This is the theory which most of agile/scrum roles aspirant should be on tips. Four manifesto values and 12 principles should be explained as much as possible as part of this question. Even if it’s not explained in 100% accurate manner it should be fine but intentions of values and principles should come out e.g.

* Working Software should be demonstrated at regular intervals
* Individuals & interaction – self-organization, self-motivating should be encouraged
* Customer collaboration
* Welcoming change at any point in time in the project

#### What is the use of burn-up and burn-down charts?

Answer: The burn-up chart illustrates the amount of completed work in a project whereas the burn-down chart depicts the amount of work remained to complete a project. Thus, the burn-up and burn-down charts are used to trace the progress of a project.

**Spike –** Spike is the type of story that can be taken between the sprints. Spikes are commonly used for the activities related to the design or technical issues such as research, design, prototyping, and exploration. There are two types of spikes – functional spikes and technical spikes.

#### What do you know about a story point in Scrum?

Answer: A story point in Scrum is the unit for the estimation of total efforts that are required to perform or complete a particular task. So, here is how you can answer such agile scrum interview questions on a single line.

SQL

**Distinct**

The SQL DISTINCT clause is used to remove duplicates from the result set of a SELECT statement.

**Finding Unique Values in a Column**

**selectdistinct** USER\_TYPE\_CODE

**from** CAS\_USER

DEALER

FINSOURCE

RTONE

SYSTEM

DMS

**Finding Unique Values in Multiple Columns**

**selectdistinct** USER\_TYPE\_CODE,USER\_LANG\_PREF

**from** CAS\_USER

SYSTEM EN

FINSOURCE EN

RTONE EN

DEALER EN

DEALER FR

DMS EN

DMS FR

FINSOURCE FR

# ORDER BY

**select\***

**from** CAS\_USER

**orderby** USER\_FRST\_NAME

default is ASC add DESC if we want to order from last

aggregate functions

**AVG** – calculates the **average** of a set of **values**.

**COUNT** – **counts rows** in a specified table or view.

**MIN** – gets the **minimum** value in a set of **values**.

MAX – gets the **maximum** value in a set of **values**.

**SUM** – calculates the **sum** of **values**.

**selectcount**(**\*** )

**from** CAS\_USER

40536

GROUP BY:

The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions. i.e if a particular column has same values in different rows then it will arrange these rows in a group.

**SELECT** department,

**MIN**(salary) **AS** "Lowest salary"

**FROM** employees

**GROUP** **BY** department;

**HAVING Clause**

SELECT NAME, SUM(SALARY) FROM Employee

GROUP BY NAME

HAVING SUM(SALARY)>3000;

This will add the salary of persons with same name.

**SELECT column-names  
 FROM table-name  
 WHERE condition  
 GROUPBY column-names  
 HAVING condition  
 ORDERBY column-names**

Top 2nd salary from the employee table.

Using MAX

Select MAX(Salary)   
from Employee  
where Salary NOT IN (Select MAX(Salary) from Employee)

USING LIMIT

Select DISTINCT(salary)  
From Employee  
Order by Salary DESC  
LIMIT n,m

n: nth record starting from index 0  
m :no of records from n

Eg:

1,1 : 2nd record only.

1,2 : 2nd and 3rd record

2,2 : 3rd record and 4th record

* **INNER JOIN**: Returns records that have matching values in both tables
* **LEFT JOIN**: Return all records from the left table, and the matched records from the right table
* **RIGHT JOIN**: Return all records from the right table, and the matched records from the left table
* **FULL JOIN**: Return all records when there is a match in either left or right table

Inner Join:

**SELECT** columns

**FROM** table1  **INNER** JOIN table2

**ON** table1.**column** = table2.**column**;

Web Services SOAP UI

Jenkins:

Install Java

Install Jenkins

MINUTE (0-59), HOUR (0-23), DAY (1-31), MONTH (1-12), DAY OF THE WEEK (0-6)

<http://localhost:8080>

Running from Maven:

Global Config: java and maven set up

Configure System :

Point to pom.XML

If we want to run the scripts from local machine, point pom.xml from local directory.

Otherwise point pom.xml fromGitHub

To Run Every 15 min :H/15 \* \* \* \*

0 13,14,15,16,17,18,19,20,21 \* \* \*

Jenkins to used to avoid Integration hell.

SO Jenkins is doing continues Integration.

Integration hell means breaking everything during final integration

Crone Examples:

Total 5 stars: Minute,Hour,Day,Month,Day(s) of week

Every Minute : \* \* \* \* \*  
Every 5 minutes : \*/5 \* \* \* \*

Every 30 Min : \*/30 \* \* \* \*  
Every Hour : 0 \* \* \* \*

10th June 8.30 AM : 30 08 10 06 \*

JENKINS Pipeline:

Pipeline is a plugin

Pipeline is a workflow with group of events or jobs that are chained and integrated with eachother in sequence.

(jenkins\_url)/safeRestart.

(jenkins\_url)/restart

* **[jenkins\_url]/safeRestart**. – This will restart Jenkins after the current builds have completed.
* **[jenkins\_url]/restart –** This will force a restart. Builds will not wait to complete.

**Jenkins is integrated with 2 components what are they?**

* The 2 components are:  
  Tools like apache maven & SVN, GIT(version control systems).

**What are the various ways in which build can be scheduled in Jenkins ?**

* Builds can be triggered by source code management  commits.
* Can be triggered after completion of other builds.
* Can be scheduled to run at specified time ( crons )
* Manual Build Request

JMeter:

Listeners : For seeting the results

View Results in Table  
View Results in Tree

Aggregate Report

Graph Result

Summary Report

Simple Data writer : Write all results to file  
During heavy load always use Data Writer Listener

Can save all Listener result to file name from browse option

Assertions : Checks the Response

Response Asserion

Duaration Assertion

Size Assertion

HTML Assertion

XML Assertion

Xpath Assertion

Latency : time to 1st byte  
Started sending request : time =0ms

Started receiving response 1010 ms. 1010 is the Latency

Executng on non GUI mode:

Open cmd and navigate to j meter bin folder

Command:

Jmeter –n –t (location to script) –l (location to result file)

LINUX

There are different directories :

Bin, dev, lib, media, opt, root, sbin, srv, tmp, var, boot, etc, init, lib64, mnt, proc, run, swap, sys, usr

Home : contains user accounts. Say amjed  
mnt : contains all drives  
pwd =print working directory   
cd = move to home directory  
$ = Normal User  
# = Root User  
su= switch user  
whoami = prints the user name  
passwd = change password

To swith to root :su root

To switch to user = suamjed

We can drirectly navigate to any of the basic directories by typing cd /directoryname.  
For eg : cd /mnt  
cd /home etc..

Start the command with / means navigate from base directories like bin, sbinetc  
eg: cd /mnt/d = navigate to drive (absolute path)

Tilede ~ = Takes to home directory  
 cd~ = navigates back to home directory (/home/amjed)

Cd .. = one step back

TO create a new directory:  
Mkdirdirectoryname

ls = list all files and folders

mnt have all the drives

Extra :

\ Back Slash : Directories

/ Forward Slash : Url

Cucumber:

A cucumber is a tool based on Behavior Driven Development (BDD) framework which is used to write acceptance tests for the web application. It allows automation of functional validation in easily readable and understandable format (like plain English) to Business Analysts, Developers, Testers, etc.

Behavior Driven Development is an extension of Test Driven Development and it is used to test the system rather than testing the particular piece of code

**Feature Files:**Feature files are the essential part of cucumber which is used to write test automation steps or acceptance tests. This can be used as the live document. The steps are the application specification. All the feature files end with .feature extension.

**Sample feature file:**

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

**Scenario**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER” and Password “PASSWORD”  
**Then** login should be successful

**Scenario**: Login Functionality

**Given** user navigates to SOFTWARETETINGHELP.COM  
**When** user logs in using Username as “USER1” and Password “PASSWORD1”  
**Then** error message should be thrown

**Scenario Outline:**

Scenario outlines are used when the same test has to be performed with different data set. Let’s take the same example. We have to test login functionality with multiple different sets of username and password.

Scenario outline basically replaces variable/keywords with the value from the table. Each row in the table is considered to be a scenario.

**Feature**: Login Functionality Feature

In order to ensure Login Functionality works,  
I want to run the cucumber test to verify it is working

**Scenario Outline**: Login Functionality

**Given** user navigates to SOFTWARETESTINGHELP.COM  
**When** user logs in using Username as <**username**> and Password <**password**>  
**Then** login should be successful

| username | password |

| amjed | abc |

| arjun | xyz |

cenario Outline: Eating

Given there are <start> cucumbers

When I eat <eat> cucumbers

Then I should have <left> cucumbers

Examples:

| start | eat | left |

| 12 | 5 | 7 |

| 20 | 5 | 15 |

The Scenario Outline steps provide a template which is never directly run. A Scenario Outline is run once for each row in the Examples section beneath it (except for the first header row).

Runner:

import cucumber.api.junit.Cucumber;

import org.junit.runner.RunWith;

@RunWith(Cucumber.class)

@Cucumber.Options(format={"SimpleHtmlReport:report/smokeTest.html"},tags={"@smokeTest",”@LoginTest”})

Public class JUnitRunner {

}

Step Defenition :

public class LoginTest {

@Given("^user navigates to SOFTWARETETINGHELP.COM$")

public void navigatePage() {

system.out.println(“Cucumber executed Given statement”);

}

@When("^user logs in using Username as \"(.\*)\" and Password \"(.\*)\"$")

public void login(String usename,String password) {

system.out.println(“Username is:”+ usename);

system.out.println(“Password is:”+ password);

}

@When("^click the Submit button$")

public void clickTheSubmitButton() {

system.out.println(“Executing When statement”)

}

@Then("^Home page should be displayed$")

public void validatePage() {

system.out.println(“Executing Then statement”)

}

@Then("^login should be successful$")

public void validateLoginSuccess() {

system.out.println(“Executing 2<sup>nd</sup> Then statement”)

}

}

***What are the keywords used in Feature file?***

* Feature
* Background
* Scenario
* Scenario Outline
* Given
* When
* Then
* And
* But

***Is it mandatory to use Given, When, Then keywords while writing scenario?***

No.  \*  can also be used to write steps in the feature file.

* Given defines the context of the scenario
* When defines the actions of the scenario
* Then defines the outcome of the scenario

***Explain Scenario Outline in feature file***

Scenario Outline keyword in feature file is used to execute scenarios multiple times using a different set of test data. Multiple sets of test data are provided by using ‘Examples’ in a tabular structure separated by pipes (| |)

***Explain background in feature file***

Steps written under the Background keyword are executed before every scenario.

For example: If you want to execute the same steps for every scenario like login to the website, you just write those common steps under the background keyword. While executing every scenario, steps written under background will be executed first.

Feature: Account Balance

  Background:

    Given I login to the account

  Scenario: Verify Positive Balance

    Given I have $100 in my account

    When I withdraw $50

    Then I should have $50 balance

  Scenario: Verify Zero Balance

    Given I have $100 in my account

    When I withdraw $100

    Then I should have $0 balance

C#.NET

* C# doesn't have checked exceptions

The "Common Language Infrastructure" or CLI is a platform on which the .Net programs are executed.

Console.ReadKey();

We then use the Console.ReadKey() method to read any key from the console. By entering this line of code, the program will wait and not exit immediately. The program will wait for the user to enter any key before finally exiting. If you don't include this statement in code, the program will exit as soon as it is run.

Enum:

An enumeration is used in any programming language to define a constant set of values. For example, the days of the week can be defined as an enumeration and used anywhere in the program. In C#, the enumeration is defined with the help of the keyword 'enum'.

class Program

{

enum Days{Sun,Mon,tue,Wed,thu,Fri,Sat};

static void Main(string[] args)

{

Console.Write(Days.Sun);

Console.ReadKey();

}

}

**What is serialization?**

When we want to transport an object through network then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should implement ISerialize Interface. De-serialization is the reverse process of creating an object from a stream of bytes.

**What are sealed classes in C#?**

We create sealed classes when we want to restrict the class to be inherited. Sealed modifier used to prevent derivation from a class. If we forcefully specify a sealed class as base class then a compile-time error occurs.

**What is the difference between Array and Arraylist?**

In an array, we can have items of the same type only. The size of the array is fixed. An arraylist is similar to an array but it doesn't have a fixed size.

What is Jagged Array in C#?

A Jagged array is an array of arrays.

You can initialize a jagged array as −

int[][] scores =newint[2][]{newint[]{92,93,94},newint[]{85,66,87,88}};

Value Type & Reference Type

A data type is a value type if it holds a data value within its own memory space. It means variables of these data types directly contain their values.

For example, consider integer variable int i = 100;The system stores 100 in the memory space allocated for the variable 'i'. 100 is stored at some hypothetical location in the memory (0x239110) for 'i':

Unlike value types, a reference type doesn't store its value directly. Instead, it stores the address where the value is being stored. In other words, a reference type contains a pointer to another memory location that holds the data.

Value type stores the value in its memory space, whereas reference type stores the address of the value where it is stored.

Primitive data types and struct are of the 'Value' type. Class objects, string, array, delegates are reference types.

Value type passes byval by default. Reference type passes byref by default.

Value types and reference types stored in Stack and Heap in the memory depends on the scope of the variable.

Stack is used for static memory allocation and Heap for dynamic memory allocation, both stored in the computer's RAM .

Variables allocated on the stack are stored directly to the memory and access to this memory is very fast, and it's allocation is dealt with when the program is compiled. When a function or a method calls another function which in turns calls another function etc., the execution of all those functions remains suspended until the very last function returns its value. The stack is always reserved in a LIFO order, the most recently reserved block is always the next block to be freed. This makes it really simple to keep track of the stack, freeing a block from the stack is nothing more than adjusting one pointer.

Variables allocated on the heap have their memory allocated at run time and accessing this memory is a bit slower, but the heap size is only limited by the size of virtual memory . Element of the heap have no dependencies with each other and can always be accessed randomly at any time. You can allocate a block at any time and free it at any time. This makes it much more complex to keep track of which parts of the heap are allocated or free at any given time.

You can use the stack if you know exactly how much data you need to allocate before compile time and it is not too big. You can use heap if you don't know exactly how much data you will need at runtime or if you need to allocate a lot of data.

In a multi-threaded situation each thread will have its own completely independent stack but they will share the heap. Stack is thread specific and Heap is application specific. The stack is important to consider in exception handling and thread executions.

This is the temporary memory where variable values are stored when their methods are invoked. After the method is finished, the memory containing those values is cleared to make room for new methods.

When a new method is invoked, a new block of memory will be created in the Stack. This new block will store the temporary values invoked by the method and references to objects stored in the Heap that are being used by the method.

Any values in this block are only accessible by the current method and will not exist once it ends.

When the method ends, that block will be erased. The next method invoked will use that empty block.

This “last in, first out” method makes it easy to find the values needed and allows fast access to those values.

When we write any c-sharp code then cse.exe compiler comes into picture and compiles the code into half-partially compiled code which is called as MSIL code or IL code. Further for full compilation of code CLR comes and takes that half compiled code and does type checking, code verification and pass it to JIT (Just in time compiler). JIT takes that half partially compiled code and as per your machine/server environment it compiles the code into full compilation.

In simple terms the code which is executed by CLR (Common Language Runtime) is called Managed Code, any application which is developed in .Net framework is going to work under CLR, the CLR internally uses the Garbage Collector to clear the unused memory and also used the other functionalities like CTS, CAS etc.

The unmanaged code is basically developed using other languages (other than .Net Framework), so it uses its own language runtime to execute the applications. The application runtime will take care of its memory management, security etc...

Unmanaged code compiles straight to machine code and directly executed by the Operating System

CLS stands for Common Language Specification and it is a subset of CTS. It defines a set of rules and restrictions that every language must follow which runs under .NET framework. The languages which follow these set of rules are said to be CLS Compliant. In simple words, CLS enables cross-language integration or Interoperability.

Common Type System (CTS) describes the datatypes that can be used by managed code. CTS defines how these types are declared, used and managed in the runtime.

The process of Converting a [**Value Type**](https://www.geeksforgeeks.org/c-data-types-2/)**(char, int etc.) to a**[**Reference Type**](https://www.geeksforgeeks.org/c-data-types-2/)**(object)** is called **Boxing**The process of converting[**reference type**](https://www.geeksforgeeks.org/c-data-types-2/)**into the**[**value type**](https://www.geeksforgeeks.org/c-data-types-2/) is known as **Unboxing**

**int** i = 1;    
**object** o = i; // boxing    
**int** j = (**int**)o; /

**Managed code** is a code which is executed by CLR (Common Language Runtime) i.e all application code based on .Net Platform. It is considered as managed because of the .Net framework which internally uses the garbage collector to clear up the unused memory.

**Unmanaged code** is any code that is executed by application runtime of any other framework apart from .Net. The application runtime will take care of memory, security and other performance operations.

**UST Global / Anthem**

Application : COINS  
Framework : BDD with Cucumber.Language JAVA  
Automation technology Selenium  
Build tool : maven  
CI Bamboo  
Unit Testing JUnit

In our Selenium Project Java language used.we are using BDD(Behaviour Driven Developement) Framework by using cucumber and design pattern is Page Object Model.

Maven is the build tool and Jenkins is the CI tool

As per the Page Object Model, Each web page has a separate class and that class holds the functionality and members of that web page.

Folder structure is as below:

Source folder has main and test.

Src/main/java

Src/main/resources

Src/test/java

Src/test/resources

* 1. The **main/java** has packages like Pages, enums,utilities,api,db,io etc
  2. The **main/resources** has properties files, exes, db mapper xmls etc.
  3. **test/java** has the java runner class, and step defenitions.
  4. **test/resources** has feature files, test data, and test results.

There is class named Hooks.java where we have @Before annotation.  
This annotation has a method onSetUp which have the pre-requisites.

DB connection after connecting to hashicorp vault

Pre-requisites gets the Testcase ID which starts with @Com from the gherkin file. Eg: COM-1234

Then we have an @After annotation in which onTearDown method is written.

TearDown gets the screen shots and writes the results to a csv file.

Class CSVWriter is used for generating the test result summary file.

Then Data files/ input files.

There are input data sheet and expected Data Sheet (don’t explain)

Input sheets have policy details like enrollment date, cancellation, agent details etc.

These are loaded into database after converting to a flat file which is called feed file.

Then loaded into DB by FTP

Expected data sheet are read by Excelparser, and data is collected on map/LinkedHashMap.

Then using get and colum header, cell

value is accessed.

Then the tests are added on feature files using gherkin language.  
 For every gherkin steps there will be corresponding step defention on class files.  
 For example for login We have a step as Given Login to COINS UI Application.  
 The login method is added on the step defention.  
 Then we have page class for the login page named LoginPage.java which contains

1. Webelement locators.

2. An overloaded Constructor with Webdriver type as argument.

3. Methods which returns the webelement.

This page has the we element locators using By keyword.  
for example By username=By.id(“user”)  
 By password=By.xpath(“/….”)

WebDriver driver;

Public Loginpage(WebDriver driver){  
 this.driver=driver;

}

Login page gets webdriver instance webdriverUtilities class.   
environment and url details are accessed from a system.properties file.

We start local execution from feature file using right click Run As>Cucumber Feature.

The initial gherkin step is Login to COINS UI Application

The corresponding step Def has a method for the gherkin step

Then Every Web Page has a page class which contains

1. Webelement locators.

2. An overloaded Constructor with Webdriver type as argument.

3. Methods which returns the webelement.

For Example LoginPage.java

Public class LoginPage{

public static WebDriver driver;

By usernameField=By.id(“USER”);

By passwordField=By.id(“PASSWORD”);

By submitButton=By.id(login-btn);

public LoginPage(WebDriver driver){

this.driver=driver;

}

public LoginPage navigateToLoginPage(){

WebDriverUtilities.driver().navigate.to(Environments.valueOf(Configuration.Execution\_Environment).getWebUrl());

WebDriverUtilities.driver().manage().timeouts()…..5 seconds

Return this;

}

Public LoginPage typeUsername(String username){

Driver.findelement(usernameField).sendkeys(username);

Return this;

}

Getting details of execution environment and url:

Execution\_Environment is a static final string which gets value for properties file

All the configuration details are kept on a class Configuration.java

The execution environment, Username password etc are added on a properties file, system.properties.

The properties file have the value mapped from pom.xml by ${ Execution.environment}

Gherkin step:

Given Login to COINS UI Application

Corresponding Step Def: StepDef\_E2E\_web.java

Public class StepDef\_E2E\_web{

LoginPage LoginPage=new LoginPage(WebdriverUtilities.driver());

Given(“^Login to COINS Application$”)  
public void login\_to\_COINS\_UI\_Application() throws Throwable{

LoginPage.navigateToLoginPage();

WebdriverUtlitiles.driver().manage.time…..

LoginPage.typeUsername(Configuration.COINSUI\_USERNAME);

LoginPage.typePassword(Configuration.COINSUI\_PASSWORD);

LoginPage.submitLogin();

}

Deliverables/ Results:

Screenshots   
CSV file

The onTearDown method is added under @After annotation.

Business and Working:

Data Loading:

There are input data added on excel sheets, which are for enrolling the policies, Cancellation, Renrollment, reinstatement, renew, plan change etc.

Data load has been done as

* 1. Updating the cert# alias policy id on the excel sheet. The first 4 digits of the policy will get updated on the sheets before data load.

This has been done using string builder.

* 1. Flat files are created.
  2. Then Data loading is done by executing shellscript

API Testing:

Gets the base url and endpoint url and payload from excel sheet,.

Then

* .

SOAP & REST

SOAP stands for Simple Object Access Protocol whereas REST stands for Representational State Transfer.  
SOAP only works with XML formats whereas REST work with plain text, XML, HTML and JSON.  
SOAP was designed with a specification. It includes a WSDL(Web service definition language) file which has the required information on what the web service does in addition to the location of the web service. JSON stands for JavaScript Object Notation.

**Rest Assuered.**

In order to test REST APIs, I found **REST Assured library** so useful. It is developed by JayWay Company and it is a really powerful catalyzer for automated testing of **REST-services**

public void exampleJsonPathTest() {

  Response res = get("/service/example");

  assertEquals(200, res.getStatusCode());

  String json = res.asString();

  JsonPath jp = new JsonPath(json);

  assertEquals("onur@swtestacademy", jp.get("email"));

  assertEquals("Onur", jp.get("firstName"));

  assertEquals("Baskirt", jp.get("lastName"));

}

Get and post:

Both GET and POST method is used to transfer data from client to server in HTTP  
Main difference between GET and POST method is that GET carries request parameter appended in URL string while POST carries request parameter in message body which makes it more secure way of transferring data from client to server in http protocol.

* In GET method, values are visible in the URL while in POST method, values are NOT visible in the URL.
* GET has a limitation on the length of the values, generally 255 characters whereas POST has no limitation on the length of the values since they are submitted via the body of HTTP.
* GET method supports only string data types while POST method supports different data types, such as string, numeric, binary, etc.
* GET request is often cacheable while POST request is hardly cacheable.

GET performs are better compared to POST

GET, POST, PUT, PATCH, and DELETE are the five most common HTTP methods for retrieving from and sending data to a server.

The GET method is used to retrieve data from the server. This is a read-only method, so it has no risk of mutating or corrupting the data. For example, if we call the get method on our API, we’ll get back a list of all to-dos.

The POST method sends data to the server and creates a new resource. The resource it creates is subordinate to some other parent resource. When a new resource is POSTed to the parent, the API service will automatically associate the new resource by assigning it an ID (new resource URI). In short, this method is used to create a new data entry.

The PUT method is most often used to update an existing resource. If you want to update a specific resource (which comes with a specific URI), you can call the PUT method to that resource URI with the request body containing the complete new version of the resource you are trying to update.

The PATCH method is very similar to the PUT method because it also modifies an existing resource. The difference is that for the PUT method, the request body contains the complete new version, whereas for the PATCH method, the request body only needs to contain the specific changes to the resource, specifically a set of instructions describing how that resource should be changed, and the API service will create a new version according to that instruction.

The DELETE request simply looks like this, for deleting a specific resource:

Maven: 3.6.3

The sequence of steps which is defined in order to execute the tasks and goals of any maven project is known as build life cycle in maven.

There are three built-in build lifecycles:

1. **default** lifecycle handles project deployment.
2. **clean** lifecycle handles project cleaning.
3. **site** lifecycle handles the creation of project's site documentation
4. **Default** Maven Life Cycle is the general model in build process, which builds, tests and distributes the project. There are 21 phases in the default life cycle.

|  |  |
| --- | --- |
| compile | Source code compilation |

|  |  |
| --- | --- |
| test-compile | Compiling the test source code to test destination. |

|  |  |
| --- | --- |
| test | With unit testing framework, one can run the tests. |
| verify | Checking the packages are valid or not. |
| install | Installin  g the package into local repository. |
| deploy | Copying the package into remote repository. |

1. Clean Lifecycle is the simple life cycle in Maven that has 3 phases. All the three are invoked one by one when clean life cycle command **mvn clean** is executed. Clean is the important phase of Clean Lifecycle that has a goal **clean:clean** to delete the output of a build.
2. Pre-clean
3. Clean
4. Post-clean
5. Site Life Cycle helps in project documentation and reporting process. There are four phases in this life cycle.
6. pre-site
7. site
8. post-site
9. site-deploy

DOCKER:

Docker is a container.

A company needs to develop a Java Application. In order to do so the developer will setup an environment with tomcat server installed in it. Once the application is developed, it needs to be tested by the tester. Now the tester will again set up tomcat environment from the scratch to test the application. Once the application testing is done, it will be deployed on the production server. Again the production needs an environment with tomcat installed on it, so that it can host the Java application. If you see the same tomcat environment setup is done thrice. There are some issues that I have listed below with this approach:

1) There is a loss of time and effort.

2) There could be a version mismatch in different setups i.e. the developer & tester may have installed tomcat 7, however the system admin installed tomcat 9 on the production server.

Dev develeops and the code dosent work on the QA system due to the difference in the environments. Solution is virtual machine.

Docker is more light weight and provide same functionality as virtual machines.

Less space, fast and performance, high efficiency.

Docker is an [open source](https://whatis.techtarget.com/definition/open-source) software platform to create, deploy and manage virtualized application containers on a common operating system

Kubernetes:

Container Management Tool

Container deployment, Container (de)Scaling, Container Loadbalancing

Handle a large volume of containers and users, simultaneously. An application may have thousands of containers and users interacting with each other at the same time; managing and keeping track of these interactions requires a comprehensive overall system designed specifically for that purpose.

**About Myself:**  
I have around 9 years of experience in Software Industry and I have been working as a Software QA for last 7 years. I have experience on both manual and Automation testing.  
(In all these years I have worked on different domains like finance, Insurance, credit reporting etc.)  
The tools and technologies which I have experience includes  
Java, Selenium web driver, TestNG, Maven, BDD cucumber, Bamboo, Jenkins, SQL, Git etc  
Then the frameworks were mostly Data driven using TestNG and BDD with cucumber.

I got a chance to be a part of some giant organizations like RouteOne, Anthem, Equifax etc

Major share of my experience was from Routeone, which is an Automobile finance organization located in Michigan. I was a member of their offshore team at Bangalore.  
My responsibilities at RouteOne were  
Daily build validations on lower environment(qa & uat) by triggering selenium scripts.  
Routeone had releases every month and I have to work on features assigned to me. These features has to be manually tested and then it has to be automated for future regression purpose.  
Regression execution on Staging environment  
Release validations

Now I am working for the client Equifax, from UST Global.

Currently the Equifax is migrating DB, in which legacy is SQL server which is migrated to Google Cloud

My current roles and reposnbilities include  
1. I am leading the QA team from the last 6 months, and reporting directly to Equifax QA leadership.  
2. I am working as an individual contributer also, working on both manual and automation testing.  
3. Attending the Agile meetings.  
4. Other meetings like weekly One to One with Equifax QA leadership,   
 Weekly meeting Equifax project manager along with dev team.  
4. Bi weekly meetings with Equifax QA leads.  
5. Conducting interviews for UST Global.   
6. Demoing the progress to the leadership.

I have to create manual test cases, automate the same, develop automation functionalities etc.  
Then creating RTM, scorecard etc.

**UST / EQUIFAX Inc**

E1AF framework Equifax.

Equifax have a framework called E1AF, which is a BDD cucumber f/w.

**MDS Datafabric Migration**

The DBmigration includes a legacy SQL which is migrated to google cloud.  
**An extraction utlity is developed to extract the data from MSSQL DB and converted to avro files.  
Different tables involved are Acc, Address, Location, attr\_Config, Phone, Gender, Ethnicity, Industry Codes, Location, Contact, Link, Certification etc.**

**The data on avro are ordered based on 3 match key.   
 ATKey/ Entity Key  
provider ID / Parent ID  
Date Reported**  
For that the an extraction utlity will extract the data joining the 16 tables and will convert to avro files. Then these avro files are injested to journaling for which some rules are applied and moved to purposing.   
All these are based on a key called AT key or entity key.

Extraction utility automation

Automation Framwerok developed using Java and Cucumber.  
The Hooks file have @Before annotation.  
The DataBase credentials are kept on Hashicorp Vault.  
@Before will invoke the class to connect to Hashicorp vault using springboot and create the connection url.

The Test Data conatins AT Key and Source Key.  
The Data is used to construct queries. This has been done with a class named SQLQueries.  
This class contains methods to construct queries for different tables.  
The keys are passed to the SQL Queries class and queries are constructed.  
Then the queries are executed across the DB and Resultset is returned.  
The avro files which need to be tested are kept under test/resources/avro\_files  
Then Compare step will parse the avro inside the result set looping and compares between tables and avro files.

After execution the results are written on a CSV file.  
A dashboard integration is also done and results can be viewd on a dashboard. This happens when the script is executed through Jenkins.

Scenario Outline:

Given <TestID> ATKey and SourceKey as Test Data  
When Construct “Acc” table query with <ATKey> and <SourceKey>  
When Execute query across Database  
Then Compare Acc Table and avro file

Examples:

|TestID | ATKey | SourceKey

|MDS\_EU\_001|12|1726|

Now I am working on developing a BDD framework to compare the api response from legacy and purposed end points.  
feature file has been created with scenario outline which will send requests for multiple AT Keys

The json payload template has been stored in the test data folder.  
Then the test data sheet have the atkey values.  
This key is used to generate the json payload using the teplated and save inside the request folder.  
Rest Assured has been used for a post call to get the response from the end point. Then file writer has been used to write the reposne to a json file.

The same has been done for the purposing end point also.

Now I am working on developing a class to compare these two responses.

Cucumber background