Java - Notes

What is Java ?

- Java is a simple and most widely used programming language.
- Java is fast, reliable and secure.
- **default Package in java :** java.lang
- **Why we go for java :** Platform independent, opensource & multi-threading
- It runs multiple application at a time.
- **Components of Java: JDK, JVM, JRE**

3 packages :

1. Java.lang – package of predefined
2. Java.util – Collections
3. Java.io - FIles

What are the main features of java?

1. Platform independent
2. Open source
3. Multithreading
4. More secure
5. Portable

What is platform independent?

- Compiler converts your source code to byte code that's why java is platform independent.
- During the compilation the java program is converted into byte code(not machine specific).
- Using this bytecode it can be runned by jvm of any platform.

What is meant by Open Source?

A program in which source code is available to the general public for use and/or modification from its original design at free of cost is called open source.

What are IDE/tools availabe in market for java?

Notepad , Netbeans , Eclipse , JDeveloper(oracle) , RAD(IBM)

OOPS Concepts definitions:

1.CLASS:

Class is the collection of objects and methods.
Class contains attributes(variables and methods) that are common to all the objects created in a class.
In our project We are using lots of predefined classes like
1.ChromeDriver
2.FireFoxDriver
3.InternetExplorerDriver
4.Actions
5.Robot
6.Select
7.RemoteWebDriver

2.OBJECT: (Object is the super class of all java)
Object is an instance of a class. Using object,we can able to access the methods in a class.
It is a runtime memory allocation.

Object creation syntax:

Classname objectname=new Classname();

3.METHODS:
Set of actions to be performed. Reusable lines of code can be kept inside a method.
We can access the method using object whenever needed.
How to access class properties using object:
object.variable; , object.method();
In real time I'm using lots of methods like :
get(), getTitle(), getCurrentUrl(), getText(), getAttribute()

Pillars (or) Principles of Java:

1. Inheritance
2. Abstraction
3. Polymorphism
4. Encapsulation

1.INHERITANCE:

→ Using inheritance we can access one class properties in another class without multiple object creation. So we can save memory. Also we can achieve code reusability.
➔ In real time, I m using inheritance concept in base class where i will maintain all reusable methods. So, by extending base class i can call reusable methods wherever i want.

Single Inheritance:
One child class extends one parent class.
Multilevel Inheritance:
One child class extends more than one parent class in tree level structure.
Multiple Inheritance:
More than one parent class parallelly supporting into one child class. We can't achieve multiple inheritance in java with extends due to
1.Syntax error/compilation error(After extends keyword we cannot specify more than one class)
2.Priority problem(When parent classes having same method name with same arguments) extending into same child class,
We can achieve multiple inheritance in interface using implements keyword.
HYBRID INHERITANCE:
It is the combination of hierarchical and multiple inheritance.
HIERARCHIAL INHERITACE:
More than one child class inheriting the same parent class.

2.ENCAPSULATION: (Also called as Data Hiding)
-->Wrapping up of data and code acting on data together in a single unit.
-->Structure of creating folders.
-->Encapsulation, is to make sure that "sensitive" data is hidden from users.
-->To achieve this, you must: declare class variables/attributes as private
-->Provide public getters and setters methods to access and update the value of a private variable.

I am using encapsulation in real time in POJO class where I create all my locator variables as private. Because private variables cannot be accessed outside the class. If we want to access private variables we have to use getters.

3.POLYMORPHISM:
Performing single action in different ways or completing task in different ways.

Method Overloading/Static Binding/CompileTime Polymorphism:
--> In same class, there are multiple functions with same name but different parameters then these functions are said to be overloaded.
--> Same method overloaded again and again using different arguments type, count and order.

For example: In realtime I'm overloading println, sendkeys and frame methods

```
System.out.println("Renu");
System.out.println(123);
System.out.println(12.2);

webElement.sendkeys("Renu");          //Here Sendkeys accepts only one argument

Actions a=new actions(driver);
a.sendkeys(webelement,"Renu");     //using actions class also we can use sendkeys, here i am passing two arguments

driver.switchTo.frame(String id);
driver.switchTo.frame(String name);
driver.switchTo.frame(int index);
driver.switchTo.frame(Webelement ); //For same methods i m passing String type,int type and Webelement type
```

Method overriding/Dynamic Binding/RunTime Polymorphism:
-->Method Overriding is when one of the methods in the super class is redefined in the sub-class.
(or)When I am not satisfied with the business logic of my super class i am overriding that method in my subclass.
In Realtime I m overriding many methods such as retry() method from IRetryAnalyser, transform() method from IAnnotationTransformer and getMessage() method from Exception.

Notes:
-->A constructor cannot be overridden.
-->Final - declared methods cannot be overridden
-->Any method that is static cannot be used to override.

5. DATA ABSTRACTION:
==========
Hiding the implementation of the program.
(or)
Process of hiding certain details and showing only essential information to the user.

Abstraction can be achieved with either abstract classes or interfaces.
The abstract keyword is a non-access modifier, used for classes and methods:

1.Abstract class:

- Class which is declared as abstract is known as abstract class.
- Contains abstract as well as non-abstract methods.
- We can't create objects for abstract class(to access the methods in abstract class, it must be inherited in another class).
- Abstract method: can only be used in an abstract class, and it does not have a body(business logic). The body is provided by the subclass which extends the abstract class.
- Non-abstract method(Concrete method):normal method with business logic.
- It can have constructor and static method also.
- In real time, I m using 'By' in my project which is an abstract class.

```
public abstract class Sample {
	public abstract void clientId();  //abstract method which has no logic
	public void clientName() {
		System.out.println("Client name is CTS");
					//non-abstract method with business logic
	}
}
```

Why And When To Use Abstract Classes and Methods?
To achieve security - hide certain details and only show the important details of an object.

2.Java Interface: (Also called as Fully abstraction)

- An interface in Java is a blueprint of a class. It contains only abstract methods. We cannot abel to create object.
- Interface default value is null.
- The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods(no business logic) in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java.
- In interface "public abstract" is default.
- Using "implements" keyword we can implement the interface in a class where we can write the business logic for all unimplemented methods.
- To access the interface methods, the interface must be "implemented" (kind of "inherited") by another class with the implements keyword (instead of "extends").
- I am using many interfaces in my project such as Webdriver, webelement, Alert, Wait, Javascript executor, Takesscreenshot, List, Set, Map etc

Notes on Interfaces:
-->Interface methods do not have a body - the body is provided by the "implement" class
-->On implementation of an interface, you must override all of its methods
-->Interface methods are by default abstract and public
-->Interface attributes(variables) are by default public, static and final
-->An interface cannot contain a constructor (as it cannot be used to create objects)

Why And When To Use Interfaces?
1) To achieve security - hide certain details and only show the important details of an object (interface).
2) Java does not support "multiple inheritance" (a class can only inherit from one superclass). However, it can be achieved with interfaces, class can implement multiple interfaces.

Advantages of Abstraction:
It reduces the complexity of viewing the things.
Avoids code duplication and increases reusability.

Difference between Abstraction and Encapsulation:
Encapsulation is data hiding(information hiding) while Abstraction is detail hiding(implementation hiding)
While encapsulation groups together data and methods that act upon the data, data abstraction deals with exposing the interface to the user and hiding the details of implementation.

STRING:

What is String?

- String is a sequence of characters that's stored in a character Array. It is a text that is enclosed in double quotes.
- String is a non- primitive data type, index based class in Java.
- It is a predefined class & it is present in java.lang package
- It stored data based on index position.
- Index starts from 0 to n-1
- Substring() & indexof() supports method overloading.

String is immutable in Java, why?

- String is immutable in Java because whenever we declare any string it will be stored in the string pool constant where we can't change values in that particular memory.
  For eg; string s1 = "hello"
  String s2= "hello"
- String will identify the values are similar and will not create separate object for each values. However, if we concat the string s1.concat("java"), the reference variable s1 will still have value "hello" instead of "hello java".

What is String Constant pool?

- String Constant Pool is the memory space in heap memory specially allocated to store the string objects created using string literals. In String Constant Pool, there will be no two string objects having the same content/value.
- Whenever you create a string object using string literal, JVM first checks the content of the object to be created. If an object in the string constant pool has same content, then it returns the same reference of that object. It doesn't create a new object. If the content is different from the existing objects then only it creates new object.
- In Java, String is a Class, strings are treated as objects.

String Literal:
String str = "Java";
->Literal string shares the same memory incase of duplicates.
->It is stored in String pool constants(String pool constants present inside heap memory)
->When String declared like this, intern() method on String is called. This method references internal pool of string objects. If there already exists a string value "Java", then str will get reference of that string and no new String object will be created.

String Object(Non-Literal String)
String str = new String("Java");
->Non-literal string will not share the same memory even if it is a duplicate.

->Since we use "new" operator everytime it creates new memory.
->Non-literal strings stored in heap memory.
In this, JVM is forced to create a new string reference, even if “Java” is in the reference pool.

Performance Comparison between String literal and String constant:

- If we compare performance, string object will always take more time to execute than string literal because it will construct a new string every time it is executed.

When there is a string class why we need to go for string buffer and string builder?

The objects of String class are immutable in nature, i.e. you can’t modify them once they are created. If you try to modify them, a new object will be created with modified content. This may cause memory and performance issues if you are performing lots of string modifications in your code. To overcome these issues, we use StringBuffer and StringBuilder classes.

Immutable String:

- Immutable string once string created, value can't be changed in reference.

Eg: String literals

Mutable String:

- For mutable string, we can change the value in reference.

Eg: StringBuffer, StringBuilder

String Buffer	**String Builder**
It is synchronized i.e. only one thread can enter an object at any point of time.	It is asynchronised i.e multiple threads can enter the object at any point of time.
It is Thread safe. Multiple threads can’t enter the object simultaneously. One Thread has to wait until another thread has finished with them.	It is not Thread safe i.e. multiple thread can enter the object parallel. Threads don’t have to wait until one thread is finished.
Because of the above reason String Buffer is slower	String builder is faster as a result of this.

- **identityHashCode() -method to find the address of the variable.**

Difference between Remove all() and Retain all

removeAll():

- removeAll() is a method , it is used to compare the 2 lists and remove all the common values

retainAll():

- retainAll() is a method, it is used to compare both lists and retains only the common values

What is meant by array?

- Storing multiple values of similar datatype in a single variable.
- In Array, the elements can be accessed randomly using the index number.
- It stores values based on index

What are the advantages and disadvantages of array?

Advantage:
In a single variable we can store multiple values.
Easier to access data using the index number and easier to manipulate the data

Disadvantages:
It support only similar data types.
Size fixed at compile time.
Memory wastage is high.

Can we change the memory size of array after initialization?
No,we can't change the memory size of array after initialization.

Difference between For loop & Enchance for loop:

For-Loop:

1. It is based on index
2. We can apply conditions
3. Possibility of occurring IndexOutOfBound exception
4. We can able to do backward / forward
5. We can able to get particular value

For-Enchanced:

1. It is based on value
2. We cannot able to apply conditions
3. We cannot able to do backward / forward
4. We cannot able to get particular value

What is Collection ?

- Collection is an interface in java.util package
- Collection Framework is a combination of classes and interface, which is used to store and manipulate the data in the form of objects. It provides various classes such as ArrayList, Vector, Stack, and HashSet, etc. and interfaces such as List, Queue, Set, etc. for this purpose.
- It will support storage of multiple values with dissimilar data types.
- It is dynamic memory allocation.
- No memory wastage like array.
- Iterable is the super class of collection

Difference between Collection and Collections

- Collections is an utility class in java.util package
- Collection-Collection is an interface under which we have list,set,queue
- Collections have lots of predefined methods which we can apply over collection objects.
- Collections class sorts and synchronise the collection elements
  Eg: Collections.min(), Collections.max(), Collections.sort()

What is Generics?

- **Java Generics <>** is a set of related methods or a set of similar types. Generics allow datatypes Integer, String, or even user-defined types to be passed as a parameter to classes, methods, or interfaces. Generics are mostly used by classes like HashSet or HashMap. Generics was added in jdk 1.5 version.

Advantages of Generics:

1. Code re-use
2. Compile time safety – invalid datatypes can be identified during compile time itself
3. Individual type-casting is not required

In Collections two interfaces are there :

- List
- Set

List : (Present in java.util package)

1. List Interface is index based and it allows duplicate elements
2. List maintains insertion order
3. We can add any number of NULL values
4. List implementation classes are **Array List, Linked List and Vector**
5. List has get() method to get the element at specific index
6. If we want to access the elements frequently by using Index, we can go for list

ArrayList	**Vector**
ArrayList is not synchronized i.e. executed parallel	Vector is Synchronous (executed one by one)
Not Thread safe	Vector is thread safe
Arraylist is not a legacy class	Vector is a legacy class
ArrayList increases its size by 50% of the Array size	Vector increase its size by doubling the Array size

What is the difference between ArrayList and LinkedList?

LinkedList:

Insertion and deletion is a best one.
Searching/retrieving is a worst.
It makes performance issue.

ArrayList:

In Arraylist retrieve/searching is a best one
In ArrayList deletion and insertion is a worst one because if we delete/insert one index value after all the index move to forward/backward.
It makes performance issue.

How to convert List into Set?

By addAll() we can convert List into set.(all the elements in list will get added to set)

Set :

1. Set interface is value based and it doesn't allow duplicate elements
2. Set doesn't maintain insertion order
3. Only one null value is allowed in Set
4. Set implementation classes are **Hashset, Linked Hashset and Treeset**
5. Methods not supported in set are indexof, lastIndexof, get, set
6. If we want to create a collection of unique elements, we can go for set

What are the methods available in list But not in set?
indexOf(); , get(); , lastIndexOf();

Difference between add(index,element) and set(index,element)?
Add(index,element) – This method is used to insert the specified element in the specified position in the list
Set(index, element) – This method is used to replace the element in a specified position with a specified element.

Hash Set:

1. Hash set is implemented using HashTable
2. HashSet allows a null object
3. Hash set use equals method to compare two objects
4. Hash set doesn't now allow a heterogeneous object
5. HashSet does not maintain any order

Tree Set :

1. The tree set is implemented using a tree structure.
2. The tree set does not allow the null object. It throws the null pointer exception.
3. Tree set use compare method for comparing two objects.
4. Tree set allows a heterogeneous object
5. TreeSet maintains an object in sorted order

Map: (Present in java.util package**)**

1. Map interface has key value pair combination where key don't allow duplicate elements but values allow duplicates
2. Map doesn't maintain insertion order
3. Map allows single Null key at most and n number of null values
4. Map implementation classes are **HashMap, Linked HashMap, TreeMap, HashTable and Concurrent HashMap**
5. If we want to store the data in the key value pair combination, we can go for Map.

How much null allows in below maps:

HashMap :	k-1 null,v- n null
LinkedHashMap:	k-1 null,v- n null
TreeMap :	k-ignore null,v- allow null
HashTable :	k-ignore null,v- ignore null

How to Iterate Map?
1. Iterating over entries using For-Each loop.
2. Iterating over keys or values using keySet() and values() **method** using for-each loop.

3.Iterating using stream() in **JAVA** 8.
4.Using entrySet()
5. Using **Iterator** through a **Maps**.

What is the return type of entrySet?
Set<Entry<key,value>>

Write the methods to get the key only and value only?
For key only keySet() method is used.
For value only values() method is used.

What is the Return type of entryset(), keyset(),Values(),getKey(), getValue()?

Entryset() – set<Entry<>>
Keyset() – set
Values () – collection
getKey() – based on datatype
getValue() – based on the datatype

Hash Map:

1. HashMap is **non synchronized**. It is not-thread safe and can't be shared between many threads without proper synchronization code.
2. HashMap **allows one null key and multiple null values**.
3. HashMap is a **new class introduced in JDK 1.2**.
4. HashMap is **fast**.
5. We can make the HashMap as synchronized by calling this code Map m = Collections.synchronizedMap(hashMap);
6. HashMap is **traversed by Iterator**.
7. Iterator in HashMap is **fail-fast**.
8. HashMap inherits **AbstractMap** class.

Concurrent Hash Map:

1. Introduced in 1.2 version
2. Best in performance
3. It is a legacy class (old)
4. Same as Hash Table
5. Random order

Hash Table:

1. Hashtable is **synchronized**. It is thread-safe and can be shared with many threads.
2. Hashtable **doesn't allow any null key or value**.
3. Hashtable is a **legacy class**
4. Hashtable is **slow**.
5. Hashtable is internally synchronized and can't be unsynchronized.
6. Hashtable is **traversed by Enumerator and Iterator**.
7. Enumerator in Hashtable is **not fail-fast**.
8. Hashtable inherits **Dictionary** class.

CONSTRUCTOR: (Purpose to initialize the variable)

- Whenever you create object for a class, default constructor will automatically invoked since class name and constructor name are same.

Purpose of constructor:

-->A constructor in Java is a special method that is used to initialize objects.
-->The constructor is called when an object of a class is created.
-->All classes have constructors by default: if you do not create a class constructor yourself, Java creates one

Rules:

-->constructor name must match the class name.
-->It cannot have a return type (like void)
--> A java constructor cannot be abstract, static, final and synchronized.

Explain the types of constructor?

Parameterized constructor - A parameterized constructor is where we can pass arguments/parameters.

Non parameterized constructor - Also called Default constructor. This is present as a default in every class where it doesn't have any parameters and has a default value of null, 0 when an object is invoked.

Constructor chaining:

- The process of calling one constructor from another constructor with respect to current object creation is called constructor chaining.
- **Keywords : this(), super()**
- this() – used to call the current class constructor
- super() – used to call the parent class constructor
- Rules : should be present at the first line of constructor.

Do constructors have any return type?

No, constructor can't have any return type.

Why a return type is not allowed for constructor?

constructor is not directly called by your code, its called by memory allocation and object initialisation in the run time.
Its return value is opaque to the user so we can’t mention it.

Can we declare constructor as 'private'?

Yes,we can declare constructor as private.

Why a compiler given constructor is called as default constructor?

If we didnt create a constructor explictly it will take the default constructor.

Access Modifiers:

It is used to set the access level for classes, attributes, methods and constructors.
We divide modifiers into two groups:

Access Modifiers - controls the access level
Non-Access Modifiers - do not control access level, but provides other functionality

Access specifiers:

- private
- default or no modifier
- protected
- public

Public(Project level access/Global level access)
In same package as well as different package, we can access the private variables and methods of one class in another class using extends keyword and creating separate object.

Protected(Global level access)
In same package, Using extends keyword and by creating separate object also we can access the properties of another class.
If it is in different package, using extends keyword only we can access, We cannot create object and access the protected attributes of a class.

Default(Package level access)
In same package only we can access(In different package we cannot access) using extends keyword and by creating object.

Private(Class level access)
Outside class, we cannot access private variables. If we want to access private variables outside the class we use getters and setters.

Non Access Modifiers:

- Static
- Final
- Abstract

1..Abstract
Class level : If we declare class as abstract, we can't create object
Method level : If we declare method as abstract, we can't write business logic
Variable level : We cannot able to declare variable as abstract.

2..Static
Class level : We cannot able to declare class as static.
Method level :
In Same Class : When a method is declared as static, we need not create object to call the particular method. Static method in java belong to the class(not to an object).
In different Class : Using extends keyword we can directly the method, without using the extends keyword we can call as Classname.methodname()
Variable level :
In Same Class : When a variable is declared as static, we need not create object to call the particular variable.
In different Class : Using extends keyword we can directly the variable, without using the extends keyword we can call as Classname.variablename()

3..Final
Class level : If we declare class as final, we can't create inherit
Method level : If we declare method as final, we can't override
Variable level : If we declare variable as final, we can't change the value

What is mean by static keyword in java?
The static keyword is mainly used for memory management.
It is used to share the same variable or method by objects of given class.

Can we override static method in java?
No,we can't override the static method because it is part of a class rather than an object.

Can we overload static method in java?
Yes, we can overload the static method in java.

TYPES OF VARIABLES:

LOCAL VARIABLE:

- It is declare inside the method block or constructor block
- It gets activated when the control enters to the method
- It gets de-activated when the control enters to the method
- We can’t declare any access specifier
- We need to initialise the value it doesn’t take the default value automatically
- It stored in Stack memory (temp memory)

INSTANCE VARIABLE: (Object Level)

- It is declare inside the class and outside the method
- It gets activated when the object is created
- It gets de-activated when the object is destroyed
- We declare any access specifier
- We no need to initialise the value it take the default value automatically
- It stored in heap memory

GLOBAL VARIABLE: (Class Level)

- It is declare inside the class and outside the method
- It gets activated when the control enters to the class
- It gets de-activated when the control exits to the class
- We declare any access specifier
- We no need to initialise the value it take the default value automatically
- It stored in static memory

What is Stack Memory?

Stack in java is a section of memory which contains methods, local variables, and reference variables. Stack memory is always referenced in Last-In-First-Out order. Local variables are created in the stack. Stack memory structure is mostly implemented for providing static memory allocation.

What is Heap Memory?

Heap is a section of memory which contains Objects and may also contain reference variables. Instance variables are created in the heap Program Counter Register: Programs are a set of instructions. Program counter register holds the address of the upcoming instructions to be executed.

What is an Exception?

Exception is an unexpected event which when occurs in a program, your program will terminate abnormally. We can avoid this abnormal termination using exception handling mechanisms(try,catch,finally,throw,throws)

2 Types:

1. Unchecked
2. Checked

1**.Unchecked Exception / run time exception**

Whenever the exception will occur in runtime

1. Arithmetic Exception
2. Null pointer Exception
3. Input mismatch Exception
4. Array index out of bound Exception
5. String index out of bound Exception
6. Number format Exception

2.**Checked Exception / compile time**

Whenever the exception will occur in compile time

1. File not found Exception
2. IO Exception
3. SQL Exception
4. Class not found Exception

What is the super class for Exception and Error?

Throwable - super class - Exception
Exception - super class - Error

Can we have try block without catch block?

Yes we can have try block without catch block.But in that case finally block must be present.(There will be no syntax error)
Possible but we will not able to handle the exception without catch block.

Exception:

Exception results in abnormal termination of the program (or)Whenever exception occurs, program terminates at the particular line itself.

We can avoid the abnormal termination of the program by handling the exception.

Object
|
Throwable
|
Exception

Exception handling:

try block:

The code which is expected to throw exceptions is placed in the try block.

Catch block:

When an exception occurs, that exception occurred is handled by catch block associated with it.
Every try block should be immediately followed either by a catch block or finally block.

finally block:
The finally block follows a try block or a catch block.
A finally block of code will always be executed whether exception occurs or not.

Possible ways of Exception handling:

- Try, finally
- Try, catch, finally
- Try, multiple catch

Impossible combination:
-->Throwable class is the super class of all exception.
-->This will handle all the exceptions.so all the child catch blocks will not be used/reached by any code.
-->so while writing the program itself,it will show error.

```
try
{
}
Catch(Throwable e){
}
Catch(Exception e){
}
Catch(ArithmeticException e){
}
```

possible combination:
-->we can specify multiple catch blocks

```
try {
}
Catch(ArithmeticException e){
}
Catch(IndexOutOfBoundException e){
}
Catch(Throwable e){
}
```

What are the differences between final finally and finalize in java?

Final:
A final class variable whose value cannot be changed.
A final method is declared in class level, they cannot be inherited.
A class declared as final can't be inherited.

Finally:
-It's a block of statement that definitely executes after the try catch block.
Exception occurs or not, finally block always get executed.

Finalize:
It is method of object class
It is a method that the Garbage collector always calls just before the deletion / destroying the object which is eligible for garbage collection.

Throw	Throws
Throw is a keyword used explicitly to throw an exception	Throws keyword is used to declare an exception
Checked Exceptions can not be propagated using Throw only	Checked Exceptions can be propagated using Throws
Throw is followed by an instance	Throws is followed by a class
Throw is used within the method	Throws is used in the method signature
We can only throw one exception	Multiple exceptions can be declared using throws Eg: Public void method()throws IO exception,SQL Exception

Errors :
These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because we cannot handle the error.
For example, JVM crash, stack overflow, insufficient memory. They are also ignored at the time of compilation.

Note:
-->A catch block cannot exist without a try statement.
-->It is not compulsory to have finally block whenever a try/catch block is present.
-->The try block cannot be present without either catch block or finally block.
-->No code can be present in between the try, catch, finally blocks.
-->We can have 'n' number of exception throwing statements in try block, once an exception caught control will be moved to catch block, all the remaining lines won't be executed.

POM framework:
Page Object Model or Pattern Object Model is a framework or design pattern mainly used to maintain the locators. In realtime – for every time of your application there will be separate POJO(Plain Old Java Object) -- we can achieve encapsulation concept (data security or data hiding)
In POM Framework,
POJO class can contain :

1. Default constructor (PageFactory.initElements)
2. Private WebElements
3. Getters to access those webElements outside the class

Types:

1. With pageFactory (StaleElementReferenceException)
2. Without pageFactory

Annotations in POM:

@FindBy----------To find a WebElement---------only one criteria/condition will be given to fetch a WebElement
@FindBys----------Multiple criterias will be given for finding single webelement-----if it is satisfying all the conditions,then only webelement will be fetched(acts like AND operator)
@FindAll----------Multiple criterias will be given for finding single webelement-----if it is satisfying any one of the given conditions,then webelement will be fetched(acts like OR operator)
@CacheLookUp---------To maintain a cache for the webElement----To increase the performance

2.without pagefactory

Annotations in POM:

@FindBy----------To find a WebElement---------only one criteria/condition will be given to fetch a WebElement
@FindBys----------Multiple criterias will be given for finding single webelement-----if it is satisfying
all the conditions,then only webelement will be fetched(acts like AND operator)
@FindAll----------Multiple criterias will be given for finding single webelement-----if it is satisfying
any one of the given conditions,then webelement will be fetched(acts like OR operator)
@CacheLookUp---------To maintain a cache for the webElement----To increase the performance

Advantages of POM:

1.It is used to maintain locators separately
2.We can avoid lots of reworks
3.We can increase the performance.
4.We can avoid StaleElementReferenceException and we can reuse the WebElements

StaleElementReferenceException:

When your page is getting refreshed or if you are moving into new window,
all the references(variables) we have found for WebElement will become stale(old).If you try to reuse the webelements you will get StaleElementReferenceException so we cant reuse the locators again and again with same reference.
But we can avoid this exception using PageFactory concept in POM.

PageFactory is a class and initElements() is a static method

PageFactory class will capture bulk of WebElements and when initElements() method is called it re-initialize all your WebElements from WebDriver.so Whenever you want to reuse your locators----You can create Object for the particular POJO class and it will invoke your default constructor automatically----so PageFactory.initElements() will get called.

findElement	findElements
==========	==========
To find single WebElement	To find more than one WebElement
Unique xpath will be given	can give common xpath
NoSuchElementException	When thr is no matching WebElement,it will return an Empty list
	li.size()==0(thr are no matching webElements)

What is Upcasting?
Upcasting is the process of accessing the method of parent class by creating an object for the child class. It is the typecasting of a child object to a parent object. Upcasting can be done implicitly.
What is downcasting?

Downcasting is the process of accessing the method of the child class by creating an object for the parent class. It is the typecasting of a parent object to a child object. Downcasting can not be implicit.

What is wrapper class?

A Wrapper class is a class whose objects wraps or contains primitive data types. It provides the mechanism for an object to be converted to a primitive datatype(Unboxing) or a primitive datatype to an object(autoboxing).
Eg: Byte, Integer, Short,Long, Float, Double,Character,Boolean

What is mean by File? In which package it is available?
File is a class and it is used to achieve the file operation.
It is available in java.io package.

What are the differences between append and updating the file?

For updating the file:
It will replace the old contents of the file.

For appending the file:
It will add the contents at the end of the file.

What is meant by Enumerator, Iterator and List Iterator?

Enumeration:
It is an interface used to iterate only legacy class or interface.
Only iterates in forward direction

Iterator:
It is an interface used to iterate the collection objects
Only iterates in forward direction

List Iterator:
It is an interface used for iterating list type classes
iterates in forward as well as backward direction

What are the methods available in Enumerator, Iterator and List Iterator?

Enumerator Methods:
hasMoreElements();
nextElement();

Iterator Methods:
hasNext();
next();
remove();

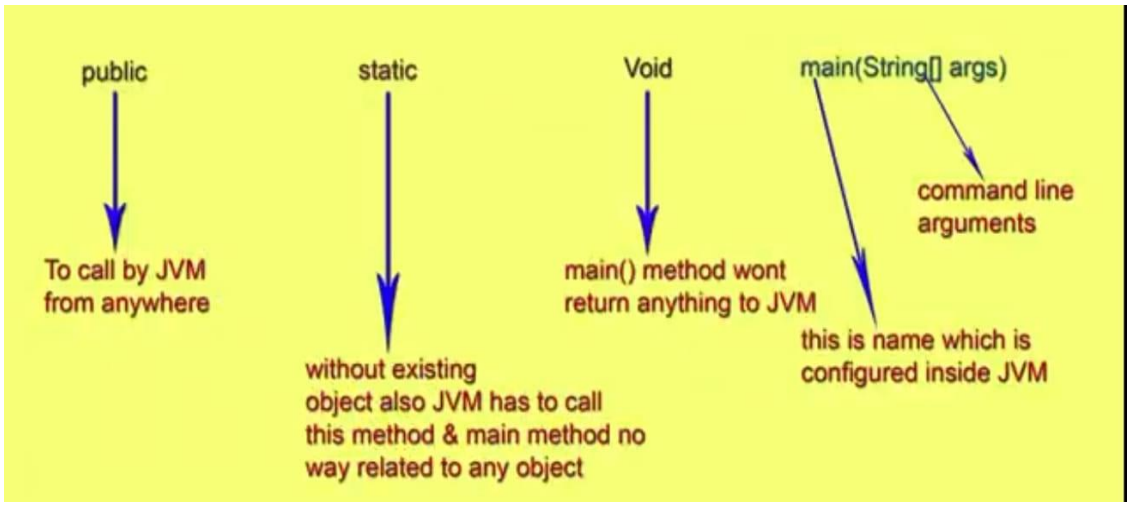
ListIterator Methods:
next();
remove();
hasPrevious();
previous();

Different control statements available in java
Break: It is used to terminate the loop
Continue: It is used to skip the current iteration.
While: It is entry check loop.
Do While: It is a exit check loop.
If : Executes only when the condition becomes true.
if else : Executes the else part when the condition becomes false and executes if part when condition becomes true

What are the difference between equals() & hashcode()?
equals:

Used to compare the two string.
Hashcode:

Used to return the address where it stored.



>>**Can you declare main method as private**
No, we cant declare main method as private because JVM call only global access specifier (public) in main method

>>Main method as -> static public void main(String[]args)
It is possible to run the program

>>Main method as -> public void static main(String[]args)
Not possible

>> Main method as final
Possible to declare but unable to run

>>Can we overload the main() Method?
Yes, we can overload the main() method

```
unitLoginpass.java  Runall.java  TestngSample1.java  Cross.java  crossbrowser.xml  LibGlobal.java
1 package org.seleniumdaytwo;
2
3 public class sample {
4     private void method1() {
5     System.out.println("hello");
6     }
7     public static void main() {
8
9     }
0     public static void main(String[] args) {
1         sample s = new sample();
2         s.method1();
3
4     }
5 }
```

>>Can we declare main method as final?
Possible to declare but unable to run the program

>>What will happen "public static final void main(String[] args)
Possible to declare & able to run the program.
>>Can we use final keyword in abstract method?
No, because if we declare abstract as final we can't override

>>Can declare constructor as static?

- Constructor will invoke when object created
- In static method no need for object, it will directly call the method (Storage area : Static memory)
- Object stored in heap memory we can't declare both memory in same constructor

>>How will sort the values in Arraylist by ascending order?

- Collections.sort(Arraylist obj. ref name)

>>How to convert array to arraylist?

- Arrays.aslist()
- Collections.addAll()

>>How to convert arraylist to array ?

- List ref name . toArray();

>>How to convert string to array ?

- String s = "Greens"
  s.tocharArray(index value);

>>Difference between replace() & replaceAll()

Replace()

- We can use both single quotes and double quotes
- Not support regex

ReplaceAll()

- We can use only double quotes
- Support regex

<u>Regex</u>: It is symbol used for string pattern design

>>How you will iterate arrayList() ?

- For loop
- Enchanced for loop
- Iterator (use in both list & set)
- List Iterator (specially for list)
- Split Iterator

>>Iterator used in while loop

- Arraylist al = new ArrayList();
  al.iterator(); //change interface to iterator
  while(iterator.next){
  ….}

>>Vector supports multithreading?

- No, becoz vector is sync. & thread safe (one by one execution)

>>What will happen if we declare constructor as private?

- Unable to extends
- Unable to create an object

>>What will happen if we declare class as final?

- Unable to inherit (extends) but we can create an object.

>>What will happen if we declare object as final?

- Yes we can declare

  Final className c = new className();

  className c1 = new className(); //we've to create new object

  Eg : final WebDriver driver = new ChromeDriver();

  >>>>>we can't access firefox or ie browser

>>Method Chaining

- We can call one method by another method by this()
- It reduce object creation / memory space

>>Main method() can be override() ?

- No, becoz main method contains static, static cannot be override

>>public class A extends B implements InterfaceName

- possible

>>public class A implements B extends class

- Not possible

>>If we've two interface has same method & implement both interface in one class what will happen?

- Which interface you gave first, compiler will take the firstone

>>Difference between CompareTo & CompareIgnoreCase() ?

- **CompareTo :** used to compare between two string
- **CompareIgnoreCase :** used to compare between two string but ignore the case

```
1  package org.seleniumdaytwo;
2
3  public class sample {
4
5      public static void main(String[] args) {
6      String s = "hello";
7      String s1 = "mello";
8      //compareTo
9      System.out.println("CompareTo :");
10     System.out.println(s.compareTo(s1));//h is 5times lesser then m
11     //compareIgnoreCase
12     String s3 = "HELLO";
13     System.out.println("CompareIgnoreCase :");
14     System.out.println(s.compareToIgnoreCase(s3));
15     }
16 }
```

```
<terminated> Sample [Java Application] C
CompareTo :
-5
CompareIgnoreCase :
0
```

>>replaceFirst() & join() ?

- **replaceFirst() :** used to change the first value
- **join() :** To add the delimiter

>>Typecasting : One format to another format

2 Types

- Upcasting : Lower data --------> Higher data
  Eg : WebDriver driver = new ChromeDriver();
- Downcasting : Higher data --------> Lower data
  Eg : TakesScreenShot s = (TakesScreenshot)driver;

>>Deserialization

- Byte Stream convert to object (java)
- Eg : FileInputStream

>> Serialization

- Object (java) to Byte Stream convert
- Eg : FileOutputStream

>>Auto Boxing

- Datatype→Wrapper class (object)
- Eg : int ---> Integer

>> Un Auto-Boxing

- Wrapper class (object) → Datatype
- Eg : Integer --> int

>> CharAt() : return type **int** as well as **char**

sample.java

```
1  package org.seleniumdaytwo;
2
3  public class sample {
4
5      public static void main(String[] args) {
6      String s = "Greens";
7      char charvalue = s.charAt(2);
8      //to get the character
9      System.out.println("Character at the position of 2: "+charvalue);
10     int intvalue = s.charAt(1);
11     //to get ascii value
12     System.out.println("ASCII value :" +intvalue);
13     }
14 }
```

Console

<terminated> Sample [Java Application] C:\Program Files\Ja

```
Character at the position of 2: e
ASCII value :114
```