OOPS Concept

Encapsulation

- The process of binding state & behaviors of an object together is known as encapsulation.
- We can achieve encapsulation in java with the help of class.
- The main advantage of encapsulated class is we can achieve data hiding.
- The process of restricting the direct access to the states of an object and providing control access with the help of behavior of some object/class is known as Data Hiding.

Step to achieve Data Hiding-:

- Prefix the state of an object by using private class modifier.
- Define getter & setter method for the private data member.

Q. What is **encapsulation** where you used in Selenium?

In POM we have use encapsulation because we declare the element using @findBy annotation along with private keyword and we accessing it outside the class with the help of getter & setter method.

Inheritance -:

- It is an oops design principle which is used to established is-a-relationship between two objects, where one object will accrue state and behaviors of another object.
- Parent Class-: the class which provides states and behaviors to another class is known as parent class.
- Child Class-: The class which accrue state and behaviors from another is known as child class/sub class.
- The purpose of inheritance is to achieve generalization and specialization.
- The bottom to up approach where the common members from the bottom layer will be taken to create the top layer is known as generalization.
- The top to bottom approach where the bottom layer will be created having members specialize only for them is known as specialization.
- We can achieve inheritance in java using extends and implements keyword.
- There are 5 types of inheritance
- Single level, Multi-level, Hierarchical level, Multiple level, Hybrid level.

Q. What is **inheritance** & where we used in Selenium?

In our framework BaseClass is an example for inheritance, because all the classes will be extended by BaseClass because of code reusability.

Similarly, architecture of selenium WebDriver is an example for inheritance

Polymorphism-:

- It is an oops principle which helps the member of a class to execute more than one form.
- In java polymorphism can be classified into two types based on what time the behavior of the member will be decided.

1. **Compiler Time Polymorphism-:**

- it is a process where compiler decides which behavior of a member should be use during compilation time.

Ex-: Method overloading, constructor overloading, shadowing.

2. **Run Time Polymorphism-:**

- In run time polymorphism the bind between method call statement and method implementation is happening during run time.
- We can achieve run time polymorphism with the help of method over riding.

Method over-riding-: The process of providing new implementation for the super class method with the help of sub class.

Abstraction-:

- It is an oops principle which help us to hide implementation from the user and provide only necessary details.
- In java we can achieve abstraction with the help of two members

1. Abstract class 2. Interface

1. Based on abstraction we can classify methods into two types

a. Concreate method **b.** abstract method

a. A method which had implementation or method body is known as concreate method.
b. A method which does not have implementation is known as abstract method.

Points to remember-:

- We can have a constructor inside abstract class.
- An abstract class can have both concreate as well as abstract method.
- We cannot create an object for abstract class.

Q. What is **abstraction** and where you used in Selenium?

WebDriver driver = new ChromeDriver();

This is an example for abstraction. Because we know all the methods of WebDriver how it functions but we don't know how they have been implemented.

Ex-: get(), close(), quit() etc…

Difference between Abstract Class and Interface?

- Abstract Class can have both abstract and concrete methods. Can have constructors, and instance variables, and supports partial abstraction.

interface -: It is a keyword. interface is a user define non-primitive data type which is use to achieve multiple inheritance and 100% abstraction.

- By default, interface is an abstract member.
- We can not create an object for interface.
- We can compile an interface and generate a class file for that.
- We can not make an interface either private or protected. It can be either default or public.
- We can not make an interface as final member.
- We can have static methods inside interface.
- We can execute the interface.
- Static method of interface will not get inherited.
- The abstract method and public static final variable of the interface will be inherited.

Class -: It is a keyword. Class is a user defined non-primitive data type which help us to store a detail of a object it is also known as blue print of an object.

- In a class we can create members of class (variable & method), variables are used to store states of an object. Methods are used to store behaviours of an object.
- Every class name is a non-primitive data type.

- Java provide us with many built in class which can also be used as non- primitive data type ex String, System etc…
- The members that we create inside the class block is known as class members.
- The class members are classified into 2 types
- Static members and non-static members

static -:

- It is a keyword. It is a modifier. Any member which is prefix with static modifier is known as static member of a class.
- A static member will have the memory allocated inside class static area.
- Here we have 3 static members -: static method, static variable, static initializer.
- A static variable is shared among all instances of a class. This means that any instance of the class can modify the value, and the change will be reflected across all instances.
- A static method belongs to the class and not to any specific instance of the class. It can be called without creating an object of the class.
- Static blocks are used for static initialization of a class. This block runs once when the class is loaded into memory and is used to initialize static variables.

Non-static members-:

- A member which is not prefix with static modifier is known as non-static member of class
- There are 4 types of non-static members, Non-static variable, method, initializer and constructor.
- The non-static member gets their memory allocated inside the object of a class (instance of class). We can access the non-static members of the class only with the help of address of the object.
- To access the non-static members, it is mandatory to create an object.
- Non-static (instance) variables are associated with an instance of the class. Each object of the class has its own copy of the instance variables.
- Instance variables are stored in the heap memory, and every object has a unique memory space for these variables.
- Non-static methods are called using an instance of the class and can access both instance and static variables and methods.
- These methods have access to instance variables and are used to

operate on the data unique to the instance.

- Non-static initialization blocks are executed every time an instance of the class is created. They are used to initialize instance variables with more complex logic than what can be done in a constructor.
- These blocks run before the constructor when an object is created.
- Constructors are used to initialize new objects and can be considered non-static methods with no return type.

What is an object-:

A block of memory created at the run time inside heap area which represents the real-world object.

Q. Constructor-:

It is a non-static member of a class

It helps us to load all the non-static members into the object.

The name of constructor should always be same as that of its class

name. It looks just similar to a method except return type.

There are 2 types constructor-:

No argument constructor -: which does not have any formal argument declared.

Parameterise constructor-: which has formal argument declared.

In java it is mandatory to have a constructor defined inside each & every class created.

If the user fails to create a constructor, then compiler will add a noargument constructor inside class during compilation, which is known as default constructor.

Q. Constructor overloading-:

A class having more than one constructor is known as constructor overloading.

We can have any number of constructors inside the class but they have to follow certain rules

- The name of the constructor should always be the class name.
- The declaration of the constructor should be differing either in length or data type or order of declaration.

Q. Constructor chaining -:

One constructor calling another constructor is known as constructor chaining.

Q. What is **Method Overloading** and where you have used in Selenium?

The process of creating more than one method with the same name but different formal arguments is known as method overloading.

Role for method overloading

- Names of methods should be same
- The formal argument must differ in either length or type or order of declaration.

In our framework all the methods of ITestListener are overridden in Listener Implementation class.

Ex-: onTestSuccess, onTestStart, onTestFailur etc…

- Similarly, all the methods of WebDriver, search context, javaScriptExecutor, TakeScreenshot and WebElement are overridden the RemoteWebDriver class

Loading process of class-:

- The class static area will be created for initial class.
- All the methods present inside the class will be loaded to method area.
- If any static method is present its reference will be store inside class static area.
- If any static variables are present, it should be assigned with default value inside class static area.
- If any static initializer is present, they should execute directly without storing.
- The loading process is said to be completed.

Q. Can you explain the use of this and super keywords in Java?

- this refers to the current object, used for accessing instance variables and methods.
- super refers to the parent class, used to call methods and constructors from the superclass.

Q. What's the difference between Comparable and Comparator?

- Comparable is used to define the natural ordering of objects within a class using the compareTo method.
- Comparator is used to define multiple ways to order objects, typically in a separate class, using the compare method."

Q. Exceptions in Java:

Exceptions are unexpected events that occur during the execution of a program.

- **Checked Exceptions:** These are checked at compile-time. For example, IOException, SQLException.
- **Unchecked Exceptions:** These occur at runtime and are not checked during compilation.
  For example, ArrayIndexOutOfBoundsException, NullPointerException.
- **Error:** These are not exceptions but problems beyond the control of the programmer. For example, OutOfMemoryError, StackOverflowError.

Q. How do HashMap and Hashtable differ?

- HashMap is unsynchronized and allows null keys and values.
- Hashtable is synchronized and doesn’t allow null keys or values.

Q. What is the difference between throw and throws?

- throw is used to explicitly throw an exception.
- throws are used in a method signature to declare that it can throw exceptions.

Q. What is the difference between final, finally, and finalize in Java?

- **final** Keyword used to define constants, prevent inheritance (classes), and prevent method overriding.
- **finally** Block in exception handling that always executes after try-catch
- **finalize** Method called by garbage collector before object destruction.

Q. Explain the Singleton Design Pattern in Java.

- Ensures a class has only one instance and provides a global point of access to it.
- It can be achieved by making the constructor private and providing a static method to get the instance.

Q. What is String?

- String is an object that represents a sequence of characters.
- It is part of the java.lang package and is immutable, meaning once created, it cannot be modified.
- Operations that seem to modify a String (like concatenation) actually create a new String object.
- It supports methods like length(), substring(), charAt(), equals(), toUpperCase(), and toLowerCase().
- It is commonly used in Selenium for locating elements, generating test data, and handling dynamic content.

String has three types of constructor

1. String()
2. String(String string) -: it is use to create a string object with the given string literal as its states.
3. String(Char [ ] ch) -: it is use to create a string object and initializing with the characters present inside character array to convert character array into string.

Convert char [ ] to string

- Using String class (String(Char [ ] ch);
- Using valueOf() method.
- Without using built-in

Example -:

```
        char[] ch = { 'A', 'p', 'p', 'l', 'e' };
        String s1 = new String(ch);
        String value = s1.valueOf(ch);
        String str = "";
        for (int i = 0; i < ch.length; i++) {
            str = str + ch[i];
        }
        System.out.println("with built in function : "+str);
```

Convert String to Char[ ]

We can convert a string to char [ ] by using

- toCharArray()-: it is a non-static method present inside string class. The return type of it is char[ ].
- charAt(index)-: it helps us to obtain the character present in given index. The return type of char() is character.

Example-:

```
        String string = "orange";
        char [] ch= string.toCharArray();
        System.out.println(ch);
        System.out.println(string.charAt(0));
        System.out.println(string.charAt(4));
```

List of important non-static methods-:

- Char[ ] **toCharArray()**
- Char **charAt(int index)**
- **length()**
- String **toUpperCase()**
- String **toLowerCase()**
- String **concat(String str)**
- Boolean **contains(charSequence String)**
- Int **indexof(char ch)** -: it returns of the character if it is present in the given String. Else it returns -1.
- String **substring(int start index)-:** it generates a new string from the given index position up to the last character of the string.

- int **indexOf(String str)-:** it return the position of given string if it is present in the string lese it return -1.
- String **substring(int start index int end index)** -: it generate a new string from the given start and end index.
- byte[] **getBytes()**-: it converts the string into an Array of bytes.
- String [ ] **split**-: It breaks the given string into multiple strings.
- Boolean **endswith(String str)**-: it returns true if the string ends with the given string.
- Boolean **startswith(String str)**-: it returns true if the string start with the given string.
- String **trim()-:** It removes all the white space present in the beginning and end of the string.
- Boolean **equals(String str)-:**
- Boolean **equalsIgnorCase(String)**
- Int **compareTo(String)**
- Int **hashCode()**

Inheritance-: OOPS Principal

Single level inheritance-: If the inheritance is the single level is known as single level inheritance.

```
class Parent { }
class Child extends Parent {--}
```

Multiple Inheritance-: the inheritance of more than one level is known as multi level inheritance.

```
class Parent {
class Chile extends Parent { --- }
class GrandChild extends Child { --- }
```

Hierarchical Inheritance-: If one parent class having more then one child class is known as hierarchical inheritance.

```
Class Parent { }
Class Child extends Parent { --- }
Class Child1 extends Parent { --- }
```

Multiple Inheritance-: If one sub class is inheriting more than one super class at the same level is known as multiple inheritance.

Hybrid Inheritance-: It is a combination of two or more types of inheritance.