

CORE JAVA

JDK:

Java Development Kit is essential whenever we need to run a program in Java. It has JRE (Java Runtime Environment which contains predefined classes) and JVM (Java Virtual Machine used for memory allocation and object creation). Mostly used JDK versions are 1.7 & 1.8.

OOPS:

OOPS is Object Oriented Programming Structure. It's a method of implementation in which program is organized as collection of Class, Method, Object, Encapsulation, Inheritance, Polymorphism and Abstractions. In our project I have implemented OOPS Concepts in lot of places.

Class:

It's a collection of methods and objects. In my project I have used lot of predefined classes like

- ChromeDriver
- FireFoxDriver
- InternetExplorerDriver
- Actions
- Robot
- Select

Methods:

It's a set of actions to be performed. We write the business logics in the methods. In my project I have used lot of predefined methods like

- get()
- getTitle()
- getCurrentUrl()
- getText()
- getAttribute()

Encapsulations:

Wrapping up of data's into single unit is encapsulation. In my project I used to maintain all the locators in the POJO class as object repository and using the object indirectly I will access the locators in the different class using POM with page factory concept.

Inheritance:

Accessing one class property in to another class using extend keyword is Inheritance. In my Project I used to maintain all the reusable methods in base class and using extends keyword I will access all the methods when required.

Types:

- Single – One parent class directly support one child class using extend keyword.
- Multilevel – More than one parent class directly support one child class using extend.
- Multiple – More than one parent class parallelly support one child class. But it won't support in java due to priority problem but we can achieve it using Interface.
- Hierarchal – One parent class support more than one child class.
- Hybrid – Combination of single and multiple inheritances.

Polymorphism:

Taking more than one form is polymorphism.

Types:

- Method Overloading (or) Static Binding (or) Compile time polymorphism.
- Method Overriding (or) Dynamic Binding (or) Run time polymorphism.

Method Overloading:

Method overloading is if the same method is going to overload again and again with different arguments. In my project I have used method overloading concepts like `println()`, `sendKeys()`, `wait()`, etc.

Method Overriding:

Changing business logic from super class in to subclass using the same method name is method overriding. In my project I have implemented method overriding concepts like

- `TakeScreenShot – getScreenShotAs()`
- `IRetryAnalyzer – retry()`
- `IAnnotationTransformer – transform()`

Abstraction:

Hiding the implementation part is called abstraction. We don't to create object for abstraction.

Types:

- Abstract Class
- Interface

Abstract Class:

The class that supports both abstract methods as well as non abstract methods is abstract class. Using `extends` keyword we use to write our business logics in any other classes. In my project I use `By` as an abstract class.

Interface:

It supports only abstract methods. Using `implements` keyword we can write implementation in any other classes. In my project I used many interfaces like

- `WebDriver`
- `WebElement`
- `JavaScriptExecutor`
- `TakeScreenShot`
- `Alert`
- `Wait`
- `List, Set, Map`

Scanner:

It's a predefined class present in `java.util` package to get input from the user at runtime.

(Note: Default Java Package is `java.lang` and `Object` is super class for all java classes.)

Syntax: `Scanner refName = new Scanner (System.in);`
`refName.ScannerMethods ();`

Singleton Class:

It's a class that can have only one object (an instance of the class) at a time.

Access Specifier:

- Public – It's a global level Access Specifier. When we use public as access Specifier, we can access in same package as well as different package by using object and extends.
- Protected – When we use Protected as Access Specifier, we can access by using object and extends in the same package and we can access using extends in different package.
- Default – It's a package level Access Specifier. We can access by using object and extends when it comes to same package.
- Private – It's a class level Access Specifier. We can use it only in a class.

Access Modifier:

- Static – It can be declared at method and variable level only. If we declare as Static, we can call using method name and variable name without creating object.
- Abstract – It can be declared at class and method level only. If we declare class as abstract, we can't create object and if we declare method as abstract, we can't write business logics.
- Final – It can be declared at class, method and variable level. If we declare class as final, we can't inherit. If we declare method as final, we can't override. If we declare variable as final, we can't change the value.

String:

Collection of characters or word enclosed within double quotes is String. It's a predefined class available in java.lang package. Default value of string is null.

Methods:

- length () – to find length of the string.
- charAt() – to pick a particular character in the string.
- toUppercase() – convert string into uppercase.
- toLowercase() – convert string into lowercase.
- replace() – to replace a particular character in the string.
- indexOf() – to find the index of a particular character in the string.
- lastIndexOf() – to find the last index of a particular character in the string.
- startsWith() – to check string starts with a particular character.
- endsWith() – to check string ends with a particular character.
- isEmpty() – to check whether the string is empty or not.
- isConcat() – to join two strings.
- equals() – to check two strings are equal or not.
- equalsIgnoreCase() – to check two strings are equal or not without case.
- trim() – to remove the unwanted spaces in the string.
- substring() – to print from any index to another index in the string.

Types:

- Literal String – Stored inside the heap memory. Duplicates will be stored in the same memory.
- Non Literal String – Stored in the heap memory. Duplicated will be stored in different memory.
- Immutable String – Stored inside the heap memory. To join two strings we use "concat". All Strings share same memory.
- Mutable String – Stored in the heap memory. To join two strings we use "append". First String and Concatenated String shares the same memory address.

Constructor:

Once the object is created, the constructor will invoke automatically. Constructor name should be same as the class name.

Types:

- Default Constructor (or) Non Parameterized Constructor.
- Argument Based (or) Parameterized Constructor.

this (): It's used for constructor chaining.

Super (): It's used to call the parent class constructor.

Array:

It's used to store multiple values of similar data type in a single reference name. The values are stored based on Index. It allows duplicate values. Memory waste will be high since memory will be allocated at compile time.

Syntax: example: `int refName[] = new int[size];`

Collections:

To overcome the disadvantages of Array, we are going for collection. It's used to store multiple values of dissimilar data type in a single reference name. Memory waste will be low since memory will be allocated at run time.

Types:

- List
- Set
- Map

List:

- Index Based One.
- Allows Duplicate.
- ArrayList, VectorList and LinkedList prints in insertion order.
- Iterate using normal and enhanced for loop.
- ArrayList is Asynchronous and Non Thread Safe.
- VectorList is Synchronous and Thread Safe.
- **Methods:**
 - `size()` – to find the size of the list.
 - `get()` – to pickup any value from the list.
 - `add()` – to add values in the list.
 - `remove()` – to remove a particular value in the list.
 - `set()` – to replace values in the list.
 - `indexOf()` – to find the index position.
 - `lastIndexOf()` – to find the last index position.
 - `addAll()` – to copy all values from one list to another list.
 - `retainAll()` – to print common values of two list.
 - `removeAll()` – to print uncommon values of two list.

Set:

- Value Based One.
- Does not allow duplicate.
- Iterate using enhance for loop as it's a value based one.
- HashSet – Random order, LinkedHashSet – Insertion order, TreeSet – Ascending order.

Map:

- Key and value pair combination.
- Key doesn't allow duplicate and value allows duplicate.
- Iterate using method "entrySet()".
 - HashMap – Random order, Key allows one null and values allow n null.
 - LinkedHashMap – Insertion Order, Key allows one null and values allow n null.
 - TreeMap – Ascending order, Key ignores null and values allow n null.
 - Hashtable – Random order, Key ignores null and values also ignore null.
- HashMap is Asynchronous and Non Thread Safe.
- Hashtable is Synchronous and Thread Safe.

Exception:

It's like an error. Whenever it occurs the program will terminate itself. It can be handled by

- Try – contains error message.
- Catch – contains solution for the error.
- Finally – prints the output in both cases (error and no error).
- Throw – handle only one exception which is mentioned inside method.
- Throws – handles multiple exceptions which is mentioned in the method level.
- **Methods:**
 - getMessage() – to print the particular exception.
 - printStackTrace() – print the unknown exception name.
- **Types:**
 - Checked Exception(Compile Time Exception)
 - ❖ FileNotFoundException
 - ❖ IOException
 - ❖ SQLException
 - ❖ ClassNotFoundException
 - Unchecked Exception (Run Time Exception)
 - ❖ Arithmetic Exception
 - ❖ NullPointerException
 - ❖ IndexOutOfBoundsException
 - ❖ ArrayIndexOutOfBoundsException
 - ❖ StringIndexOutOfBoundsException

[**Note:** Super Class for all the Exception is Exception (or) Throwable.]

Generics:

It supports only similar data types. It's from JDK 1.5 Version. Inside the generics we need to pass only the wrapper class.

Wrapper Class:

It's of classes of data types. It converts all data types in to objects.