OOP SET 1- EASY

(Interview Questions 1 to 10)

1. What is the OOP concept in java? (Important)

For any programming language to follow OOP they need to follow 5 principles:

1. Classes and Object
2. Encapsulation
3. Inheritance
4. Polymorphism
5. Abstraction

2. What is encapsulation?

- In any class the instance variable plays an very important role. Hence they should not be accessed out the class to avoid illegal assignment.

- Hence we wrap the instance variable with special methods called getters and setters which can be used for assigning and retrieval of the value.

- Getters and setters will be public functions and can be accessed outside the class by object creation.

- Inside the Getter and Setters we can add special validation code to ensure that the value which we try to assign are valid values or not!

3. What is Polymorphism? (Important)

- Polymorphism means the entities are going to have same name but will take different form or body.
- In OOPS Polymorphism are of 2 Types
    - ***i.*** **Compile Time (Static Polymorphism)**
        1. Method Overloading
    - ***ii.*** **Run Time (Dynamic Polymorphism)**
        1. Method Overriding
- Benefit of Polymorphism:
    i. In order to make the learning curve easy we need to ensure that we use less function names and use the same functions names to perform different task. (Overloading)
    ii. This way getting comfortable with the framework functionalities will be quick and people (non technical team members) will adapt to our framework or code faster!

4. ***Difference Between Classes and Objects?***
    - Classes are the rules which are imposed on the objects
    - Objects are real world entities and rules created in Classes (Variables and functions) are imposed on these objects.

5. ***What is collection in Java? (Important)***
    - One of the most used data structures in Java is Array.
    - But Array has One major problem
        i. ***Size of the Array is fixed!***
            Meaning we cannot increase the size of the array when dynamically.
    - So either we will over utilize the array and get the Array Index out of Exception
    - Or do under utilization of the array and have memory wastage
    - To over come the drawback the Array. Collections was introduced!
    - Collection is a set of 5 classes which are present in java.lang package. [**Remember the package name]**
        - ArrayList
        - LinkedList
        - Vector
        - HashSet
        - TreeSet
    - Collection is used to store large set of data:

Collection	**Non-Idempotent/Idempotent**	**Key Feature**
ArrayList	Non-Idempotent	Fast Retrieval
LinkedList	Non-Idempotent	Fast Storage
Vector	Non-Idempotent	Capacity increases by 2X
HashSet	Idempotent	Stores unique value using Hash Code for faster retrieval
TreeSet	Idempotent	Retrieves the value in Alphabetical or Ascending order

6. ***What is out in System.out.println?***

In Java System.out.println(): The statement can be broken into 3 parts which can be understood separately as:

- **System:** It is a final class defined in the **java.lang package**
- **.out:** This is an instance of PrintStream type, which is a public and static member field of the System class
- **.println():** As all instances of PrintStream class have a public method println(), hence we can invoke the same on out as well

7. In How many ways we can create an object? (Very Important)

Points	*Snippet*	*Comment*
New Keyword	Bank b = new Bank();	
Class.newInstance()	Bank b1 = Bank.class.newInstance(); System.out.println(b1);	
Object.clone()	ClassName implements **Cloneable** { @Override protected Object clone() throws **CloneNotSupportedException** { ***//invokes the clone() method of the super class*** **return super.clone();** } }	

8. Why Java is not 100% Object-oriented?

- No Java is not 100% programming language.
- Why?
  - Java still supports **Primitive datatypes** even though **Wrapper classes.**
    - **Which breaks the rule of reference variable and encapsulation**
  - Java also uses static keywords which enables to access the ***variables and functions*** without creating the project.
    - Which breaks the rule that object needs to be created for accessing variables and function.

9. What is Object Oriented Programming?

- Object oriented Programming is a style of developing real world application.
- It ensures that the code that we write are scalable/maintainable and easy to refactor!
- Principles of OOPS:
  1. ***Classes and Object***
  2. ***Encapsulation***
  3. ***Inheritance***
  4. ***Polymorphism***
  5. ***Abstraction***

10. What is Java Package and which package is imported by default? (Important)

- Java Packages are used to store/hold Java related files like Classes, Interfaces, Enums and records for better management
- **The default package** used in java **is java.lang**

OOP SET 2- EASY

(Interview Questions 11 to 18)

11. Have you used OOP concept in your Framework ? Where and How? (very Important)

1. **Classes and Objects:**
    i. Page Object Classes were created for each unique ui component.
2. **Encapsulations:**
    i. As Instance variables are the most important variable in your entire app code. We ensure that instance variables are private and are only accessed via getters and setters.
        1. Example POJO Classes.
3. **Inheritance:**
    i. In order to make code reusable we implement Child and Parent Class relationship.
    ii. Test Base class is parent class for test classes
    iii. IJson Is an interface implement by all api pojo which helped in achieving **loose coupling**
4. **Abstraction:**
    i. We have hidden the complexity of selenium, rest assured and Appium by creating browser utility classes/android utility and api helper classes
    ii. These classes have wrapped the complex selenium/restassured and appium code so that people who are writing test need not need to learn or deal with any of these libraries. Hence in this way abstraction was achieved.
5. **Polymorphism:**
    i. We have ensured that function names are reusable because it helps in remembering less names and functionalities.
    ii. So, for that we did method overloading. That's the compile time polymorphism
6. **<u>PRO Tip</u>**
    In order to use OOPs effectively we have ensured to use Design Patterns for our classes.

Design Pattern	**Comment**
Singleton Design pattern	For DB Connectivity and Logger
Transfer Object Design Pattern	POJO classes, DAO classes
Fluent Design Pattern	For making scripts more human readable
Factory Design Pattern	For producing certain data or specify objects for our classes

12. What Upcasting and down casting in Java? (Important)

- *Converting lower-level data type to higher level data type is called as Upcasting or Widening*
    - Parent p = new Child(); //child class is a lower DT and is converted to Parent or higher datatype;
    - double d = 10;
- ***Whereas converting higher level data to lower data type is called as down casting***
    - Child c =(Child) p // Parent ref is casted to Child Class
    - int x = (int) 10.5;

13. What are object classes and name some object class methods? (Important)

- Object class is parent class or super class for the all Classes in Java.
- Some popular functions that are present in Object class:
    i. **toString():** returns the string representation of this object.
    ii. **hashCode():**returns the hashcode number for this object.
    iii. **equals(Object obj**): compares the given object to this object.
    iv. **finalize():** is invoked by the garbage collector before object is being garbage collected.

14. Which class is the superclass of all classes? (GK)

- All classes have an inherit parent classes and that class is **Object Class**.
- Object class belongs to **java.lang**

15. What is difference between Heap and Stack Memory? (Important)

Heap	**Stack**
It is created when the JVM starts up and used by the application as long as the application runs	The stack memory is a physical space (in RAM) allocated to each program or thread at run time
Objects/arrays/Instance variables will be created in HEAP memory	Local variables – variables created inside a method will be created in STACK
Any variables created in HEAP is initialized with default value	Any variable created inside the stack is a never initialized with default value
It's a slower compared to HEAP	FASTEST memory in your computer because it deals with execution.

16. What is the difference between equals() and == in Java (Important)

==	*equals()*
== is considered an operator in Java.	equals() is considered as a method in Java.
It is majorly used to compare the reference values and objects.	It is used to compare the actual content of the object.

17. Differentiate between the constructors and methods in Java?

- ***Constructor***
    i. A constructor is a special function in Java which has the same name as that of the class
    ii. A constructor is invoked automatically when the object of the class is instantiated.
    iii. The job of the constructor is to initialize the instance variable of the class
    iv. A constructor does not return any value.
    v. A constructor cannot be static
- **Method:**
    i. A method can be static or non static
    ii. A method needs to explicitly called using either using object reference (in case of non static) or class Name(in case of static)
    iii. A method may or may not return any value.
- Both in methods and Constructor - overloading is possible
- **Constructor Overriding is never possible in Java – Remember**

18. Can we override static method?

- ***For Method Overriding Object creation is required***
- When we have static method we use ClassName to access the method
- We can declare static methods with the same signature in the subclass, but it is not considered overriding as there won't be any run-time polymorphism.

- ***Hence the answer is 'No'***

OOP SET 3

(Interview Questions 18 to 30)

18. ***Why Method Overloading is not possible by changing the return type of method only?***

- In order to do method overloading the java focuses on the name of the method and it's parameter.
- Method overloading is decided on:
    i. Parameter Counts
    ii. Sequence of the datatypes for those parameters
- Method overloading is not decided by
    i. Visibility label (public/private/protected/default
    ii. Its not decided by the return type of the function.
- **Remember Method overloading is an example of Compile Time Polymorphism**
- **Compile time Polymorphism is also called as static polymorphism. (<u>The word static has nothing to do with Static keyword in Java)</u>**

19. ***Difference between Compile time polymorphism and run time polymorphism?***

- **Compile Time Polymorphism happens before the execution of the program**
    - Method Overloading is an example of Compile Time Polymorphism
    - Method Overloading will involve only one class.
    - Compile Time Polymorphism is also called as Static Polymorphism
- **Runtime Polymorphism happens during the execution of program.**
    - Method Overriding is an example of Run Time Polymorphism.
    - Method Overriding involves 2 classes and these classes needs to have child parent relationship.
    - Runtime Polymorphism is also called as Dynamic Polymorphism.

20. Why cannot we override static methods in Java? (IMPORTANT)

- For Method Overriding three things are mandatory:
    i. 2 Classes and inheritance relationship between them.
    ii. Both the classes should have same method name/parameters and visibility label and return type
    iii. Reference should be created of Parent and Object should be created of child!
- Method overriding is an example of Run time Polymorphism means object creation is mandatory
- When use static keyword we try to use the class name instead of object for accessing variables and functions.
- Hence, we are breaking the rules of Method Overriding.
- In Java static methods cannot be Overridden.

21. What are four types of Java access modifiers? (Important)

Access Modifier	***Comment***
Private	Private methods and variables are accessible only within the same class
Public	Public methods and variables are accessible anywhere in the project
Protected	Protected methods and variables are accessible only within the child classses
Default	1. Default methods and variables act as PUBLIC within in the same package and will act PRIVATE outside the package. 2. Default visibility label should be avoided

22. What is a constructor? why is it used? where u cannot use the constructor?

- A constructor is a method in Java which has the name as that of the class
- The job of the constructor is to initialize the instance variable of the class.
- A constructor is called automatically during the **object creation.**
- A constructor cannot be called explicitly.
- **A constructor cannot be referred with ClassName.**
    - ***i.*** **Hence a constructor cannot be static.**

23. What is default constructor?

- When we don't create a constructor for the class, then java will create a default constructor the class.
- A default constructor is just an empty constructor
    - ***i.*** Example `className(){`

`}`

24. Is there Constructor class in Java? (Important)

- Constructor class is used to manage the constructor metadata like the name of the constructors, parameter types of constructors, and access modifiers of the constructors.
- This class is present in **package java.lang.reflect**

25. Does constructor return any value?

- No a constructor cannot return any value.
- The job of the Constructor is to construct i.e to initialize the instance variable of the class!

26. ***Can we create a constructor as private? (Important)***

- Yes we can have a constructor that can be private.
- This can only happen in a scenario when we use Singleton Design Pattern

27. Can we create a constructor of abstract class? (Important)

- Yes you can have the constructor of the abstract class.
- Abstract class constructor will be called by the Child class constructor using the super keyword.

28. Can we make a constructor as Static? (Important)

- Nope!
- A constructor belongs to the instance of the class.
- Static variables and functions belongs to class
- Hence We cannot make the constructor as static

29. Can constructor be overridden? (Important)

- No constructor cannot be overridden.
- Constructor can be overloaded.

30. What is constructor chaining in Java? (Important)

- When a constructor call another constructor then its called as constructor chaining
- Constructor chaining can be done using 2 keywords
    i. this: this keyword is used to call the another constructor within the same class.
    ii. Super: super keyword is used to call the parent constructor (abstract class constructor) from the child class constructor
        1. Super keyword should be the first statement inside the child class constructor

OOP SET 4

(Interview Questions 31 to 36)

31. ***What is Serialization and Deserialization?***

- When we convert a java object to a ***JSON*** object it is called as Serialization
- The reverse (When we convert a JSON object to Java Object its called as De-serialization)
- There are 2 popular 3rd party libraries that can help us in achieving this
    i. JACKSON
    ii. GSON

32. ***What is the purpose of Jackson library***

- *The purpose of Jackson library is to achieve serialization and deserialization in our API Testing*
- *We can convert a POJO object to JSON object and vice versa using JACKSON library*

33. ***How to convert Java Object to JSON Object? (Important)***

- ***Snippet:***
- Create an object of Gson Class
  Gson g = new Gson();
- Pass the object reference of the class that we want to convert into JSON object to the ***toJson*** method.

```
Customer c = new Customer();//Java object
String data =g.toJson(c);
```

- The Gson will return the JSON representation of the Java object in string format

34. ***How to you convert the API JSON Response to Java Object for validation?? (Important)***

- This is the purpose of Deserialization. We need to deserialize the JSON repose to Java Object and then we can do the field validation using getters or equals() functions.

```
Gson g = new Gson();
Customer customerReference = g.fromJson(json, Customer.class);
```

35. How to convert a Java Object to JSON object using Jackson? (Important)

- *Note: Although we have used gson library for the Serialization and Deserialization. The same result can be achieved with JACKSON library too.*

- *You need to add the dependency in the pom.xml*

```
<dependency>
<groupId>org.codehaus.jackson</groupId>
<artifactId>jackson-mapper-asl</artifactId>
<version>1.9.13</version>
</dependency>
```

```
ObjectMapper mapper = new ObjectMapper(); Customer
 customer = new Customer();
 //Object to JSON in String
String jsonInString = mapper.writeValueAsString(customer);
```

31. How to convert a JSON object to java object using Jackson? (Important)

```
ObjectMapper mapper = new ObjectMapper(); String
 response = "{'name' : abc}";

//JSON from String to Object
User user = mapper.readValue(jsonInString, User.class);
```

OOP SET 5

(Interview Questions 37 to 42)

37. What is static and final? (Important)

- ***Static Keyword***
  - ***i.*** Static keyword deals with memory.
  - ***ii.*** When a variable is static it means it will only one memory allocated throughout the entire application
  - ***iii.*** We may be able to update the value of static variable but there will only going to 1 memory allocation
  - ***iv.*** Static keyword is applicable at **5 places**:
    1. **Classes**: Inner classes in Java can be static variable.
    2. **Variables**: Pure Class variables.
    3. **Functions**: functions can be static.
    4. **Blocks**: Static code block.
    5. **Static Import**: import all static variables and functions from other class without taking Classname.
  - ***v.*** **Constructor cannot be static**
  - vi. Static variables and Static functions can be accessed without object creation!
  - ***vii.*** **<u>You should use static keyword as minimum as you can as it breaks OOPs Fundamentals</u>**
- **Final Keyword**
  - *i.* Final Keyword is applicable at 3 places
    1. Variables: Final Variables are constant
       - ***a.*** **All Final variables will be marked as static keyword.**
       - **b.** **<u>But All static variables may not be final!</u>**
    2. Function: Final functions cannot be overridden.
       - *a.* This means only functions from Parent class be final
    3. Classes:
       - *a.* Classes made final cannot be extended
       - *b.* This means that Final Classes cannot be Parent Classes
       - *c.* Note : Parent Classes are always abstract Class

38. What is abstraction?

- Abstraction is an OOPs concept which means that all complex implemation of the code should be hidden or wrapped over simple methods and classes for better understanding and maintenance of the project.
- Example **(Benefit of Abstraction):**
    i. We should never use Selenium WebDriver or Appium Scripts directly into our Test Scripts
    ii. They are always wrapped over our special util classes so that people who are writing testscript need not worry of Selenium Scripts.
    iii. They can contribute in writing tests with just knowledge of Java.
    iv. This reduces the learning curve of our framework.

39. What is a Super keyword ?

- Super Keyword is used to refer Parent Class variables/functions from child class when they have same name.
- Super Keyword is also used to call the parent class constructor from the child class constructor.
- ***Note only Child constructor can only Parent Class constructor.***

40. What is this keyword?

- This keyword is used to differ instance variable and local when they have same name.
- This keyword is also used for refer to the current object
- This keyword is also used to do constructor chaining within the same class.

41. What is differences between Abstraction Class and Interfaces?

- Abstract Class and Interfaces are both used to make system code Loosely Coupled.
- We need to achieve Loose Coupling in our code so that is easy to refactor and make changes quickly.
- ***Abstract Class***
    i. All parent classes are abstract in nature.
    ii. Abstract Class may or may not have abstract function.
    iii. Abstract functions are functions with out any body and their implementation is taken care by Child Classes
    iv. Abstract Class can also have a constructor/ and proper functions
- ***Interface:***
    i. All functions in Interface are Abstract in nature
    ii. All variables are static final i.e they are constant
    iii. Interface is also used to achieve Multiple Inheritance in Java

42. What is the abstract method? can we write static methods in abstract class?

- *Abstract Methods are methods without any body.*
- *All Parent classes are abstract class and they may or may not contain abstract method*
- *We may have static methods in abstract for sure!*
- ***But abstract methods can never be static!***
- ***Abstract methods can never be private!***

OOP SET 6

(Interview Questions 42 to 49)

42. How do you convert integer to string?

a. *To convert integer/float/double/Boolean/char/short/byte/long into String all you need to do is add an empty String to it*
    i. *Example*

```
int x = 10;
String d = x+ "";
```

43. Advantages of ENUMS over Strings?

a. Enums are constant in JAVA.
b. They were created so that developers can pass contant string value in a more disciplined way.
c. All the values in Enums are static final.
d. Enums are used to hold configurational constant values like
    i. Roles
    ii. LOCAL or REMOTE

44. How do you convert string to integer?

a. To convert String to Primitive Data Type we use Wrapper Classes
b. To Convert String to Integer
    i. String x =”100”;
    ii. int x = Integer.parseInt(x);
c. To Convert String to Long
    i. String x =”100”;
    ii. long x = Long.parseLong(x);
d. To Convert String to Double
    i. String x =”100.5”;
    ii. double x = Double.parseDouble(x);

45. Is String class final?

a. *Yes String is final class.*
b. *It cannot be extended*

46. Whats the difference between String and String Buffer?

a. The String class is immutable*.*
b. The StringBuffer class is mutable.

c. String class is slower while performing concatenation operation.
d. StringBuffer class is faster while performing concatenation operation.

e. String class uses String constant pool.

f. StringBuffer uses Heap memory

47. Difference between String, String Builder, and String Buffer?

String	***String Builder***	***String Buffer***
String is immutable in Java.	String Builder is mutable classes and added in Java 1.5	String buffer is a class for String manipulation and was created in Java 1.4 and they are mutable class
String overrides equals() and hashCode() methods. **(Non Syncronized)**	String builder is **not syncronized**	All methods are **syncronized**
String is a final class	String builder is a final class	StringBuffer is a final class
Low Performance as Thread Safe	High Performance	Low Perfomace as Thread Safe

48. What are other immutable classes in Java apart from String?'

a. The immutable objects are objects whose value can not be changed after initialization.
b. immutable means unmodified or unchangeable.
c. Rules for creating an immutable class:
    i. Final class, which is declared as final so that it can't be extended.
    ii. All fields should be private so that direct access to the fields is blocked.
    iii. No Setters
    iv. All mutable fields should be as final so that they can not be iterated once initialized.
d. Examples of Immutable Classes
    i. String
    ii. Wrapper Classes
    iii. File Class
    iv. Enums are Immutable
    v. URL and URI Classes are Immutable

49. String is immutable or mutable?

a. String is Immutable Class

OOP SET 7

(Interview Questions 49 to 55)

49. What is Inheritance?

- ***a.*** Inheritance is one of the fundamentals of the Object-Oriented Concept
- ***b.*** In Inheritance one class can access the **Non private properties** of another class without creating the object
- ***c.*** There is a child parent relationship that gets created between the 2 classes
- ***d.*** Child Class **extends** Parent Class
- ***e.*** Inheritance is used
  - ***i.*** to reduce code duplicacy.
  - ***ii.*** Make the code maintainable.
  - ***iii.*** Achieve **Loose Coupling** in our application.
- ***f.*** Child Class is called as sub classes
- ***g.*** Parent class is a called as super class.

50. What is inheritance? Can we use inheritance in Interface?

- ***a.*** Explain Question 49 and add the below points
- ***b.*** Can we use Interface in Inheritance ? -Yes
- ***c.*** Interface is used to achieve Multiple Inheritance in Java
- ***d.*** Interface is used to achieve Loose Coupling in your source code

51. What are multiple inheritances?

a. When a child class tries to access the properties of 2 parent simultaneous it called as Multiple Inheritance
b. Multiple Inheritance is not directly supported in Java
c. To achieve multiple inheritance, we need to use Interface
d. A class can only extend one class at a time
e. A class can implement N numbers of interfaces

52. Where you used Inheritance in your framework?

a. Yes we have used Inheritance in **APITestBase** Class
b. We used inheritance in **BrowserUtils** for WebDriver where all **UIPage classes extends BrowserUtils**
c. We used inheritance in **AndroidUtils** for Appium where all **Android Page classes extends Android**

53. Can we achive Multiple Inheritance in Java?

a. Multiple Inheritance is achieved with the help of Interface in Java

54. What is the differences between multilevel and multiple Inheritance?

- ***a.*** Multi level inheritance is consist of more than one single inheritance
- ***b.*** Example GrandParent – Parent – Child
  - ***i.*** GrandParent- Parent is Single Inheritance
  - ***ii.*** Parent- Child is single Inheritance
- c. When a child class tries to access the properties of 2 parent simultaneous it called as Multiple Inheritance
- d. Multiple Inheritance is not directly supported in Java
- e. To achieve multiple inheritance, we need to use Interface
- f. A class can only extend one class at a time
- g. A class can implement N numbers of interfaces

55. What is runtime polymorphism or dynamic method dispatch?

- a. Dynamic method dispatch is the mechanism by which a call to an overridden method is resolved at run time, rather than compile time

OOP SET 8

(Interview Questions 56 to 60)

56. Explain Singleton Design Pattern (Important)

- ***a.*** Singleton Design Pattern is a way of creating only one instance of a class.
- ***b.*** In Singleton design Pattern the constructor will be private.
- ***c.*** There will static reference variable of the same class.
- ***d.*** We use Singleton Design Pattern for DB connectivity and Logger Creation for the our framework

57. Explain POJO ? (Important)

- ***a.*** POJO stands for plain old java object
- ***b.*** In POJO the instance variables of the class will be private
- ***c.*** There will one or more parameterized constructor
- ***d.*** There will be getters and setters for the setting and retrieving the values of the instance variables
- ***e.*** There will be a **toString**() to get the entire state of Object. (what are the current values of the instance variables)

58. Explain Page Design Pattern?

- ***a.*** Page Object Design Pattern is a very commonly used Design Patter for the maintaining the UI Page Components .
- ***b.*** In Page Object Design Pattern Class
  - ***i.*** Variables are private static final and of By DataType
  - ***ii.*** Functions or Methods will perform the Test Steps for our Test Scripts
  - **iii.** **Every Page Function should return some data or page objects**

59. Explain Transfer Object Design Pattern? (Important)

a. POJO is called the Transfer Object Design Pattern.
b. We call POJO Transfer Object Design Pattern because we can covert POJO to different forms like JSON or XML or try to retrieve the values from database and store into POJO.
c. So POJO takes the data from one for and converts to another form.

60. What is comparable and comparator?(Important)

a. Comparable and Comparator in Java are very useful for sorting the collection of objects.
b. The Comparable interface has **compareTo(T obj)** method which is used by sorting methods.
c. Comparator interface **compare(Object o1, Object o2)** method need to be implemented that takes two Object argument.

d. Comparable interface is in java.lang package whereas Comparator interface is present in java.util package.
e. We don’t need to make any code changes at client side for using Comparable, Arrays.sort() or Collection.sort() methods automatically uses the compareTo() method of the class. For Comparator, client needs to provide the Comparator class to use in compare() method.

OOP SET 9

(Interview Questions 61 to 66)

61. What is interface? (Important)

- Interface is Java Entity In which variables are public static final and functions are public abstract
- You cannot have private function in an interface
- As the functions are abstract in interface you cannot create the object of interface.
- Interface is used to achieve loose coupling in our system.
- It ensures methods are overriding by the classes which implements the Interface
- A class can implement n number of interfaces.
- We can achieve multiple inheritance only with help of Interfaces

62. Can we write non-abstract methods in Interface? (Important)

- **Prior to Java 8**, it wasn't possible to add non-abstract methods in Java but now we can add non-abstract static, default, and private methods in the Java interface.
- The static and default methods were supported as part of interface in Java 8 and you can even add private methods on an interface **from Java 9 onwards**.

63. What is an interface and how have you used it in your framework

- We have used different Interfaces:
    i. ITestListeners in TestNG
    ii. IJsonBody
    iii. IRetryAnalyzer in TestNG
    iv. RequestSpecification in Rest Assured
    v. Response in RestAssured

64. Difference between interface and abstract class? (Important)

- Both Abstract and Interface act as a parent entity for a child class
- They both used to achieve inheritance in your system
- In Abstract Class we can concrete or full methods but in Interface we cannot have concrete methods (prior to Java 8 only) from Java 8 you can have concrete functions
- We can have instance variable in Abstract Class.
- We don’t have instance variables in Interface
- Abstract Class can have constructor
- Interface don’t have constructor

65. There is a variable declared inside the Interface. Can we change the value of the variable?

- No you cannot because variables in interface are public static final.
- Which means they are constant and only meant to be read and not to be updated

66. What is the Functional Interface? (Very Important)

- A functional interface is an interface that contains only one abstract method.
- A functional interface can have any number of default methods.
- Functional Interface is also known as Single Abstract Method Interfaces or SAM Interfaces.
- Runnable, ActionListener, Comparable are some of the examples of functional interfaces.
- We use the annotation @FunctionalInterface for marking the interface as functional interface.
- A functional interface can extend another interface ***only when it does not have any abstract method.***

  Examples of Functional Interface in Java:

  1. Predicate<T>
  2. Consumer<T>
  3. Function<T,R>

OOP Set 10

(Interview Questions 67-71)

67. What is the diff between array and Array List?

- ***a.*** ArrayList is an extension of an Array
- ***b.*** The drawback of an Array that the once the array is declared with a particular size we cannot change the size of the array.
- ***c.*** So if we are creating a real world application we cannot know before hand, the exact size of the array
- ***d.*** Hence we are either going to under utilize the array or over utilize the array and reach **ArrayIndexOutOfBoundException**
- ***e.*** To over the draw back of Array came ArrayList. ArrayList is an resizable Array in which the size of array are dynamically increased or decreased.
- ***f.*** ArrayList is part of **Collection Series (**and belongs to java.util package.
  - ***i.*** **Note Collections is a Class and support a different purpose**
  - ***ii.*** **ArrayList/LinkedList/Vector/HashSet/TreeSet Belong to COLLECTION and COLLECTIONS**
- ***g.*** Both Array and ArrayList will store the continuous memory block
- ***h.*** Both have the concept of Indexing
- ***i.*** Both are created in the HEAP Memory
- ***j.*** Array List has a special concept called Capacity and the default capacity of the ArrayList is 10

68. How to convert array to Array List?

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

ArrayList<String> al = new ArrayList<String>(); al.add("Raj");

al.add("Prerna");

al.add("Ankit");

                    Object[] d = al.toArray(); //Convert ArrayList to Object[]

System.out.println(Arrays.toString(d));

          }
```

Address to previous memory location	Value	Address to next memory location

69. What is linked list and array list?

LinkedList	*ArrayList*
Different memory locations or nodes are connected in the via links	All the data is stored in continuous memory
LinkedList uses the concept of node. A node basically consist of 3 things: NODE=	ArrayList Uses a concept of Capacity. The capacity is a limited memory space given to Arraylist during its instantiation. ArrayList<String> al = new ArrayList<String>(); ArrayList with capacity 10 is created!
Data is scattered across different memory location in HEAP and then connected via links	Data is stored in a continuous memory location. The data is well organized and stored in the HEAP memory
Faster add() and remove() opertation	Faster get() opertation.

70. how to traverse array list?

a. ForLoop

```
for(int index =0; index<al.size();index++) {
System.out.println(al.get(index));
}
```

b. ForEach:

```
for (String data: al) { System.out.println(data);
}
```

c. Iterator:

```
Iterator<String> allterator=al.iterator();
 while (allterator.hasNext()) {
String data = (String) allterator.next();
     System.out.println(data);
 }
```

71. Explain Collections Class?

a. It is a utility class.

b. It defines several utility methods that are used to operate on collection.

c. It contains only static methods.

OOP Set 11

(Interview Questions 72)

72. What is the diff between List and Set?

- ***a.*** Both List and Set are part of the Collection Series
- ***b.*** Both of List and Set are Interface and extends the Super Interface Collection
- ***c.*** List Interface is implemented by 3 classes
  - ***i.*** **ArrayList** :
    1. It's a resizable array.
    2. Stores the elements in continuous memory
    3. Uses the concept of capacity to increase and decrease the size of the arraylist.
    4. ArrayList is **non-syncronized**
    5. **Hence ArrayList is Fast in a MultiThreaded Program**
  - ***ii.*** **LinkedList:**
    1. Linked List is another which implements List Interface
    2. Linked List also implements Queue Interface
    3. Linked List stores the data in a **concept** of node
    4. A node consist of 3 things
       a. Address of previous node
       b. Data
       c. Address to next node
    5. Every node is connected with other node and the data can be stored in different memory address.
  - ***iii.*** **Vector*:***
    1. Vector is an extension of Array List
    2. Array List increases the capacity with 50% of size
    3. Vector increases the capacity with 2x of size
    4. Vectors are **synchronized.**
    5. ***Hence Vector is slow in a MultiThreaded Program***

Thread safety is a computer programming concept applicable to multi-threaded code. Thread-safe code only manipulates shared data structures in a manner that ensures that all threads behave properly and fulfill their design specifications without unintended interaction.

- All the 3 classes are Non Idempotent in Nature.
- Which means they can store duplicated values in them!

- **d. Set:**
  - i. **Set** is a data structure which does not store duplicate values in them
    1. **Set** Interface is implemented by 2 classes
       - **a. HashSet**
         - i. HashSet is a class

which will store
the data based on
the HashCode of
the elements

ii. These HashCode helps in
faster retrieval of values

iii. Duplicate Values cannot be stored inside the HashSet

b. TreeSet:

i. TreeSet is a class which also implements Set Interface

ii. TreeSet retrives the value either in

1. Alphabetical or Ascending order of the value of elements

✔ ArrayList, LinkedList, HashSet,LinkedHashset and TreeSet in Collection Interface and HashMap,LinkedHashMap and Treemap are all non-synchronized. ✔ Vector in Collection Interface is Synchronized

OOP Set 12

(Interview Questions 73-81)

73. What is an exception? And what's the usage of it?

a. Java execute that code at runtime but For reasons if Java fails to execute that statement or block of code or method then Java gives an exception.

b. Exception must be handled properly. ***If it is not handled, program will be terminated abruptly.***

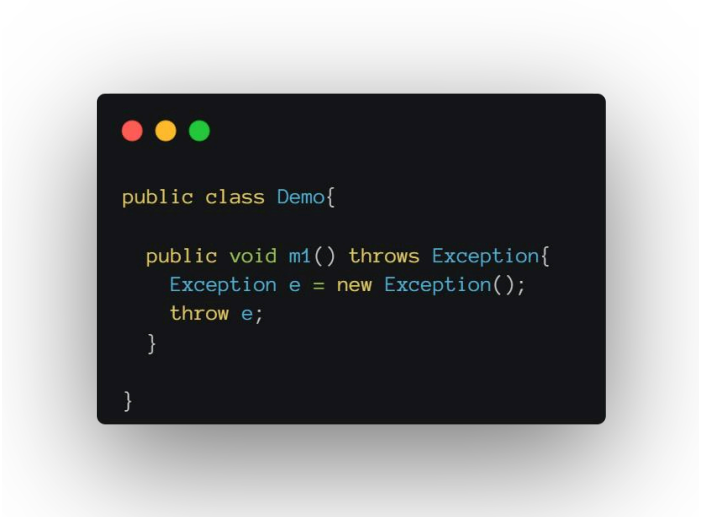
74. What are different types of exception?

a. In Java there are 2 types of Exceptions in Java

- ***i. Compile Time also called as Checked Exception***
  1. *Exception that are shown by Java before the execution of the code is called as Compile Time Exception*
  2. Example: SQL Exception, IOException
- ***ii. Run Time also called as Unchecked Exception***
  1. *Exception that are shown by Java during the execution of the code is called as Run Time Exception*
  2. Example: Null Pointer Exception, ArrayIndexOutBoundException

75. What does Throw do?

a. throw is a keyword in java which is used to throw an exception manually

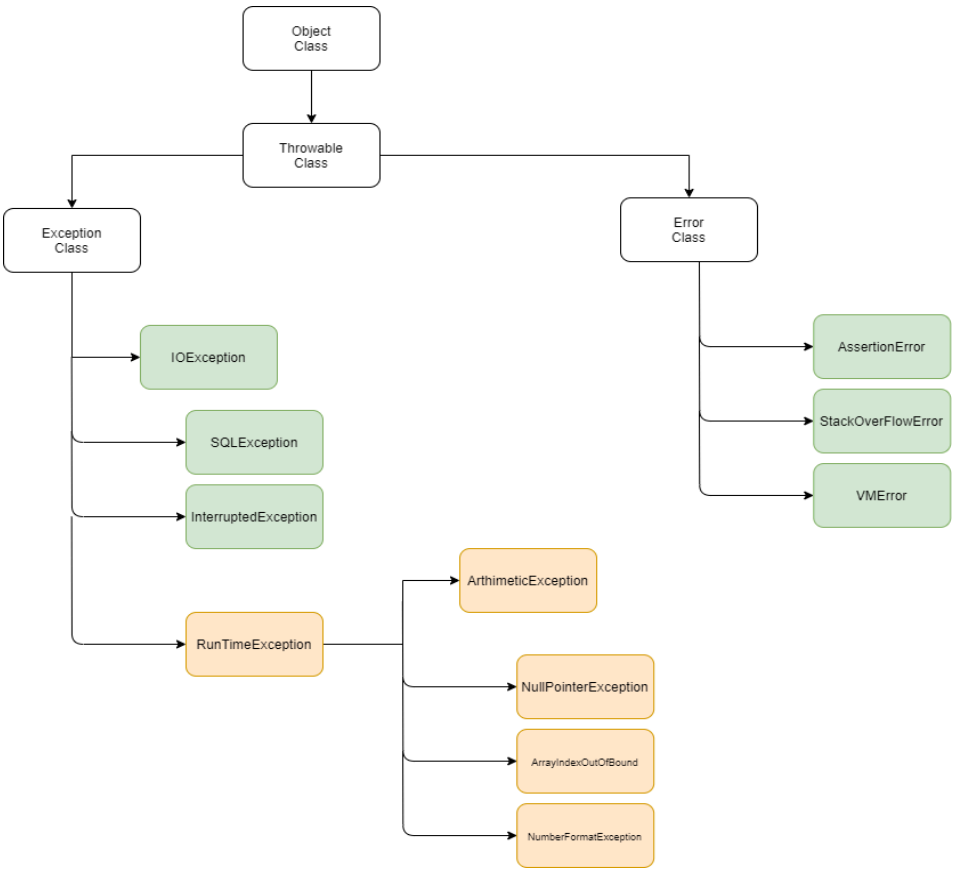


```
public class Demo{

  public void m1() throws Exception{
    Exception e = new Exception();
    throw e;
  }

}
```

76. Architecture of Exceptions in Java?



77. What is the difference between throw and throws?

- ***a.*** There are 2 ways using which exceptions can be handled
  - ***i.*** Throws
  - ***ii.*** Try catch
- b. **throw** is a keyword in java which is used to throw an exception manually

Most Preferred approach of Handling Exception is always via Try Catch. NEVER USE throws for handling exception

78. What is checked and unchecked exceptions?

- ***a.*** Compile Time Exceptions are called as Checked Exceptions
- ***b.*** Run Time Exceptions are called as Unchecked Exceptions.

79. Can try with multiple catches possible?

- ***a.*** Yes ,we can a single try block with multiple catch block.

80. Can multiple try with a single catch be possible?

- ***a.*** No, we cannot have multiple try block with a single catch block

81. Can we rethrow an exception?

- a. An exception can be rethrown in a catch block

Rest Assured Interview Questions

1. Can you explain RequestSpecification request = RestAssured.given(); ?

- RequestSpecification is an interface in the RestAssured library that defines the request specification for an HTTP request. It allows you to specify things like the HTTP method (GET, POST, etc.), headers, parameters, and authentication details for the request.

- RestAssured.given() is a static method that returns an instance of RequestSpecification

- ***This implementation is based on the Builder pattern, which allows you to easily construct an HTTP request by chaining method calls.***

2. How can you send a request body/payload for a POST request?

- To send a request body or payload for a POST request using RestAssured, you can use the body() method on the RequestSpecification object.
- Snippet:

```
RequestSpecification request = RestAssured.given();
    request.header("Content-Type", "application/json");
    request.body("{ \"name\": \"Jatin\", \"age\": 30 }");
    Response response = request.post("/api/users");
```

Explanation:

- In this example, we first create an instance of RequestSpecification using RestAssured.given().
- we set the Content-Type header to application/json using the header() method.
- then request body using the body() method, passing a JSON string as the argument.
- And we then a POST request to the end point “api/users”

3. What are the types of Status codes?

- HTTP status codes are three-digit numbers that indicate the outcome of an HTTP request. There are five categories of status codes, each with a different range of values:
- **Informational (1xx):** These status codes indicate that the request has been received and is being processed. Examples include 100 (Continue) and 102 (Processing).
- **Success (2xx):** These status codes indicate that the request was successfully received, understood, and accepted. Examples include 200 (OK), 201 (Created), and 204 (No Content).
- **Redirection (3xx):** These status codes indicate that the client must take additional action to complete the request. Examples include 301 (Moved Permanently), 302 (Found), and 307 (Temporary Redirect).
- **Client Error (4xx):** These status codes indicate that there was an error on the client side of the request. Examples include 400 (Bad Request), 401 (Unauthorized), and 404 (Not Found).
- **Server Error (5xx):** These status codes indicate that there was an error on the server side of the request. Examples include 500 (Internal Server Error), 502 (Bad Gateway), and 503 (Service Unavailable).

4. What is REST?:

- REST stands for **Representational State Transfer**, which is an architectural style for designing web services. It is a set of guidelines and constraints that define how web services should be designed and operated.

- At its core, REST is **based on a client-server model**, where the client sends requests to the server, and the server sends responses back to the client. These requests and responses are typically sent over HTTP, using standard HTTP methods like **GET, POST, PUT, and DELETE.**

5. What is GET Method?

- The GET method is one of the standard HTTP methods used for retrieving resources from a server. It is used to request data from a specified resource, using a URL-encoded query string.

- When a client sends a GET request to a server, it is asking the server to return a representation of the specified resource. The server will then search for the requested resource, and if found, will send a response back to the client containing the requested data.

6. What is POST Method?

- The POST method is one of the standard HTTP methods used for submitting data to a server. It is used to send data to a server to create or update a resource.

- When a client sends a POST request to a server, it includes a request body that contains the data to be submitted. The server will then process the request body, and either create a new resource or update an existing one, depending on the specific API.

7. What is PUT Method?

- The PUT method is one of the standard HTTP methods used for updating a resource on a server. It is used to send data to a server to update an existing resource.

- When a client sends a PUT request to a server, it includes a request body that contains the data to be updated. The server will then process the request body, and update the specified resource with the new data.

8. What is DELETE Method?

- The DELETE method is one of the standard HTTP methods used for deleting a resource on a server. It is used to send a request to a server to delete the specified resource.

- When a client sends a DELETE request to a server, it includes a URL that specifies the resource to be deleted. The server will then process the request and delete the specified resource.

9. What is HEAD method?

- The HEAD method is one of the standard HTTP methods used for retrieving metadata about a resource, without actually retrieving the resource itself. It is similar to the GET method, but instead of returning the entire response body, it only returns the HTTP headers for the resource.

- When a client sends a HEAD request to a server, the server will process the request and send back a response that includes only the HTTP headers for the specified resource, without the actual content of the resource.

10. What is OPTIONS method?

- The OPTIONS method is one of the standard HTTP methods used for retrieving information about the communication options available for a resource. It is used to determine the HTTP methods that can be used to interact with the resource, as well as the supported request and response formats.

- When a client sends an OPTIONS request to a server, the server will process the request and send back a response that includes a list of HTTP methods that can be used to interact with the resource, as well as other communication options.

11. How to use Basic authentication in automation?

- Basic authentication is a simple authentication mechanism that involves sending a base64- encoded username and password in the HTTP headers of a request

```
RestAssured.baseURI = "http://example.com/api"; RestAssured.authentication =
      basic("username", "password");

Response response = given()
.when()
        .get("/users");

      System.out.println(response.getBody().asString());
```

- Remember:
- Basic authentication is not considered a very secure method of authentication, as the username and password are transmitted in plaintext over the network, making them susceptible to interception and eavesdropping.

- While the password is base64-encoded, it is not encrypted, and an attacker with access to the network traffic can easily decode the password and gain access to the system.

- For this reason, it is generally recommended to use other more secure authentication mechanisms, such as OAuth or JSON Web Tokens (JWT).

12. How do you extract the values of JSON and how do you validate response?

- In RestAssured, you can extract values from JSON responses using the JsonPath class. Here's an example:

```
Response response = RestAssured.get("/api/users");

String firstName = response.jsonPath().getString("data[0].first_name"); int
        userId = response.jsonPath().getInt("data[0].id");

System.out.println("First name: " + firstName);
        System.out.println("User ID: " + userId);
```

API Testing Interview Question

(Interview Questions 1)

1. How to Test an API ? or What will you test in an API?

- There are several areas that should be tested in an API to ensure that ***it is reliable and meets the needs of its end users. API Testing should include:***

    - **Functionality**: Test that the API is able to perform the functions it is designed to do. This may involve testing specific API endpoints and the input and output data for those endpoints.
        - *Correct Response*
        - *Correct Status Code*
        - *Schema Validation*

    - ***o*** **Performance**: Test the performance of the API to ensure that it can handle the expected load and response times. This may involve load testing to simulate a high volume of requests and ***measuring the API's response times.***

    - **Security**: Test the security of the API to ensure that it is secure against attacks such as ***injection attacks and unauthorized access***. This may involve testing the API's ***authentication and authorization mechanisms***, as well as its input validation.

    - ***o*** **Compatibility**: Test the compatibility of the API with different clients and systems. This may involve testing the API with ***different client libraries and platforms to ensure that it works as expected.***

    - **Error handling*****: Test the API's error handling capabilities*** to ensure that it handles errors gracefully and provides useful error messages to its users.

Scenario Testing Interview Question

(Interview Question 1)

1. When we have many production fixes, how do you prioritize which one should be deployed first

When you have many production fixes that need to be deployed, it is important to prioritize them in order to ensure that the most critical issues are addressed first.
There are several factors that you may want to consider when prioritizing production fixes:

Severity: Fixes that address critical issues, such as security vulnerabilities or data loss, should generally be prioritized over fixes for less severe issues.

Impact: Fixes that affect a large number of users or have a significant impact on business operations should generally be prioritized over fixes that have a smaller impact.

Urgency: Fixes that need to be deployed immediately to prevent further issues or downtime should generally be prioritized over fixes that can wait.

Dependencies: If a fix depends on other fixes being deployed first, you may need to prioritize those dependencies in order to ensure that the fix can be deployed successfully.

- It can be helpful to create a priority matrix or use a tool such as a Kanban board to track and prioritize production fixes.

- This can help you to clearly visualize the status of each fix and make informed decisions about which ones to deploy first.