Java Interview Questions

1. **Write a program to swap 2 numbers without a temporary variable? Swap 2 strings without a temporary variable?**

//swap numbers

```
int a = 10;
int b = 20;
a = a + b;// first this should be there a=10+20=30
b = a - b; // b= 30-20=10
a = a - b; // a=30-10=20
System.out.println("a is " + a);
System.out.println("b is " + b);
```

//swap strings

```
String x = "Hello";
String y = "Welcome";
x = x + y; // HelloWelcome
System.out.println(x.length()); // 12
System.out.println(y.length()); // 7
y = x.substring(0, x.length() - y.length());
System.out.println(y.length()); //5
// start from y length
x = x.substring(y.length());
System.out.println(x);
System.out.println(y);
```

2. **Find out how many alpha characters present in a string?**

```
String given="wefeqf878979797fewfewrf879797efds&^&^*^*^";
```

Regular expression → [^a-zA-Z]

//1 way

```
String replaced=given.replaceAll("[a-zA-Z]", "");
int alphaChar = given.length()-replaced.length();
System.out.println(alphaChar);
```

//2 way

```
//^ caret negates and provides only alphabets
String replaced= given.replaceAll("[^A-Za-z]","");
System.out.println(replaced.length());
```

3. How to find whether given number is odd number?

```
int a=22;

if (a % 2 != 0) {
        System.out.println(a + " is an odd number");
} else {
        System.out.println(a + " is not an odd number");
}
```

4. How to find out the part of the string from a string? What is substring? Find number of words in string?

```
String a="Welcome to the interview sessions with CodeinHub";
System.out.println(a.substring(39));

String[ ] words = a.split(" ");
System.out.println(words.length);

for (String string : words) {
        System.out.println(string);
}
```

5. Write java program to reverse String? Reverse integer? Reverse a string word by word?

Reverse String:

```
String a= "Hello CodeinHub";
// 1 way

StringBuffer sb=new StringBuffer(a);
System.out.println(sb.reverse());

//2 way
```

```
String reverse="";
for (int i=a.length()-1; i>=0; i--) {
        reverse=reverse+a.charAt(i);
}
System.out.println(reverse);
```

// 3 way

```
String reverse1="";
for (int i=a.length(); i>=1; i--) {
        reverse1=reverse1+a.substring(i-1, i);
}
System.out.println(reverse1);
```

Reverse Integer

// 1 way

```
int a=123;
String numbers=String.valueOf(a);
StringBuffer sb=new StringBuffer(numbers);
System.out.println(sb.reverse());
```

// 2 way

```
int number = 1234567;
String convert = String.valueOf(number);

char[] arrayChar = convert.toCharArray();
int size = arrayChar.length;
for (int i= size -1; I >= 0; i--) {
        System.out.print(arrayChar[i]);
}
```

//3 way

```
int currentNum=12345;
int reversedNum=0;
```

```
while(currentNum!=0) {
        reversedNum=reversedNum*10+currentNum%10;
        currentNum=currentNum/10;
}
System.out.println(reversedNum);
```

6. Write a program to sort array in ascending order?

```
//1 method
int [] a= {12,24,2,4,1};
Arrays.sort(a);
System.out.println(Arrays.toString(a));
```

7. How can you convert a String into an array? Array to String?

Converting String to an Array

```
String a="CodeinHub";
char[] array=a.toCharArray();
System.out.println(array.length);
for (int i=0; i<array.length; i++) {

        System.out.println(array[i]);
}
```

Converting Array to String

```
//1 way
System.out.println(String.valueOf(array));

//2 way
System.out.println(Arrays.toString(array));
```

8. Verify whether given two strings are equal?

```
String a="Hello";
String b= "Hello";
```

1 method

```
if (a.equals(b)) {
        System.out.println("String "+a+" and String "+b+" are equal");
}else {
```

```
            System.out.println("String "+a+" and String "+b+" are not equal");
    }
```

2 method

```
if (a.compareTo(b)==0) {
            System.out.println("String "+a+" and String "+b+" are equal");
}else {
            System.out.println("String "+a+" and String "+b+" are not equal");
}
```

9. Write java program to find second largest number in the array? Maximum and minimum number in the array?

// second largest number in the array

1. easiest way

```
int[] numArray= {12,13,12,15,0, -1};
Arrays.sort(numArray);
System.out.println(numArray[numArray.length-2]);
```

2. more efficient way

```
int arr[] = {14, 46, 0, 86, 92,-1};
int largest = arr[0];
int secondLargest = arr[0];

for (int i = 0; i < arr.length; i++) {
        if (arr[i] > largest) {
                secondLargest = largest;
                largest = arr[i];
        } else if (arr[i] > secondLargest) {
                secondLargest = arr[i];
        }
}
System.out.println("2nd largest number is:" + secondLargest);
```

// maximum and minimum number in the array

//1 way

```
int[] numArray = { -3, 12, 13, -5, 12, 15, 0, -1 };
int smallest = numArray[0];
int biggest = numArray[0];
for (int i = 0; i < numArray.length; i++) {
        if (numArray[i] > biggest) {
                biggest = numArray[i];
        } else if (numArray[i] < smallest) {
                smallest = numArray[i];
        }
}

System.out.println(smallest);
System.out.println(biggest);
```

//2 way

```
Integer[] array = {20,0,40,-1,60,70};
int min = Collections.min(Arrays.asList(array));
int max = Collections.max(Arrays.asList(array));

System.out.println("Maximum value is "+max);
System.out.println("Minimum value is "+min);
```

10. How can you remove all duplicates from ArrayList?

```
ArrayList aList = new ArrayList();
aList.add("John");
aList.add("Jane");
aList.add("James");
aList.add("Jasmine");
aList.add("Jane");
aList.add("James");
```

// 1 way

```
HashSet set = new HashSet(aList);
System.out.println(set);
```

// 2 way

```
HashSet hset=new HashSet();
for (Object name : aList) {
        hset.add(name);
}
System.out.println(hset);
```

11. Versions of java you worked with? What version of java do you currently use in your framework? Difference between JRE, JDK, JVM ?

Version Name	**Release Date**
Java SE 7	July 2011
Java SE 8	March 2014
Java SE 9	September 2017
Java SE 10	March 2018
Java SE 11	September 2018
Java SE 12	March 2019

JRE stands for **Java Runtime Environment** which we usually download as a Java software. The JRE consist of the Java Virtual Machine, Java platform classes, and supporting libraries. The JRE is the runtime component of Java software and is all we need to run any Java application.

JDK stands for **Java Development Kit** is a superset of the JRE and includes everything that the JRE contains. Additionally, it comes with the compilers and debuggers tools required for developing Java applications.

JVM stands for **Java Virtual machine**. It translates and executes the Java bytecode. It's the entity which transform Java to become a "portable language" (i.e. write once, run anywhere). Java compiler generates bytecode for all the Java code and converts into class files.

12. What are the primitives and wrapper classes?

Primitives are data types in Java. There is total of **8 primitive data types in Java: byte,**

short, int, long, float, double, char, boolean.

Every primitive data type has a class dedicated to it and these are known as wrapper classes. **These classes wrap the primitive data type into an object** of that class.

13. Difference between = = and =.

= we are assigning value
= = comparison operator (reference and address comparison) When we use it with primitives it is checking the value.

14. What is a main method? Why do we need one in java? Do we have to have a main method in java?

Main method is the starting point of an application. JVM starts execution by invoking the main method of some specified class, passing it a single argument, which is an array of strings. Whenever we execute a program, the main() is the first function to be executed. We can call other functions from main to execute them. It is not mandatory to have main method in java, without main() our Java code will compile but won't run.

15. What is the difference between String and StringBuffer? String and StringBuilder? What is mutable and immutable? StringBuffer vs StringBuilder?

The most important difference between **String and StringBuffer** in java is that **String object is immutable** whereas **StringBuffer object is mutable.** Once String Object is created **we cannot change** it and everytime we change the value of a String there is actually a new String Object getting created. For example we cannot reverse string directly, only through using StringBuffer class.

There are **2 ways to make String mutable:** 1. by using **StringBuffer** 2. by using **StringBuilder**.

The StringBuffer and StringBuilder Class are mutable means we can change the value of it without creating a new Object. Objects of StringBuilder and StringBuffer Classes live inside **heap memory.**

immutability vs. mutability

✓ String is immutability class it means once we are creating String objects it is not possible to perform modifications on existing object. (String object is fixed object)

✓ StringBuffer and StringBuilder are mutability class it means once we are creating StringBuffer/ StringBuilder objects on that existing object it is possible to perform modification.

```
class Test {
        public static void main(String[] args) {
                String a="Hello";
                String b="Hello";

          StringBuffer sb=new StringBuffer("Hello CodeinHub");

          a=a.concat(" Codein");
                System.out.println(a);

                sb=sb.append(" Hub");
                System.out.println(sb);

        }
}
```

StringBuffer vs StringBuilder?

Both Classes are mutable, except **StringBuffer is thread-safe (synchronized)** and **StringBuilder is not thread-safe (non synchronized)** which makes **StringBuilder faster compare to StringBuffer.**

16. How can we access variable without creating an object instance of it? What is Instance variables and how you use it? What is difference between local and instance variable?

Variables which are declared **inside a method or constructor or blocks are called local variables.** Local variables are created when a method is called and destroyed when the method exits.
Variables which are declared **inside the class, but outside a method, constructor or any block are called instance variables.** We can access instance variable by creating an Object of the class they belong to. Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
By declaring variable as a static we can access it from different class without

creating an Object - those variables called class variables and also known as static variables.

Example of Instance Variable from the framework

```
@FindBy(xpath="//img[contains(@src, 'logo')]")
        public WebElement logo;
```

17. Difference between Instance Variable and static Variable? What is static keyword in java? Where did you use static in your framework?

1. Static variables are declared with the **static** keyword in a class, but outside a method, constructor or a block. Whereas, **Instance** variables are declared in a class, but outside a method, constructor or any block. To access instance variables we need to create an object of the Class they belong to whereas static variables can be accessed without object creation by the classname.

```
public class Q13 {

String ApplicationUrl = "https://www.codeinhub.com/"; // Instance variable

static String ApplicationUrl2 = "https://www.orangeHRM.com/"; // Static
variable

        public static void main(String[] args) {

                System.out.println(ApplicationUrl2);
                Q13 t1 = new Q13();
                System.out.println(t1.ApplicationUrl);
        }
}
```

2. **Class variables only have one copy** that is shared by all the different objects of a class, whereas every object has its own **personal copy of an instance variable**. So, **instance variables across different objects can have different values** and when we make changes to the instance variable they don't reflect in other instances of that class **whereas class variables across different objects can have**

only one value.

3. Static variables are created when the program starts and destroyed when the program stops whereas instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.

Static keyword in java:

- Static keyword means that the variable or method belongs to the class and shared between all instances.
- Using static keyword we can access class variables and method without object reference
- Static methods can not call/refer Non Static members

Usage of static keyword in framework:

In our utility package we have a class where we store common methods, such as wait, switch between frames, clicking on buttons, selecting values from drop down. So those methods are written using static keyword and we can can easily access them in our program.

```
public static WebElement waiting(WebElement element) {
     WebDriverWait wait = new WebDriverWait(driver, 30);
     return wait.until(ExpectedConditions.elementToBeClickable(element));
}
```

18. What is constructor? Use of constructor in class ? Can you make the constructor static? What is difference between constructor and method? Can we overload a constructor?

What is constructor

A constructor in java is a block of code similar to a method. Constructor called when an instance of a class is created. A constructor is a special method whose task is to initialize the object of its class.
Constructors cannot be abstract, final, static.
Rules to create constructor:

1. Constructor name class name must be same.

2. Constructor do not have any return type.
3. Constructor may or may not have parameters.

Usage of Constructor

The primary use of constructor is to **initialize the instance and/or class variables.** Constructors are special function which are called automatically when we create object of the class. So, once we create object of the class all the variables get initialize, and we don't need to write extra code for initialization of variables.

Constructor is the property of an object while static has nothing to do with object. That's why there is nothing like static constructor. But we have static block to do the similar task as constructor i.e. initialization of fields etc.

Difference between Constructor and Method

- Constructor must not have return type whereas method must have return type.
- Constructor name same as the class name whereas method may or may not the same class name.
- Constructor will be called automatically whenever object is created whereas method invoke explicitly.
- Constructor compiler provide default constructor whereas method compiler doesn’t provide.

Example of constructor from framework

creating constructor to initialize instance variables

```
public class LoginPage extends BaseClass{

        public LoginPage() {
                PageFactory.initElements(driver, this);
        }

        @FindBy(xpath="//img[contains(@src, 'logo')]")
        public WebElement logo;
}
```

WE CAN OVERLOAD CONSTRUCTOR (using different parameters)

19. Super vs super()? this vs this()? Can a super() and this() keywords be in same constructor?

this vs this()

- **this keyword** is used to refer current object and differentiate between local and instance variables

```
public class ThisKeyword {

        String name;
        int age;

        ThisKeyword(String name, int age){
                this.name=name;
                this.age=age;
        }
 }
```

- **this()** is used to access one constructor from another where both constructors belong to the same class.

```
 public class ThisKeyword4 {

        int z;

        ThisKeyword4() {
                System.out.println("This a default constructor");
        }

        ThisKeyword4(String a) {
                this();
                System.out.println("Parameterized constructor);
        }
 }
```

super vs super()

Both are used in a subclass as a way to invoke or refer to its superclass.

- **super keyword** is used to call super class (parent class/ base class) variables and methods by the subclass object when they are overridden by subclass.

- **super()** is used to call super class constructor from subclass constructor.

```
public class SuperKeyword1 extends SuperKeyword{

    SuperKeyword1(){
        super(4);
        System.out.println("This is a child default constructor");
    }
```

We can use super() and this() only in constructor not anywhere else, any attempt to do so will lead to compile-time error. this() and super() are always have to be in first line within constructor and for that reason we **CANNOT** use them within same constructor.
We have to keep either super() or this() as the first line of the constructor but NOT both simultaneously.

20. How can you handle exceptions? Types of exceptions you faced in your project? What is the parent of all exceptions?

An Exception is a problem that can occur during the normal flow of an execution. Depending on situation, we can use try catch finally blocks.

In **try block:** Code that might throw some exception
In **catch block:** We define exception type to be caught and what to do if exception happens in TRY block code

Throwable class is parent of all Exceptions:

```
try {
    int a=10; int b=0;
        int c=a/b;
  }catch (ArithmeticException e) {
        System.out.println(e.getMessage());
  }
```

Types of Exception:

1. **Checked Exception -** are the exceptions that are checked at compile time.
   Example of checked exceptions:

- **ClassNotFoundException** - Class not found
- **InstantiationException** - Attempt to create an object of an abstract class or interface
- **FileNotFoundException -** Attempt to open file that doesn't exist or open file to write but have only reading permission

2. **Unchecked Exception -** are the exceptions that are not checked at compile time, they are Runtime Exceptions.

   Exception faced as part of java perspective:
   - **ArithmeticException** - Arithmetic error, such as divide-by-zero.
   - **ArrayIndexOutOfBoundsException** - Array index is out-of-bounds.
   - **NullPointerException** - Invalid use of a null reference.
   - **IllegalArgumentException** - Illegal argument used to invoke a method.

18. How many catch blocks can we have? Which catch block will get executed if u get ArithmeticException?

There can be any number of catch block for a single try block and it is not necessary that each try block must be followed by a catch block. It should be followed by either a catch block or a finally block.
However only the catch block encountered first on the call stack that satisfies the condition for the exception will be executed for that particular exception, rest will be ignored.

```
try {
            int a=10; int b=0;
            int c=a/b;
             }catch (ArithmeticException e) {
                    System.out.println(e.getMessage());
             }catch (Exception e) {
                    System.out.println(e.getMessage());
             }
```

Example of multiple catch blocks from current framework:

```
public static void initProperties(String filePath) {
      prop = new Properties();
      try {
```

```
                FileInputStream fis = new FileInputStream(filePath);
                prop.load(fis);
        } catch (FileNotFoundException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        }
 }
```

19. What is the difference between throw and throws?

Throws :

- is used to declare an exception, which means it works similar to the try-catch block.
- is used in method declaration.
- is followed by exception class names.
- you can declare multiple exception with throws
- throws declare at method it might throws Exception
- used to handover the responsibility of handling the exception occurred in the method to the caller method.

1 Example:

```
public void readPropFile() throws FileNotFoundException, IOException{
        Properties prop=new Properties();
        FileInputStream fis=new FileInputStream("fileNamePath.properties");
                        prop.load(fis);
 }
```

2 Example:

```
public class Test {
        public static void main(String[] args) throws InterruptedException {
                Test test = new Test();
                test.company();
        }


        void studentDetails() throws InterruptedException {
                System.out.println("Interview Prep has started");
                Thread.sleep(3000);
                System.out.println("Please work hard");
        }

    void codeinhub() throws InterruptedException {
            studentDetails();
        }

        void company() throws InterruptedException {
                codeinhub();
        }
}
```

Throw :

- is used in the method body to throw an exception
- throw is followed by an instance variable
- you cannot declare multiple exceptions with throw
- The throw keyword is used to handover the instance of the exception created by the programmer to the JVM manually.
- throw keyword is mainly used to throw custom exceptions.

Example

```
class Test {
        public static void main(String args[]) {
                        policyAge(15);
        }
        public static void policyAge(int age) {
                try {
                        if (age > 18) {
                                System.out.println("You are eligible for car insurance
                                policy");
                } else {
                                                                                }
                               throw new ArithmeticException("User is less than 18
                       years old and not eligible to have any insurance policy");
                } catch (Exception e) {
                         System.out.println(e.getMessage());
                }
}
```

20. What is the difference between final, finally and finalize?

Final keyword:

- **Used to apply restrictions on class, methods, and variable.**
- Used to declare constant values. The variable declared as final should be initialized only once and cannot be changed.
- Used to prevent inheritance. Java classes declared as final cannot be extended.
- Used to prevent method overriding. Methods declared as final cannot be overridden.

Example 1:

```
final int b=30;
b=37; //cannot change value of final variable
```

Example 2:

```
public final class Test {
        public static void main(String args[]) {
                System.out.println("I am parent");
        }
}

//you will get an error "Cannot subclass final class"
public class Child extends Test{
        public static void main(String[] args) {
                System.out.println("I am a child");
        }
}
```

Example 3:

```
public class Test {
        public final void testFinalKey() {
                System.out.println("Parent final method");
        }
}

//you will get an error “Cannot override the final method”

public class Child extends Test{
        public void testFinalKey() {
                System.out.println("Child final method");
        }
}
```

Finally block :

- The finally block **always** executes when the try block exits. This ensures that the finally block is executed even if an unexpected exception occurs.

```
try {
        Properties prop = new Properties();
        FileInputStream fis = new FileInputStream("FilePath");
        prop.load(fis);
} catch (Exception e) {
        System.out.println("I am an exception block");
} finally {
        System.out.println("I am final block");
}
        System.out.println("Running script after exception");
}
```

Finalize() method :

- finalize() is protected method of java.lang.object class and it is inherited to every class we create in java.
- finalize() method is used to perform some clean up operations on an object before it is removed from memory.

21. Explain Public Static void main (String args[])?

public: it is a access specified that means it will be accessible by any Class.
static: is a keyword to call this method directly using class name without creating an

object of it.
void: it is a return type i.e. it does not return any value.
main(): it is the name of the method which is searched by JVM as a
starting point for an application with a particular signature only.
it is the method where the main executions occurs.
string args[]: it's a command line argument passed to the main method.

22. What is the difference between interface and a class? Can we create an object for an interface? Why do we need Interface in test? Example from your framework?

Class :

- Class will contain concrete methods
- Class is extended
- We can create an Object of the class
- Class can inherit only one Class and can implement many interfaces

Interface :

- Interface will have Interface keyword.
- Interface will contain only abstract methods
- In java 8 → **Default Methods** are introduced which allow the interfaces to have methods with implementation without affecting the classes that implement the interface
- We cannot create object of interface
- Interface needs to be implemented
- Class can extend many interfaces
- We need to provide implementation to all methods when we implement interface to the class

Practical Example:

Basic statement we all know in Selenium is

```
WebDriver driver = new FirefoxDriver();
```

WebDriver itself is an Interface. We are I initializing Firefox browser using Selenium WebDriver. It means we are creating a reference variable (driver) of the interface (WebDriver) and creating an Object. Here WebDriver is an Interface and FirefoxDriver is a class.

Practical usage:

```
public class Listener implements ITestListener {

        @Override
        public void onTestStart(ITestResult result) {//for method
                System.out.println("Starting Test: "+ result.getName());
        }
        @Override
        public void onTestSuccess(ITestResult result) {
                System.out.println("Test case passed: "+ result.getName());
        }

        @Override
        public void onTestFailure(ITestResult result) {
                System.out.println("Test case failed: "+ result.getName());
        }
        @Override
        public void onTestSkipped(ITestResult result) {
                System.out.println("Test case skipped: "+ result.getName());
}
```

23. What is Access Modifiers (Private, public, protected)? How did you use them?

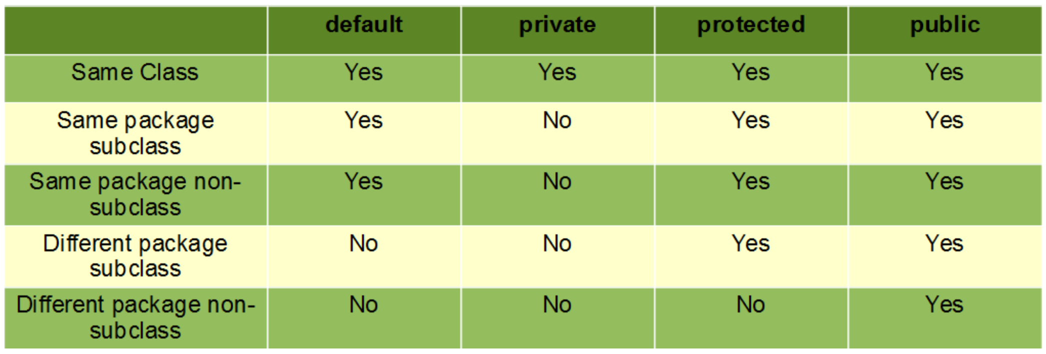
Java provides access modifiers to set access levels for classes, variables, methods and constructors.

public: A class or interface may be accessed from outside the package. Constructors, inner classes, methods and field variables may be accessed wherever their class is accessed.

protected: Accessed by other classes in the same package or any subclasses of same package or different package.

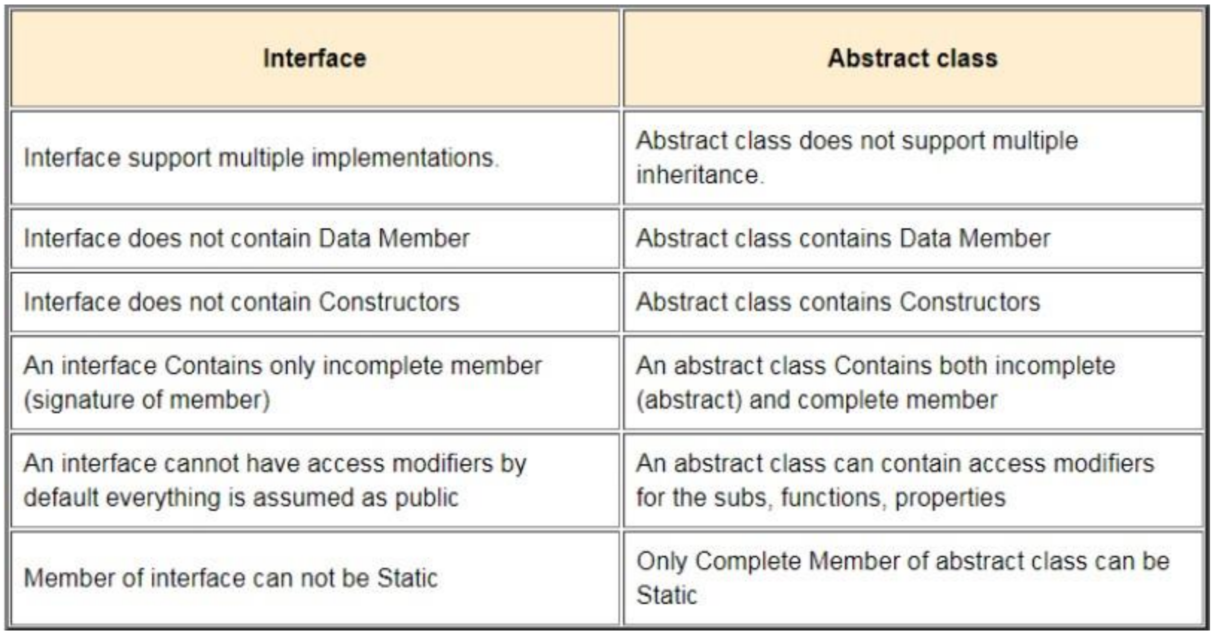
private: Accessed only within the class in which they are declared.

default: When no access modifier is specified for a class , method or data member – It is said to be having the default access modifier by default.



	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

24. Difference between abstract class and interface? When to use abstract class and interface in Java?



Interface	Abstract class
Interface support multiple implementations.	Abstract class does not support multiple inheritance.
Interface does not contain Data Member	Abstract class contains Data Member
Interface does not contain Constructors	Abstract class contains Constructors
An interface Contains only incomplete member (signature of member)	An abstract class Contains both incomplete (abstract) and complete member
An interface cannot have access modifiers by default everything is assumed as public	An abstract class can contain access modifiers for the subs, functions, properties
Member of interface can not be Static	Only Complete Member of abstract class can be Static

- An abstract class is good if you think you will plan on using inheritance since it provides a common base class implementation to derived classes.
- An abstract class is also good if you want to be able to declare non-public members. In an interface, all methods must be public.
- If you think you will need to add methods in the future, then an abstract class is a better choice. Because if you add new method headings to an interface, then all of the classes that already implement that interface will have to be changed to implement the new methods. That can be quite a hassle.

Practical Usage of Abstraction

In Page Object Model design pattern, we write locators (such as id, name, xpath etc.,) in a Page Class. We utilize these locators in tests but we can't see these locators in the tests. Literally we hide the locators from the tests.

25. Explain OOPS concepts? Is java 100% object oriented?

OOP concepts in Java are the main idea behind Java's Object Oriented Programming. They are an **abstraction, inheritance, polymorphism and encapsulation.**

Inheritance is a mechanism in which one object acquires all the properties and behaviors of parent object.
Polymorphism is the ability of an object to take on many forms.
Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.
Encapsulation is a mechanism of binding code and data together in a single unit.

No, **Java is not 100% object oriented**, since it has primitive data types, which are different from objects.

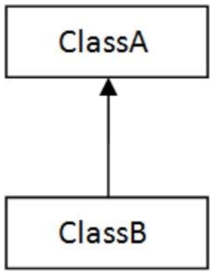
26. What is inheritance and benefits of it? Types of inheritance? How do you use it in your code?

Inheritance

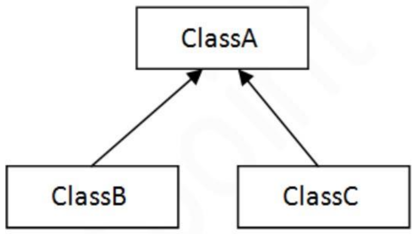
- The process of acquiring properties (variables) & methods (behaviors) from one class to another class is called inheritance.
- We are achieving inheritance concept by using extends keyword. Also known as is-a relationship.
- Extends keyword is providing relationship between two classes.
- The main objective of inheritance is code extensibility whenever we are extending the class automatically code is reused.

Types of Inheritance:

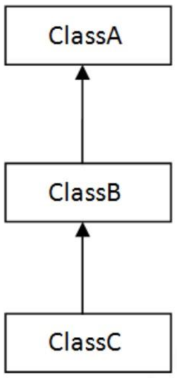
- **Single Inheritance** - single base class and single derived class.



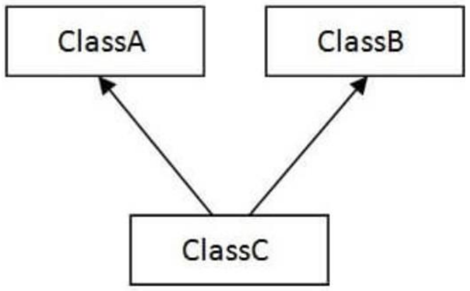
- **Hierarchical Inheritance** - when a class has more than one child classes (sub classes)



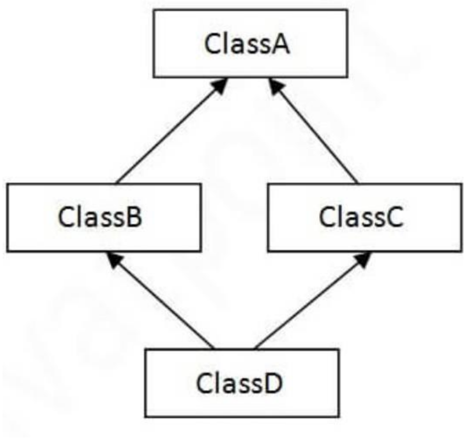
- **Multilevel Inheritance** - single base class, single derived class and multiple intermediate base classes.



- **Multiple Inheritance** - multiple classes and single derived class (Possible through interface only)



- **Hybrid Inheritance** - combination of both Single and Multiple Inheritance (Possible through interface only)



Usage of inheritance in real time project

In our current Cucumber framework we have BaseClass where we initialize and read Property file, initialize WebDriver interface. And after we extend the Base Class in other classes such as Pages, Tests and Utility Class.

27. What is polymorphism? Types of polymorphism?

Polymorphism is the ability of an object to take on many forms. Polymorphism allows us to perform a task in multiple ways.

Combination of overloading and overriding is known as Polymorphism.

There are two types of Polymorphism in Java

1. **Compile time polymorphism (Static binding) – Method overloading**
2. **Runtime polymorphism (Dynamic binding) – Method overriding**

28. Method overloading & overriding? How do you use it in your framework? Any example or practical usage of Run time polymorphism?

Method overloading in Java occurs when two or more methods in the same class have the exact same signature (name) but different parameters (remember that method parameters accept values passed into the method).
Overloading: Same method name with different arguments/parameters **in same class**

Practical Example

```
public static void selectValue( WebElement element, String value)
        { Select obj = new Select(element);
        obj.selectByVisibleText(value);
}

public static void selectValue( WebElement element, int index)
        { Select obj = new Select(element);
        obj.selectByIndex(index);
}

public static void switchToFrame( WebElement element)
        { driver.switchTo().frame(element);
}
public static void switchToFrame(int index) {
        driver.switchTo().frame(index);
}

public static void switchToFrame(String name) {
        driver.switchTo().frame(name);
}
```

Method overriding

Declaring a method in child class which is already present in the parent class is called Method Overriding. In simple words, overriding means to override the functionality of an existing method.
With method overriding a child class can give its own specific implementation to an inherited method without modifying the parent class method. Assume we have multiple child classes. In case one of the child classes want to use the parent class method and other class want to use their own implementation then we can use overriding feature.

Practical Usage:

1. Implementation of iTestListener interface.

```
public class Listener implements ITestListener {
```

```
                @Override

                public void onTestStart(ITestResult result) {//for method
                        System.out.println("Starting Test: "+ result.getName());
                }

                @Override
                public void onTestSuccess(ITestResult result) {
                        System.out.println("Test case passed: "+ result.getName());
                }
                @Override
                public void onTestFailure(ITestResult result) {
                        System.out.println("Test case failed: "+ result.getName());
                }
                @Override
                public void onTestSkipped(ITestResult result) {
                        System.out.println("Test case skipped: "+ result.getName());
 }
```

2. Selenium WebDriver provides an interface WebDriver, which consists of abstract methods getDriver() and closeDriver(). So any implemented class with respect to browser can override those methods as per their functionality, like ChromeDriver implements the WebDriver and can override the getDriver() and closeDriver().

29. Can we override/overload main method? Explain the reason? Can you override static method? Can we overload and override private method?

We cannot override static method, so we cannot override main method. However, you can overload main method in Java. But the program doesn't execute the overloaded main method when you run your program; you have to call the overloaded main method from the actual main method.
Practically I do not see any use of it and we don’t use it in my framework.

```
        public class MainMethodOverload {
                public static void main(String[] args) {
                        main(5);//if comment this line nothing will get executed
                }
                public static void main(int r) {
                        System.out.println("Hello");

                }
        }
```

Static methods are bounded with class **it is not possible to override static methods**

```
class Parent {
        static void m1() {
                System.out.println("parent m1()");
        }
}

class Child extends Parent {
        static void m1() {
                System.out.println("child m1()");
        }

        public static void main(String[] args) {
                Parent p = new Child();
                p.m1();
        }
}
```

In java not possible to override private methods because these methods are specific to classes, not visible in child classes.

30. How does method override differ from abstraction and inheritance?

Override

When methods with same name and arguments present in child and parents class and class inheriting the method from its superclass (parent class) has the option to override it. Benefit of overriding is the ability to define behavior specific to particular class. Method Overriding is possible only by inheritance.

Inheritance is a process where one class inherits the properties of another class or simply, we cans say that extending one class into other class is known as Inheritance.

Abstraction is the methodology of hiding the implementation of internal details and showing the functionality to the users.

31. **Can we achieve 100% abstraction in JAVA? Can we achieve 100% abstraction in JAVA with use of the interfaces?**

We cannot achieve 100% abstraction in JAVA unless we use Interfaces

32. **What is encapsulation?**

It is the technique of making the fields in a class private and providing access to the fields via public methods. If a field is declared private, it cannot be accessed by anyone outside the class, thereby hiding the fields within the class. Therefore encapsulation is also referred to as data hiding. The main benefit of encapsulation is the ability to modify our implemented code without breaking the code of others who use our code. With this Encapsulation gives maintainability, flexibility and extensibility to our code.

Practical usage:

Partial example of Encapsulation we can achieve in our framework through POM classes, where we declare the data members as private and initialization of data members will be done using Constructor to utilize those in methods.

33. **What is singleton and have used singleton concept in your project ?**

A singleton class is a class that can have only one object (an instance of the class) at a time. After first time, if we try to instantiate the Singleton class, the new variable also points to the first instance created. So whatever modifications we do to any variable inside the class through any instance, it affects the variable of the single instance created.

- Singleton pattern restricts the instantiation of a class and ensures that only one instance of the class exists in the java virtual machine.
- The singleton class must provide a global access point to get the instance of the class.
- Singleton pattern is used for logging, drivers objects

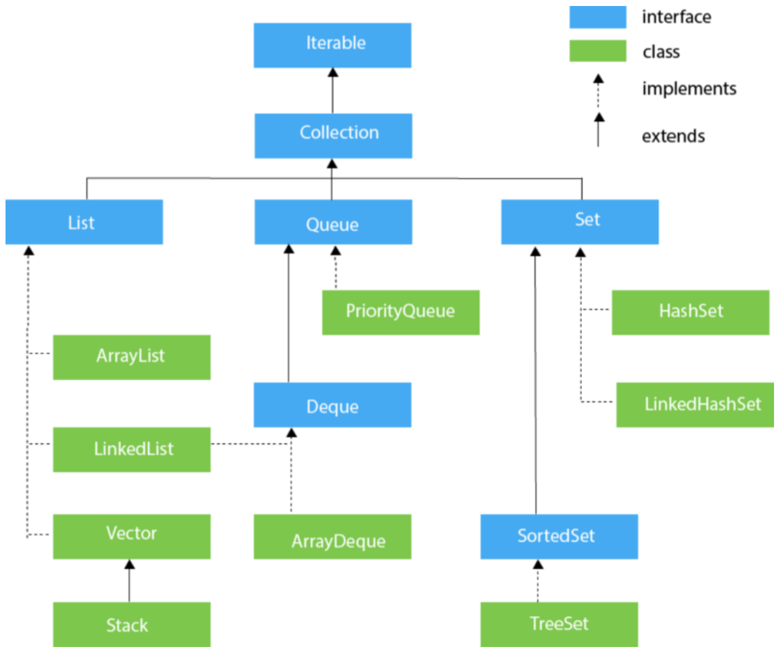
Example:

```
public class SingletonExample {
        //static member holds only one instance of the singleton class

        private static SingletonExample singletonInstance;

        //creating private constructor to prevent instantiation
        private SingletonExample(){
        }
        //create public method to return instance of the class
        public static SingletonExample getInstance() {
                singletonInstance=new SingletonExample();
                return singletonInstance;
        }
}
```

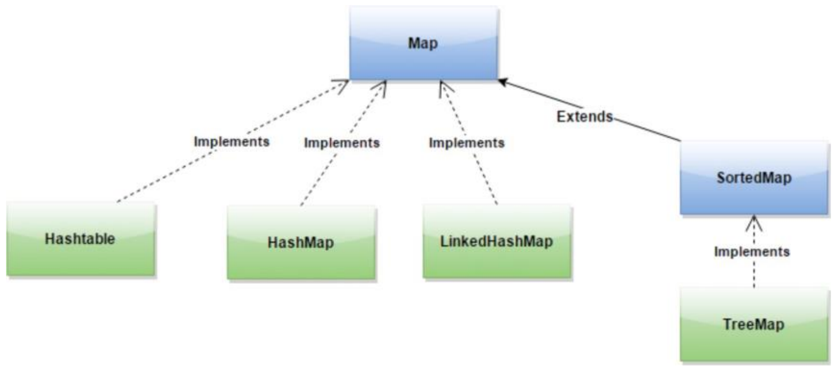
In my current project I do not use the concept of singleton class.

34. What is collection in Java and what type of collections have you used?

Java provides Collection Framework which defines several classes and interfaces to represent a group of objects as a single unit.



Maps are not part of collection but built based on the collection concepts



Mostly in my current project we use ArrayList and HashMap.

35. What is array and Arraylist (List)? Difference between them?

- Arrays are fixed in size but ArrayLists are dynamic in size.
- Arrays can store homogeneous elements whereas collections can store both. Example: in Array we can store either int or String or boolean whereas in Array list we can store all of them together
- To find the size on an Array we use ArrayName.length and for arrayList we use ArrayListName.size()

ArrayList:

```
ArrayList names = new ArrayList();
names.add("Daniela");
names.add("Patrick");
```

How to print all values from arrayList
//1. using for loop

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

//2. use advance for loop/enhanced/for each loop

```
for (String value: names) {
        System.out.println(value);
}
```

```
//3 using Iterator

Iterator<String> it=names.iterator();//create/initialize Iterator
while(it.hasNext()) {
        String name=it.next();
        System.out.println(name);
}

//4 using while loop

int count=0;
while(names.size()>count) {
        System.out.println(names.get(count));
        count++;
}
```

Array:

```
String[] array=new String[3];
array[0]="Jordan";
array[1]="Jack";
array[2]="Jack";

for(int i=0; i<array.length; i++) {
        System.out.println(array[i]);
}
```

36. Difference between ArrayList vs LinkedList?

ArrayList and LinkedList, both implements List interface and provide capability to store and get objects as in ordered collections. Both are non-synchronized classes and both allow duplicate elements.

ArrayList

- ArrayList internally uses a dynamic array to store the elements.
- Manipulation with ArrayList is slow because it internally uses an array. If any element is removed from the array, all the bits are shifted in memory.

- ArrayList is better for storing and accessing data.

LinkedList

- LinkedList internally uses a doubly linked list to store the elements (consist on value + pointer to previous node and pointer to next node)
- Manipulation with LinkedList is faster than ArrayList because it uses a doubly linked list, so no bit shifting is required in memory.
- LinkedList is better for manipulating data.

```
//Create linked list
LinkedList linkedList = new LinkedList();

//Add elements
linkedList.add("A");
linkedList.add("B");

System.out.println(linkedList);

//Add elements at specified position
linkedList.add(2, "C");
linkedList.add(3, "D");

System.out.println(linkedList);

//Remove element
linkedList.remove("A");    //removes A
linkedList.remove(0);      //removes B

System.out.println(linkedList);
```

Bonus:

ArrayList vs Vector?
Both implement List Interface and maintains insertion order

ArrayList - is not synchronized, so it is fast.
Vector - is synchronized, so it is slow.

37. Difference between HashSet vs HashMap ?

HashSet

1. HashSet class implements Set interface
2. In HashSet, we store objects(elements or values).
3. HashSet does not allow duplicate elements that mean you cannot store duplicate values in HashSet.
4. HashSet permits to have a single null value.
5. HashSet is not synchronized.

```
//create HashSet
HashSet hset = new HashSet();
hset.add("BMW");
hset.add(2018);
```

How to print all values from hashSet

```
// 1. advance loop
for (Object obj : hset) {
        System.out.println(obj);
}

// 2. using iterator
Iterator itr = hset.iterator();
while (itr.hasNext()) {
        Object words = itr.next();
        System.out.println(words);
}
```

HashMap

1. HashMap class implements the Map interface
2. HashMap is used for storing Key, Value paired objects.
3. HashMap does not allow duplicate keys however it allows having duplicate values.
4. HashMap permits single null key and any number of null values.
5. HashMap is not synchronized.

Bonus:

■

ArrayList vs HashSet?

Both ArrayList and HashSet are non synchronized collection class
Both ArrayList and HashSet can be traversed using Iterator

ArrayList

- ArrayList implements List interface
- ArrayList allows duplicate values
- ArrayList maintains the order of the object in which they are inserted
- In ArrayList we can add any number of null values
- ArrayList is index based

HashSet

- HashSet implements Set interface
- HashSet doesn't allow duplicates values
- HashSet is an unordered collection and doesn't maintain any order
- HashSet allow one null value
- HashSet is completely object based

38. What is Map/ HashMap? How did you use it in your framework?

Java Map Interface. A map contains values on the basis of key, i.e. key and value pair. Each key and value pair is known as an entry. A Map Contains unique keys. A Map is useful if we have to search, update or delete elements on the basis of a key.

In our current Cucumber framework we work with map object whenever we use cucumber tables.
Also another example: whenever we store external data in excel we bring data in a test in a form of key and value pair.

In framework we have verify month name using HashMap, Number in Key from 1 to 12 and respective number has month name.

39. Difference HashTable vs HashMap ?

Both **HashMap** and **Hashtable** implement Map Interface

HashMap

- HashMap is non synchronized, so it is not-thread safe
- HashMap is fast
- HashMap allows one null key and multiple null values

```
HashMap <Integer, String> hmap=new HashMap<Integer, String>();

hmap.put(1, "January");
hmap.put(1, "January");
hmap.put(2,"February");
hmap.put(null, "February");
hmap.put(null, "February");

for (Map.Entry obj: hmap.entrySet()) {
        System.out.println(obj.getKey()+" "+obj.getValue());
}
```

Output:

```
null February
1 January
2 February
```

Hashtable

- Hashtable is synchronized, so it is thread-safe
- Hashtable is slow
- Hashtable doesn't allow any null key or value

```
Hashtable <Integer, String> htable=new Hashtable<Integer, String>();

htable.put(1, "January");
htable.put(1, "January");
htable.put(2, "February");
htable.put(null, "February");
htable.put(null, "February");


for (Map.Entry obj: htable.entrySet()) {
        System.out.println(obj.getKey()+" "+obj.getValue());
}
```

Output:

```
Exception in thread "main" java.lang.NullPointerException
```

■

```
at java.util.Hashtable.put(Hashtable.java:464)
```

40. What is garbage collector and how to call Garbage Collector?

Garbage collection is the process of looking at heap memory and identifying which objects are in use and which are not and deleting unused objects. Once object is created it uses some memory and the memory remains allocated till there are references for the use of the object. When there are no references to an object, it is assumed to be no longer needed. There is no explicit need to destroy an object as Java handles the deallocation automatically by using Garbage Collection process.
Garbage collection in Java happens automatically during the lifetime of the program.
Garbage collection in java can not be enforced. But still sometimes, we call the System.gc( ) method explicitly. System.gc() method provides just a "hint" to the JVM that garbage collection should run. It is not guaranteed!

41. What is java regular expression?

Regular Expressions or Regex (in short) is an API for defining String patterns that can be used for searching, manipulating and editing a text.
Regular Expressions are provided under java.util.regex package.

42. What the difference is between wait and sleep in java?

Both wait() and sleep() methods are used to pause the execution of current thread for some period of time.

sleep()

- is a method which is used to pause the process for few seconds or the time we want to
- usage - just to put thread on sleep for time-synchronization

wait()

- method, thread goes in waiting state and it won't come back automatically until we call the notify() or notifyAll().
- usage - is normally done on condition, Thread wait until a condition is true so we use it for for multi-thread-synchronization.

43. What is the output for this program?

```
for (int i = 0; i < 3; i++) {
    for (int j = 3; j >= 0; j--) {
        if (i == j)
        continue;
        System.out.println(i + " " + j);
    }
}
```

The continue keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop. So the output of the program will be:

```
0 3
0 2
0 1
1 3
1 2
1 0
2 3
2 1
2 0
```

44. Here is the arrayList, how can I remove all duplicates from it?

```
List<String> al = new ArrayList<String>();
     al.add("Ajay");
    al.add("Becky");
    al.add("Chaitanya");
    al.add("Ajay");
    al.add("Rock");
    al.add("Becky");
```

```
HashSet hs=new HashSet();
for (int i=0; i<al.size(); i++) {
    hs.add(al.get(i));
}
System.out.println(hs);
```

45. What is the output of the following program?

```
class Parent{
      m1(){
                System.out.println("In parent class m1");
      }
}

class Subclass extends parent{
       m1(){
                System.out.println("In child class m1");
      }

      m2(){
                System.out.println("In m2");
      }

public static void main(String args[]){

      Parent obj= new Subclass();
      obj.m1();
      obj.m2();
}
}
```

Program won't run !

```
public static void main(String args[])
      Parent obj= new Subclass();
      obj.m1(); - will give child output ("In child class m1")
      obj.m2(); - this method won't be accessible (child class object is referred by
```
parent class reference variable)

==================================

What is Multithreading?

Multithreading refers to two or more tasks executing concurrently within a single program.
A **thread** is an independent path of execution within a program. Many**threads** can run concurrently within a program. Every **thread in Java** is created and controlled by the **java**.lang.**Thread** class.

Thread-safe code is code that will work even if many **Threads** are executing it simultaneously. A piece of code is **thread-safe** if it only manipulates shared data structures in a manner that guarantees **safe** execution by multiple **threads** at the same time.