**INTERVIEW PREPRATION**

Site = full stack cafe

Link=https://snowdream.github.io/115-Java-Interview-Questions-and-Answers/115-Java-Interview-Questions-and-Answers/en/rmi.html

Que) Tell me about yourself?

Ans :- MY name is Shubhanshu Rajneesh jain ,I am from a city which is famous for it’s **corn ,** so it is also known as corn city , chhindwara .

I live in a joint family, there are total 22 members living together happily. My father occupation is Advocate. My mother is housewife. I have one elder brother, he is doing job in Bangalore.

I had completed my 10th from saket public school, gondia maharashta in 2013 with 9.4 cgpa.

I had completed my 12th from First step higher secondary school chhindwara in 2015 with 79.6%

I am have pursued BE in computer science and engineering from rajiv Gandhi college of engineering and research, Nagpur in 2020 .

My hobbies are:-

Learning new things like new Languages, different recipe.

Ready to go travelling & exploring new area.

Active in playing carom, cricket.

Interacting with people.

Ques) storage class in c?

Ans:- there ares 4 types of storage class.

1. Auto 2.Register 3.Static 4.Extern.

Ques) how to change two variable without using third variable?

Ans :- logic

Int a=5 ,b=6;

A=a+b; (11)

B=a-b; (5)

A=a-b; (6)

Ques) explain abs and fabs?

Ans:- abs( ) function in C returns the absolute value of an integer. The absolute value of a number is always positive. Only integer values are supported in C.

* “stdlib.h” header file supports abs( ) function in C language. Syntax for abs( ) function in C is given below.

**int abs ( int n );**

int m = abs(200); // m is assigned to 200

int n = abs(-400); // n is assigned to -400

printf("Absolute value of m = %d\n", m);

printf("Absolute value of n = %d \n",n);

#### ****OUTPUT:****

|  |
| --- |
| Absolute value of m = 200 Absolute value of n = 400 |

**Fabs :-**

The fabs() function takes a single argument (in double) and returns the absolute value of that number (also in double).

1. x = -1.5;
2. result = fabs(x);
3. printf("|%.2lf| = %.2lf\n", x, result);
4. x = 11.3;
5. result = fabs(x);
6. printf("|%.2lf| = %.2lf\n", x, result);

**Output**

|-1.50| = 1.50

|11.30| = 11.30

|0.00| = 0.00

Ques) to find the biggest of the three number?

Ans: logic

If(a>=b && a>=c)

Print =a

If(b>=c && b>=a)

Print =b

If(c>=a &&c>=b)

Print =c

Ques)to find leap year?

Ans:- logic

Int x;

If(x%400==0)

{ Printf(“leap year”);

}

Else

{ If(x%100==0)

{ Printf(“not leap year”);

}

Else

{ If(x%4==0)

{ Printf(“ leap year”);

}

Else

Printf(“ not leap year”);

Q. output of the program is

#include <stdio.h>

int main()

{

int a=2,b=5,c=2;

if(a<b<c)

printf("b is between a and c");

else

printf("b is not between a and c");

return 0;

}

Ans:- **b is between a and c**

Q. output of the program is

#include <stdio.h>

int main()

{

int i;

for(i=0;i<5;i++)

{

int i=10;

printf("%d",i);

}

return 0;

}

Ans:- **1010101010**

Q. output of the program is

#include <stdio.h>

int main()

{

int i;

for(i=0;i<5;i++)

{

i=10;

printf("%d",i);

}

return 0;

}

**Ans:- 10**

Q. what is link list explain?

Ans=A linked list is a linear data structure, in which the elements are not stored at contiguous memory locations. The elements in a linked list are linked using pointers as shown in the below image:  


In simple words, a linked list consists of nodes where each node contains a data field and a reference(link) to the next node in the list.

**Q)Why Linked List?**  
Ans=Arrays can be used to store linear data of similar types, but arrays have the following limitations.  
**1)** The size of the arrays is fixed: So we must know the upper limit on the number of elements in advance. Also, generally, the allocated memory is equal to the upper limit irrespective of the usage.  
**2)** Inserting a new element in an array of elements is expensive because the room has to be created for the new elements and to create room existing elements have to be shifted.

For example, in a system, if we maintain a sorted list of IDs in an array id[].

id[] = [1000, 1010, 1050, 2000, 2040].

And if we want to insert a new ID 1005, then to maintain the sorted order, we have to move all the elements after 1000 (excluding 1000).  
Deletion is also expensive with arrays until unless some special techniques are used. For example, to delete 1010 in id[], everything after 1010 has to be moved.

**Advantages over arrays**  
**1)** Dynamic size  
**2)** Ease of insertion/deletion

**Drawbacks:**  
**1)** Random access is not allowed. We have to access elements sequentially starting from the first node. So we cannot do binary search with linked lists efficiently with its default implementation. Read about it [here](https://www.geeksforgeeks.org/binary-search-on-singly-linked-list/).  
**2)** Extra memory space for a pointer is required with each element of the list.  
**3)** Not cache friendly. Since array elements are contiguous locations, there is locality of reference which is not there in case of linked lists.

Ques :- Trim() method in java?

# Ans :- Trim (Remove leading and trailing spaces) a string in Java

Given a string, remove all the leading and trailing spaces from the string and return it.

Examples:

Input : str = " Hello World "

Output : str = "Hello World"

Input : str = " Hey there Joey!!! "

Output : str = "Hey there Joey!!!"

* We can eliminate the leading and trailing spaces of a string in Java with the help of **trim()**.
* trim() method is defined under the String class of java.lang package.
* It does not eliminated the middle spaces of the string.
* By calling the trim() method, a new String object is returned.
* It doesn’t replace the value of String object. Therefore if we want the access to the new String object, we just need to reassign it to the old String or assign it to a new variable.

Ques:- purpose of jdk ,jvm & jre in java?

Ans:-

1. **JDK**

Java Development Kit aka JDK is the core component of Java Environment and provides all the tools, executables, and binaries required to compile, debug, and execute a Java Program.

JDK is a platform-specific software and that’s why we have separate installers for Windows, Mac, and Unix systems.

We can say that JDK is the superset of JRE since it contains JRE with Java compiler, debugger, and core classes.

### JVM

### JVM is the heart of Java programming language. When we execute a Java program, JVM is responsible for converting the byte code to the machine-specific code.

JVM is also platform-dependent and provides core java functions such as memory management, garbage collection, security, etc.

JVM is customizable and we can use java options to customize it. For example, allocating minimum and maximum memory to JVM.

JVM is called **virtual** because it provides an interface that does not depend on the underlying operating system and machine hardware.

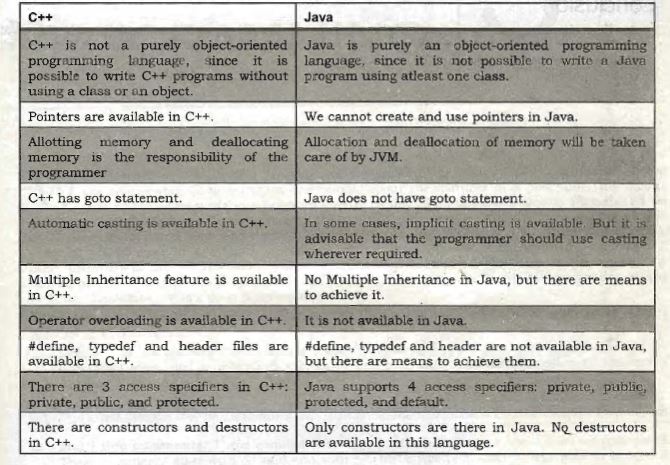
This independence from hardware and the operating system makes java program write-once-run-anywhere.

### 3. JRE

JRE is the implementation of JVM. It provides a platform to execute java programs. JRE consists of JVM, Java binaries, and other classes to execute any program successfully.

JRE doesn’t contain any development tools such as Java compiler, debugger, JShell, etc.

If you just want to execute a java program, you can install only JRE. You don’t need JDK because there is no development or compilation of java source code is required.



Ques :-without jdk will our program compile or not?

Ans :- no our program will not compile.

Ques :- if our system has only jre than?

Ans :- it will execute our program but it will not compile our program.

Yes, you can execute Java program with out JDK. But to do that you need JVM. JDK is basically used to compile your code. Once you have .class file of your .java program you don’t need JDK. You simply run your code with JVM.

**Ques :- what is native library?**

**Ans:**  A native library is a library that contains "native" code. That is, code that has been compiled for a specific hardware architecture or operating system such as x86 or windows

"Native Library" generally means a non-Java library that's used by the system (so C/C++, etc). Think normal DLLs or libs.

**PROGRAM TO REVERSE A STRING**

**IN JAVA**

**1.**

import java.lang.\*;

import java.io.\*;

import java.util.\*;

// Class of ReverseString

class ReverseString {

    public static void main(String[] args)

    {

        String input = "GeeksForGeeks";

        // convert String to character array

        // by using toCharArray

        char[] try1 = input.toCharArray();

        for (int i = try1.length - 1; i >= 0; i--)

            System.out.print(try1[i]);

    }

}

**2.**

import java.lang.\*;

import java.io.\*;

import java.util.\*;

public class Test {

    public static void main(String[] args)

    {

        String str = "Geeks";

  // conversion from String object to StringBuffer

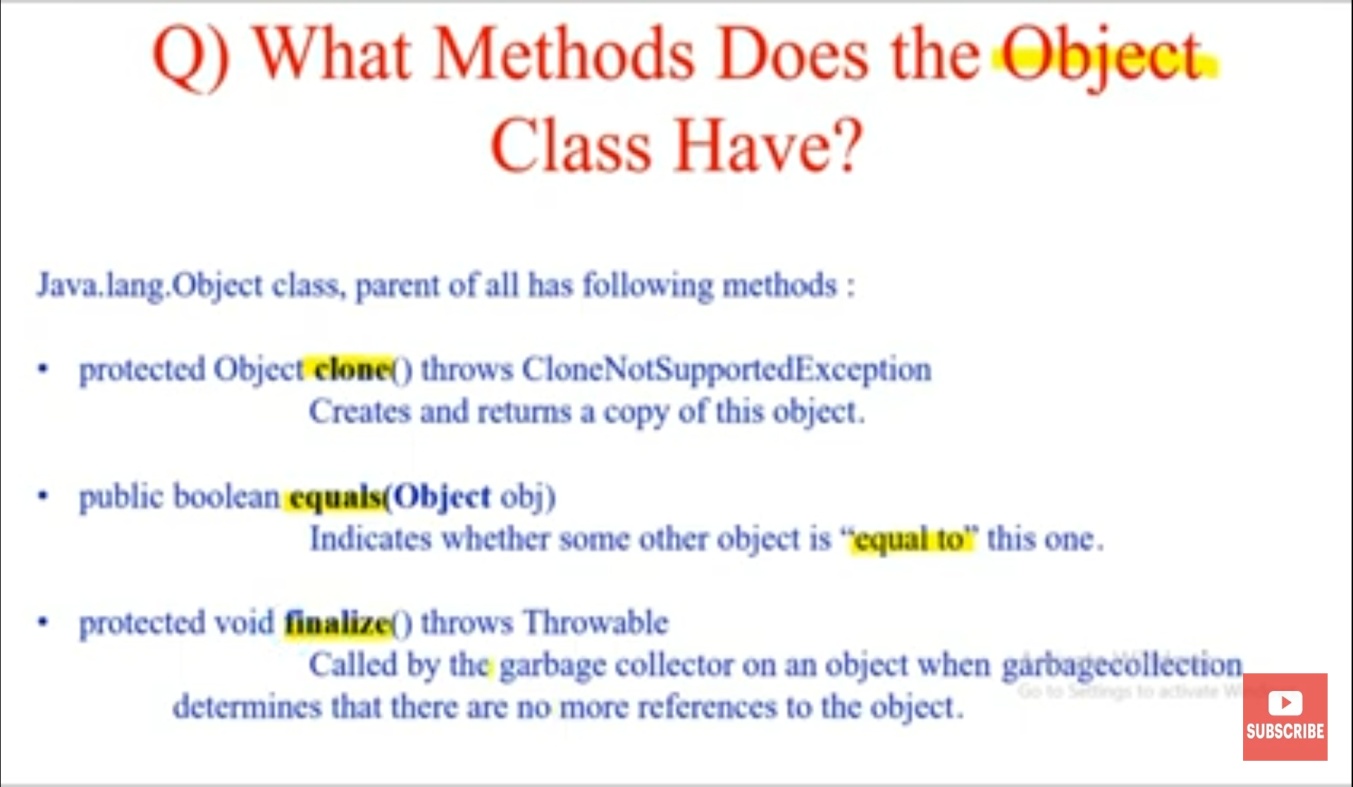
        StringBuffer sbr = new StringBuffer(str);

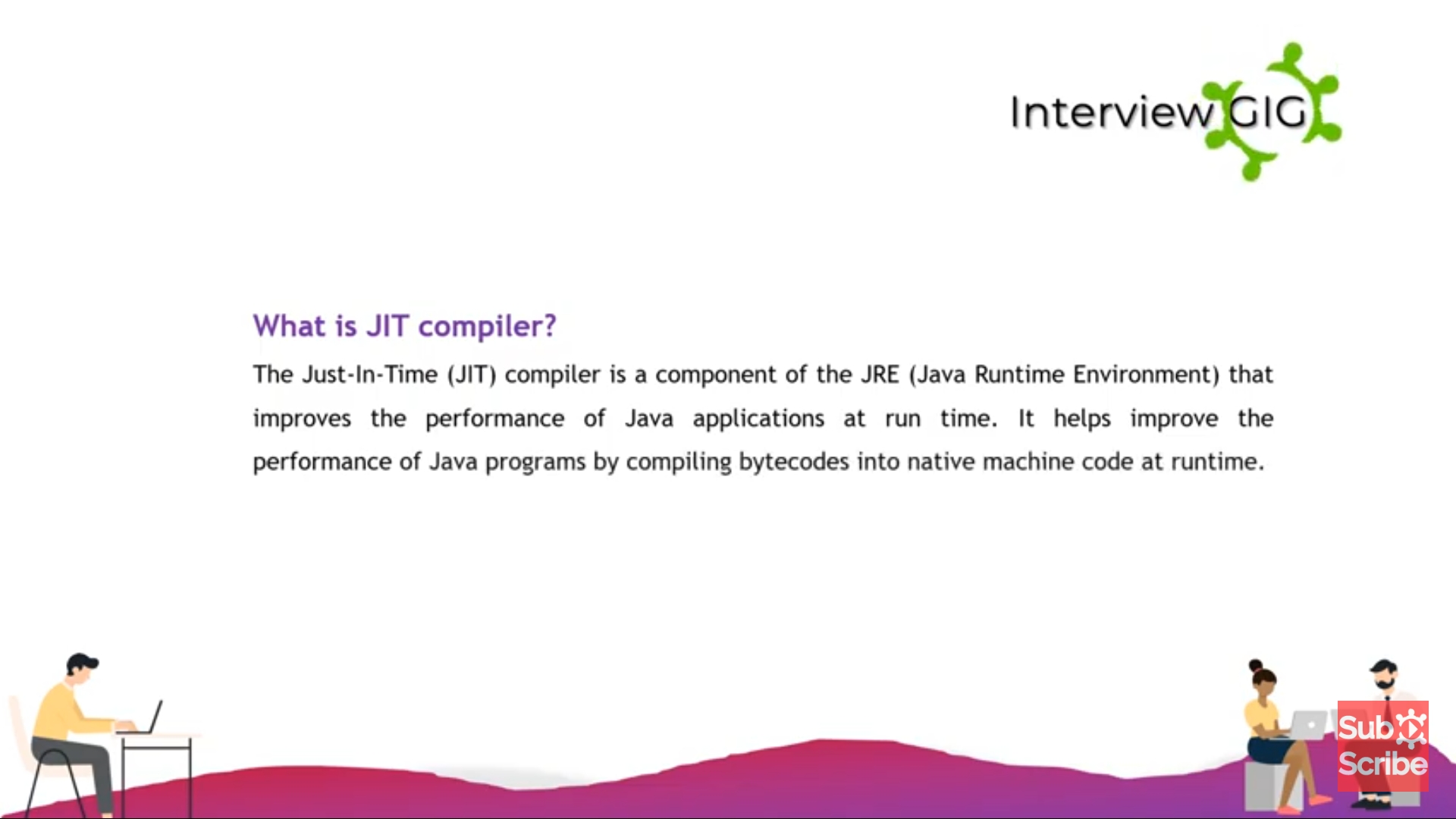
        // To reverse the string

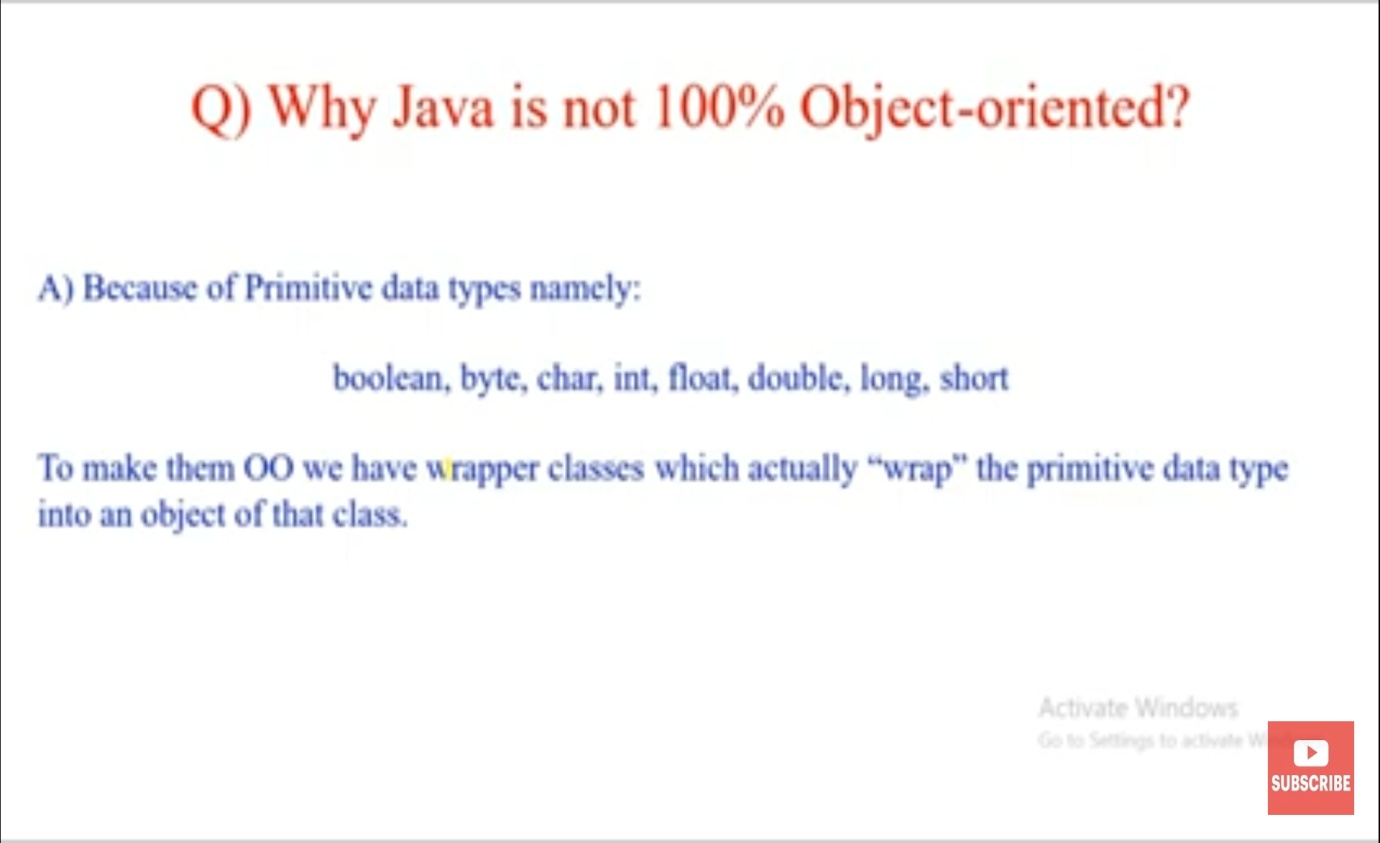
        sbr.reverse();

        System.out.println(sbr);

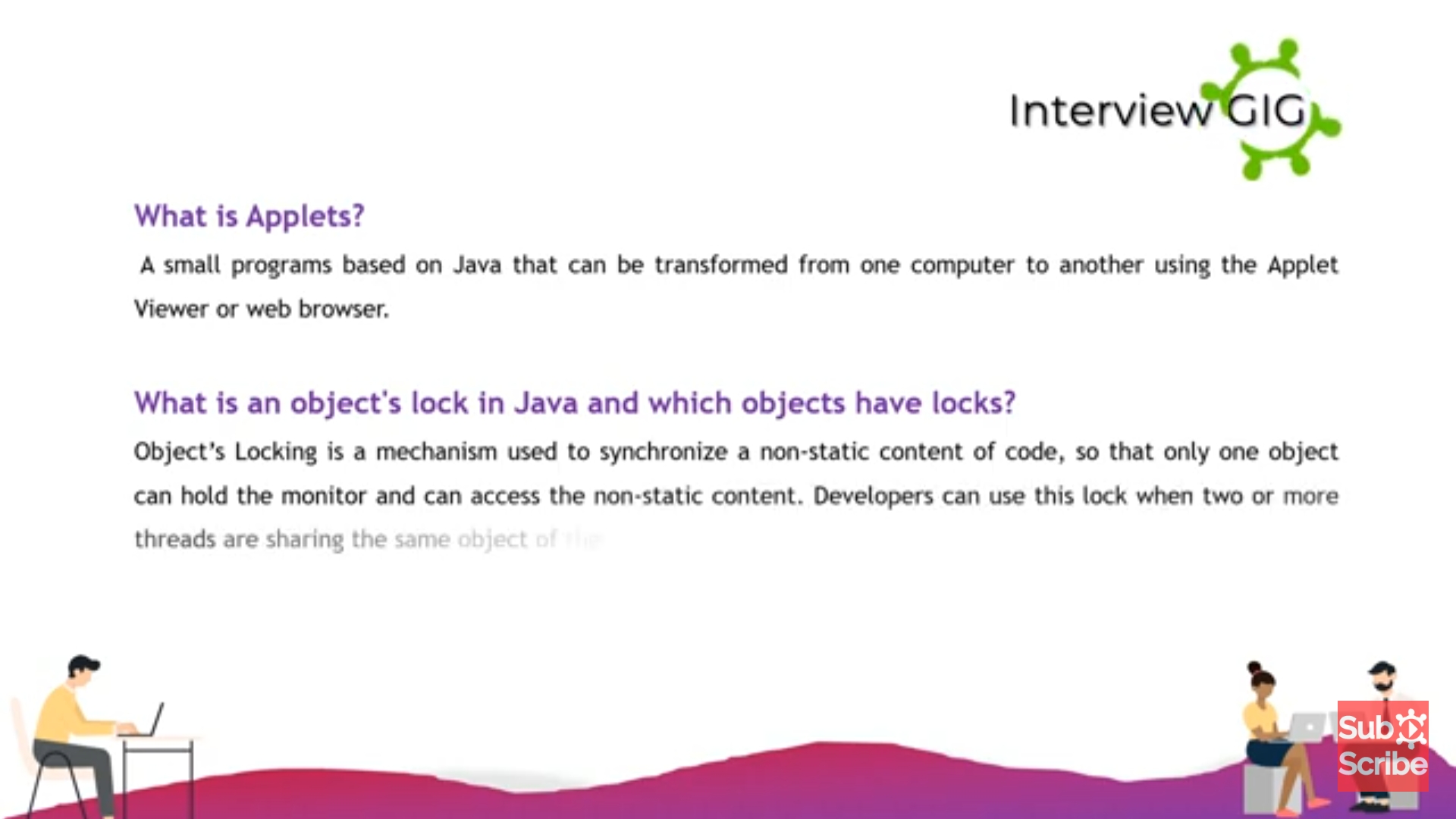
    }

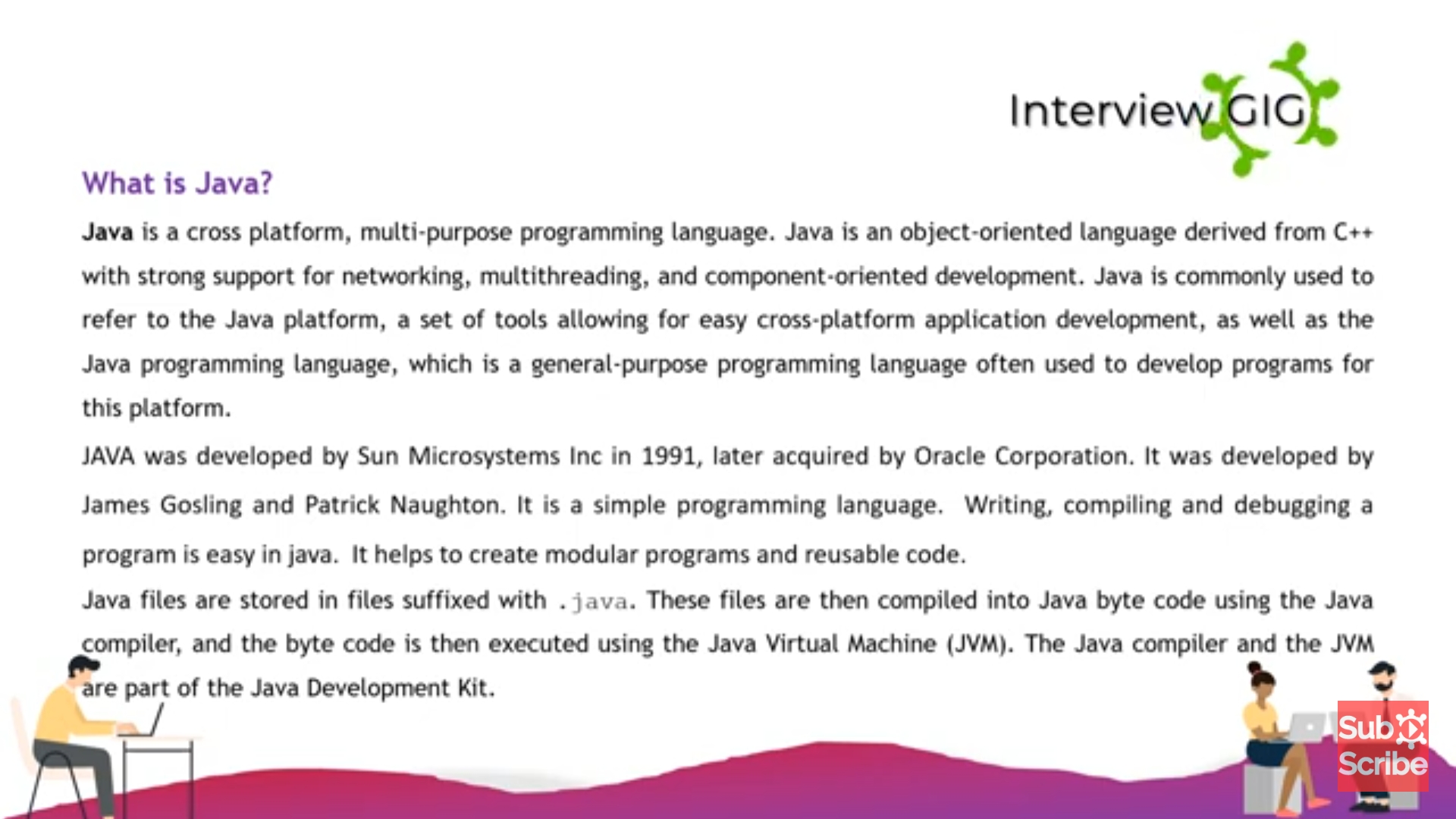


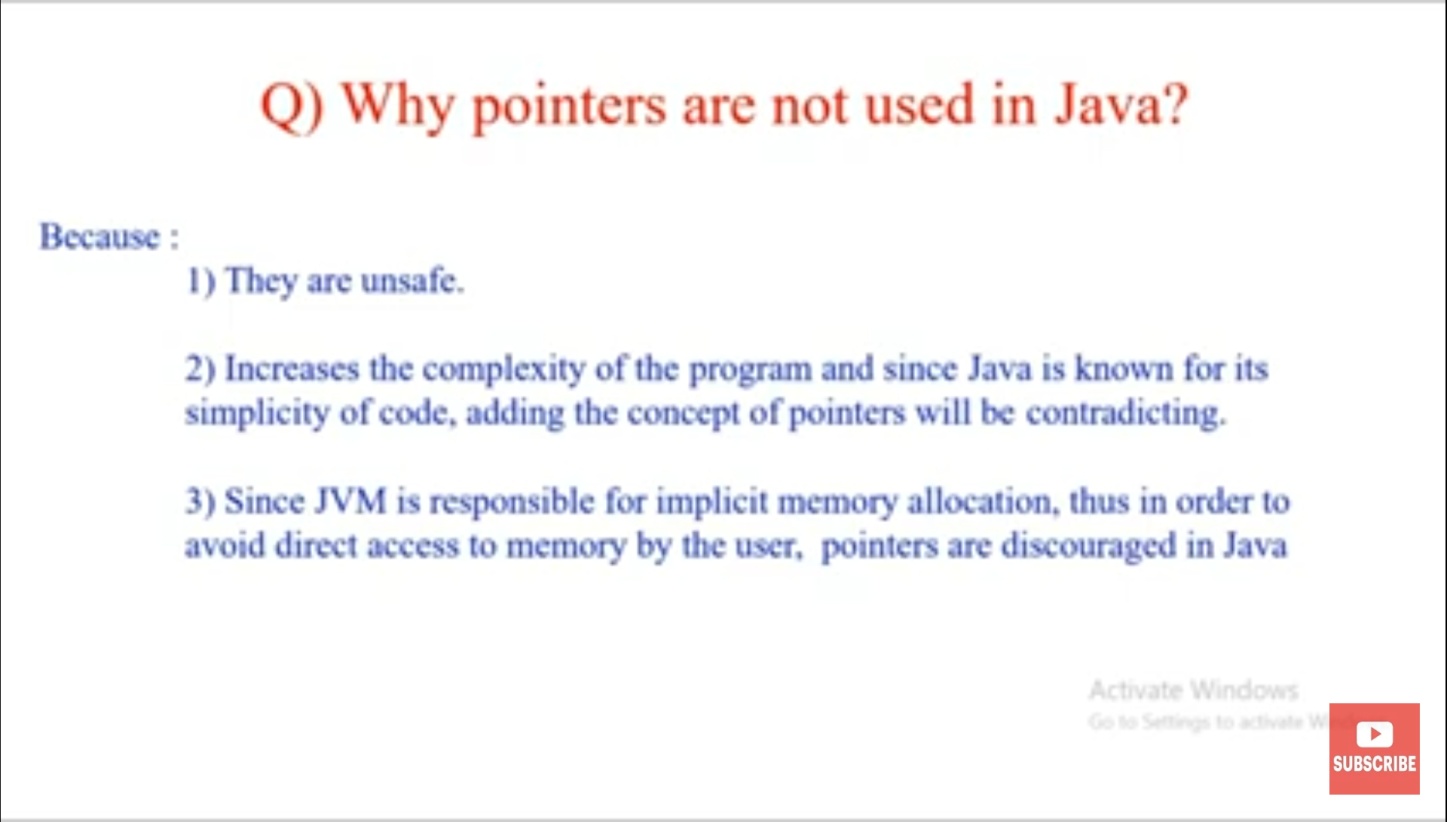




This primitive data type we use while programming in java & hence java is not object oriented programming language .because in oop everything should be around object & primitive data type is not object . more over java people develop WRAPPER class .to make them object oriented . we have Wrapper classes which actually “wrap” the primitive data type into an object of that class.







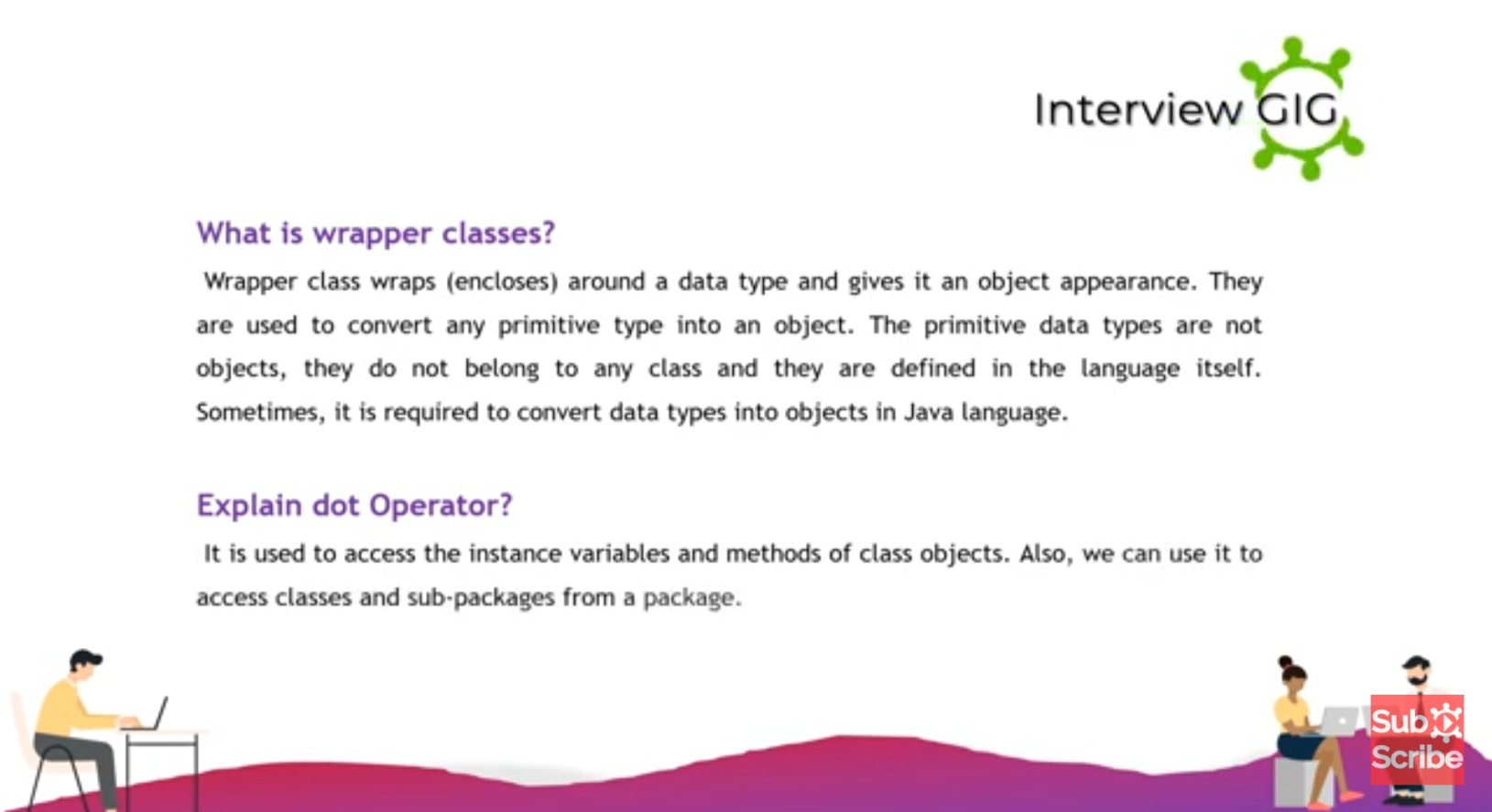
1. Since JVM is responsible for memory allocation & deallocaton in java.

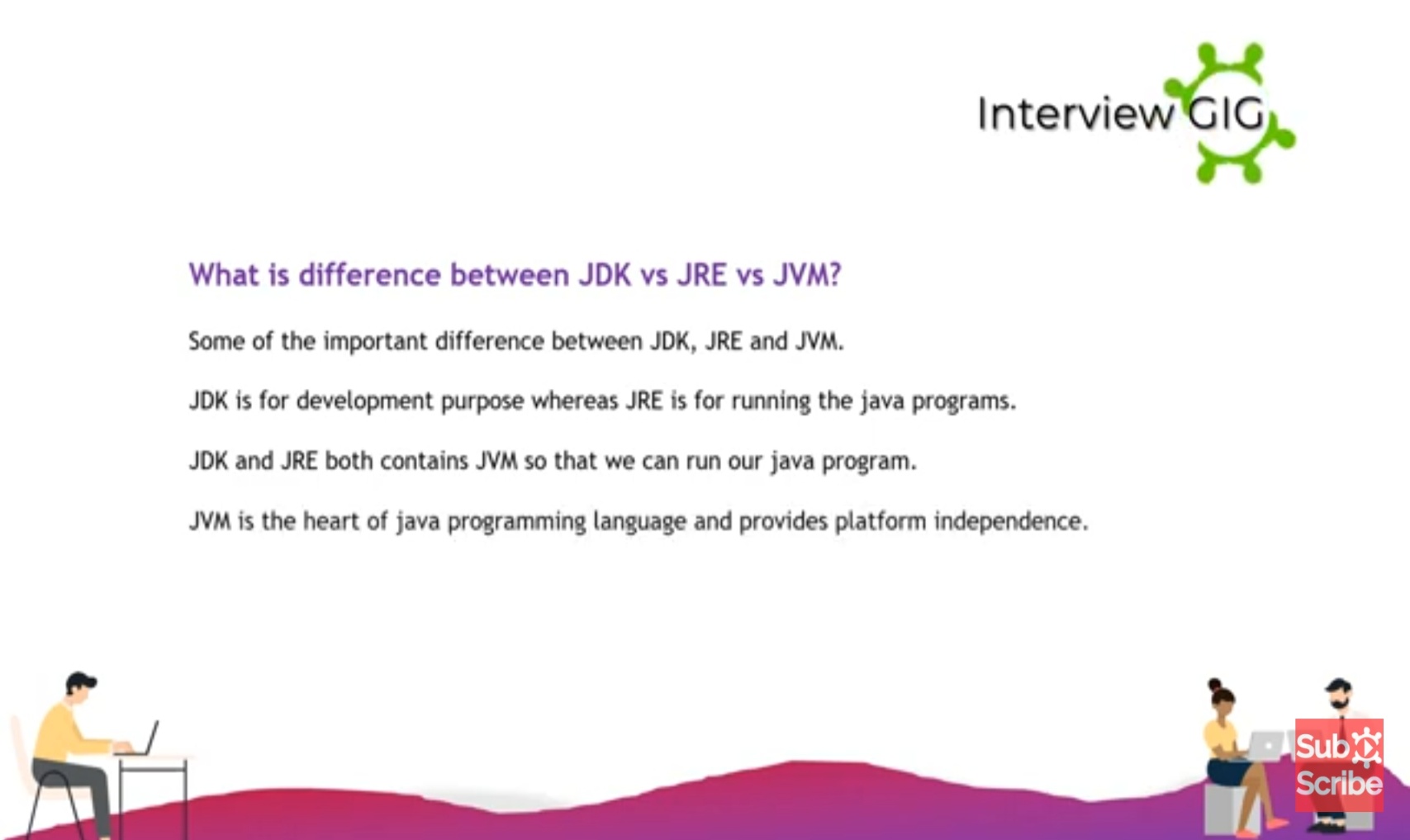
So what pointer do? = they point to the particular memory allocation & if that thing is actually done by jvm internally then why do we need pointers for the same. Thus to avoid direct access to the memory by user because user can perform faulty operation thus pointers is discouraged in java.

((IN SHORT ) JVM is responsible so why do we need pointer.)

#### 53. What are untrusted applets ?

Untrusted applets are those Java applets that cannot access or execute local system files. By default, all downloaded applets are considered as untrusted





**Java 8 Features**

Some of the important Java 8 features are;

1. default and static methods in Interfaces
2. Functional Interfaces and Lambda Expressions
3. Java Stream API for Bulk Data Operations on Collections

**If I download the latest version of Java JDK, do I need to install JRE separately in the system?**

JDK is the Java Development Kit, the base software system that needs to be installed on an operating system if you want to program in Java. The JDK installer comes with JRE as well. If you install JDK, JRE also gets installed. When you run the JDK installer, make sure you go through the entire installation process. This will install JRE as well. JRE is the Java Runtime Environment which lets you execute programs you've created and compiled using the Java Development Kit.

QUES :- TO FIND EVEN OR ODD WITH OUT USING IF ELSE?

**public** **class** Even {

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

// **TODO** Auto-generated method stub

Scanner sc=**new** Scanner(System.***in***);

**int** x= sc.nextInt();

System.***out***.println(x%2==0?"even":"odd");

}

}

QUES :- TO FIND LCM;

int n1 = 72, n2 = 120, gcd = 1;

for(int i = 1; i <= n1 && i <= n2; i++) {

// Checks if i is factor of both integers

if(n1 % i == 0 && n2 % i == 0)

gcd = i;

}

int lcm = (n1 \* n2) / gcd;

System.out.printf("The LCM of %d and %d is %d.", n1, n2, lcm)

OVERRIDDING CONCEPT

**public** **class** overMain {

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

// **TODO** Auto-generated method stub

Employee employee =**new** Marketing();

employee.getSalary();

**Sales sales=(Sales)employee;**

/\*Exception in thread "main" java.lang.ClassCastException: class overloading.Marketing cannot be cast to class overloading.Sales (overloading.Marketing and overloading.Sales are in module ZENSAR of loader 'app')

\*/

sales.getSalary();

}

}

**public** **class** Employee

{

String name="shubhanshu";

**public** **void** getSalary()

{

System.***out***.println("Employee class");

}

}

**public** **class** Marketing **extends** Employee

{

**public** **void** getSalary()

{

name="hello";

System.***out***.println("marketing class");

System.***out***.println(name);

}

}

**public** **class** Sales **extends** Employee

{

**public** **void** getSalary()

{

System.***out***.println("sales class");

}

}

Do class is immutable?

Ans :- class can be both mutable & immutable

**Immutable class** means that once an object is created, we cannot change its content. In Java, all the [wrapper classes](https://www.geeksforgeeks.org/wrapper-classes-java/) (like Integer, Boolean, Byte, Short) and String class is immutable. We can create our own immutable class as well. 

Following are the requirements: 

* The class must be declared as final (So that child classes can’t be created)
* Data members in the class must be declared as private (So that direct access is not allowed)
* Data members in the class must be declared as final (So that we can’t change the value of it after object creation)
* A parametrized constructor should initialize all the fields performing a deep copy (So that data members can’t be modified with object reference)
* Deep Copy of objects should be performed in the getter methods (To return a copy rather than returning the actual object reference)
* No setters (To not have the option to change the value of the instance variable)

### What are Mutable Objects

The mutable objects are objects whose value can be changed after initialization. We can change the object's values, such as field and states, after the object is created. For example, **[Java.util.Date](https://www.javatpoint.com/java-util-date),**[**StringBuilder**](https://www.javatpoint.com/StringBuilder-class)**, [StringBuffer](https://www.javatpoint.com/StringBuffer-class)**, etc.

### What are Immutable Objects

The immutable objects are objects whose value can not be changed after initialization. We can not change anything once the object is created. For example, **primitive objects** such as [int](https://www.javatpoint.com/int-keyword-in-java), [long](https://www.javatpoint.com/long-keyword-in-java), [float](https://www.javatpoint.com/float-keyword-in-java), [double](https://www.javatpoint.com/double-keyword-in-java), **all**[**legacy classes**](https://www.javatpoint.com/legacy-class-in-java)**,**[**Wrapper class**](https://www.javatpoint.com/wrapper-class-in-java)**,**[**String class**](https://www.javatpoint.com/methods-of-string-class), etc.

**Index in Sql?**

**AnS :-** Indexes are **special lookup tables** that the database search engine can use to speed up data retrieval. Simply put, an index is a pointer to data in a table. An index in a database is very similar to an index in the back of a book.

#### What happens when an applet is loaded ?

First of all, an instance of the applet’s controlling class is created. Then, the applet initializes itself and finally, it starts running.

#### What are the restrictions imposed on Java applets ?

Mostly due to security reasons, the following restrictions are imposed on Java applets:

* An applet cannot load libraries or define native methods.
* An applet cannot ordinarily read or write files on the execution host.
* An applet cannot read certain system properties.
* An applet cannot make network connections except to the host that it came from.
* An applet cannot start any program on the host that’s executing it.

#### What is the applet security manager, and what does it provide

The applet security manager is a mechanism to impose restrictions on Java applets. A browser may only have one security manager. The security manager is established at startup, and it cannot thereafter be replaced, overloaded, overridden, or extended.

#### What is the purpose Class.forName method ?

This method is used to method is used to load the driver that will establish a connection to the database.

What is RMI ?

Text, letter

Description automatically generated

What is stub & skeletion in RMI ?

Text, letter

Description automatically generated

Text

Description automatically generated

Text, letter

Description automatically generated

#### What is the basic principle of RMI architecture ?

The RMI architecture is based on a very important principle which states that the definition of the behavior and the implementation of that behavior, are separate concepts. RMI allows the code that defines the behavior and the code that implements the behavior to remain separate and to run on separate JVMs.

#### What are the layers of RMI Architecture ?

The RMI architecture consists of the following layers:

* Stub and Skeleton layer
* Remote Reference Layer
* Transport layer: This layer is responsible for connecting the two JVM participating in the service.

#### Explain Marshalling and demarshalling.

When an application wants to pass its memory objects across a network to another host or persist it to storage, the in-memory representation must be converted to a suitable format. This process is called marshalling and the revert operation is called demarshalling.

#### What is the role of the java.rmi.Naming Class ?

The java.rmi.Naming class provides methods for storing and obtaining references to remote objects in the remote object registry. Each method of the Naming class takes as one of its arguments a name that is a String in URL format.

#### 75. What is the advantage of PreparedStatement over Statement ?

PreparedStatements are precompiled and thus, [their performance is much better](http://examples.javacodegeeks.com/core-java/sql/batch-statement-execution-example/). Also, PreparedStatement objects can be reused with different input values to their queries.

#### 76. What is the use of CallableStatement ?

Name the method, which is used to prepare a CallableStatement. A [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html" \t "_blank) is used to execute stored procedures. Stored procedures are stored and offered by a database. Stored procedures may take input values from the user and may return a result. The usage of stored procedures is highly encouraged, because it offers security and modularity.The method that prepares a [CallableStatement](http://docs.oracle.com/javase/7/docs/api/java/sql/CallableStatement.html" \t "_blank) is the following:

CallableStament.prepareCall();

#### What is difference between fail-fast and fail-safe ?

All the collection classes in java.util package are fail-fast, while the collection classes in java.util.concurrent are fail-safe. Fail-fast iterators throw a [ConcurrentModificationException](http://examples.javacodegeeks.com/java-basics/exceptions/java-util-concurrentmodificationexception-how-to-handle-concurrent-modification-exception/" \t "_blank), while fail-safe iterator never throws such an exception.

#### What does System.gc() and Runtime.gc() methods do ?

These methods can be used as a hint to the JVM, in order to start a garbage collection. However, this it is up to the Java Virtual Machine (JVM) to start the garbage collection immediately or later in time.

#### If an object reference is set to null, will the Garbage Collector immediately free the memory held by that object ?

No, the object will be available for garbage collection in the next cycle of the garbage collector.

#### What is a JSP Page ?

A Java Server Page (JSP) is a text document that contains two types of text:

static data and JSP elements.

Static data can be expressed in any text-based format, such as HTML or XML. JSP is a technology that mixes static content with dynamically generated content

**Can an enum be extended?**

**No**, we **cannot extend** **an enum** in Java. Java enums can extend**java.lang.Enum** class **implicitly**, so enum types cannot extend another class.

Ques :- what is connection pooling ?

Ans = if we required to communicate wit data base multiples times then it is not recommended to create separate. connection object every time because creating and destroying connection object creates performance problem.

To overcome this problem, we should go for connection pool.

Connection pool is pool of already created connection object which are ready to use.

If we want to communicate with database, then we request connection pool to provide connection. Once we got the connection, by using that we can communicate with database. After completing our work, we can return connection to pool instead of destroying.

Advantage = we can reuse same connection object multiple times, so performance is improved .

Data source is responsible to create connection pool

Spring boot dependency

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

Or

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-JDBC</artifactId>

</dependency>

# Caching in Hibernate

Hibernate caching improves the performance of the application by pooling the object in the cache. It is useful when we have to fetch the same data multiple times.

There are mainly two types of caching:

* First Level Cache, and
* Second Level Cache

#### First Level Cache

Session object holds the first level cache data. It is enabled by default. The first level cache data will not be available to entire application. An application can use many session object.

#### Second Level Cache

SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.

Different vendors have provided the implementation of Second Level Cache.

1. EH Cache
2. OS Cache
3. Swarm Cache
4. JBoss Cache

**if we have abstract class then why do we need interface?**

For abstract class only partial implementation is allowed while for interface we have requirement specification but don’t know anything about implementation.

also interface force programmer to provide implementation for each & every method otherwise we have to declare that class as abstract.

**what is stored procedure?**

=>stored procedures are stored & offered by database we use callable statement to call stored procedure.

by database point of view: -

stored procedure contains query, which is written by us, then we call that stored procedure to execute that query. stored procedure may or may not always return result.

**why do we need functional interface?**

=>to implement **functional programming.**

The functional interface has been introduced in Java 8 to support the lambda expression. On the other hand, it can be said lambda expression is the instance of a functional interface.

**“**[**Functional Programming in Java: Lambdas**](https://www.cognizantsoftvision.com/blog/functional-programming-within-java-part-1/)**,”  lambadas expressions can be used to provide an implementation for interfaces that have only one abstract method and this was the reason that the functional interface concept was introduced in Java.**

A functional interface is an interface that contains only one abstract method. They can have only one functionality to exhibit. From Java 8 onwards, lambda expressions can be used to represent the instance of a functional interface. A functional interface can have any number of default method

Java has forever remained an Object-Oriented Programming language. By object-oriented programming language, we can declare that everything present in the Java programming language rotates throughout the Objects, except for some of the primitive data types and primitive methods for integrity and simplicity.

Functions in the Java programming language are part of a class, and if someone wants to use them, they have to use the class or object of the class to call any function.

**Why doesn’t java support multiple inheritance?**

Because of ambiguity problem java doesn’t support multiple inheritance

//first parent class

class ParentA

{

//same signature method in both class named as walk()

void walk()

{

System.out.println("Parent A is walking");

}

}

//second parent class

class ParentB

{

void walk()

{

System.out.println("Parent B is walking");

}

}

//multiple inheritance in child class results in error

class child extends ParentA, ParentB

{

public static void main(String[] args)

{

child object = new child();

object.walk();

}

}

Java

Copy

Output :

Compile time Error

Java provide support for multiple inheritance through interface.

//interface 1

interface ParentA

{

//interfaces method are only declared not defined

public void walk();

}

//interface 2

interface ParentB

{

//any number of methods can be declared in the interface

public void walk();

public void run();

}

//multiple inheritance achieved

class child implements ParentA,ParentB

{

//overidden methods

public void walk()

{

System.out.println("ParentA is walking ");

}

public void run()

{

System.out.println("ParentB is running ");

}

public static void main (String args[])

{

child object = new child();

object.walk();

object.run();

}

}

Java

Copy

output:

ParentA is walking

ParentB is running

Java

Copy

hence, in the above code we have achieved Multiple inheritance in java with the help of interfaces where we have declared our methods in the interfaces and overridden them in the child class. (override means when a method that has already been declared in parent class is being defined again in the child class having same name, number and type of parameters, and return type as the method that it overrides) so, we have re-implemented the walk() and run() method of parent class in the child class to use them. When the object of child class will call method they will be called without getting ambiguity and will print the output.

***Conclusion :-***Java does not support "multiple inheritance" (a class can only inherit from one parent class). However, it can be achieved with help of interfaces, because the class can implement multiple interfaces.

## What is String Pool in Java?

String Pool in Java is a special storage space in [*Java*](https://www.scaler.com/topics/java/) Heap memory where string literals are stored. It is also known by the names - **String Constant Pool** or **String Intern Pool**. Whenever a string literal is created, the JVM first checks the String Constant Pool before creating a new String object corresponding to it.

By default, the String pool is empty. A pool of strings decreases the number of String objects created in the JVM, thereby reducing memory load and improving performance.

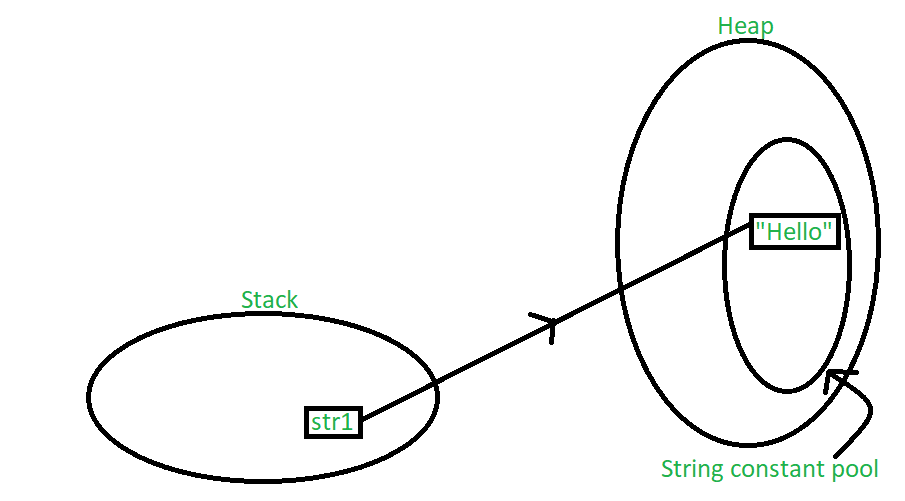
Let us first discuss the memory allocation methods used in Java - **Stack Memory Allocation** and **Heap Memory Allocation**.

* In stack memory, only the primitive data types like- int, char, byte, short, boolean, long, float and double are stored.
* Whereas, in the heap memory, non-primitive data types like strings are stored. A reference to this location is held by the stack memory.

A string constant pool is a separate place in the heap memory where the values of all the strings which are defined in the program are stored. When we declare a stringith string literal then, an object(variable) of type String is created in the stack, while an instance with the value of the string is created in String Constant Pool inthe heap. On standard assignment of a value to a string variable, the variable is allocated stack, while the value is stored in the heap in the string constant pool. For example, let’s assign some value to a string str1. In java, a string is defined and the value is assigned as:

String str1 = "Hello";

The following illustration explains the memory allocation for the above declaration:

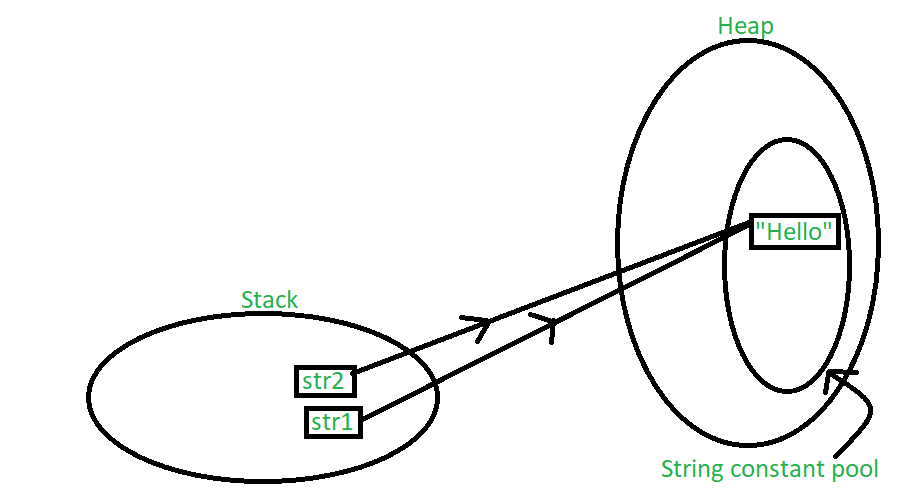
[](https://media.geeksforgeeks.org/wp-content/uploads/20200601211147/string_pool_11.png)

In the above scenario, a string object is created in the stack, and the value “Hello” is created and stored in the heap. Since we have normally assigned the value, it is stored in the constant pool area of the heap. A pointer points towards the value stored in the heap from the object in the stack. Now, let’s take the same example with multiple string variables having the same value as follows:

String str1 = "Hello";

String str2 = "Hello";

The following illustration explains the memory allocation for the above declaration:

[](https://media.geeksforgeeks.org/wp-content/uploads/20200601211203/string_pool_2.png)

In this case, both the string objects get created in the stack, but another instance of the value “Hello” is not created in the heap. Instead, the previous instance of “Hello” is re-used. Whenever a string literal is created, the JVM first checks the String Constant Pool before creating a new String object corresponding to it.if it is present the previous instance is reused. The string constant pool is a small cache that resides within the heap. Java stores all the values inside the string constant pool on direct allocation. This way, if a similar value needs to be accessed again, a new string object created in the stack can reference it directly with the help of a pointer. In other words, the string constant pool exists mainly to reduce memory usage and improve the re-use of existing instances in memory. When a string object is assigned a different value, the new value will be registered in the string constant pool as a separate instance. Lets understand this with the following example:

String str1 = "Hello";

String str2 = "Hello";

String str3 = "Class";

The following illustration explains the memory allocation for the above declaration:

[Diagram

Description automatically generated](https://media.geeksforgeeks.org/wp-content/uploads/20200602014728/string_pool_3.png)

One way to skip this memory allocation is to use the [new keyword](https://www.geeksforgeeks.org/new-operator-java/) while creating a new string object. The ‘new’ keyword forces a new instance to always be created regardless of whether the same value was used previously or not. Using ‘new’ forces the instance to be created in the heap outside the string constant pool which is clear, since caching and re-using of instances isn’t allowed here. Let’s understand this with an example:

String str1 = new String("John");

String str2 = new String("Doe");

The following illustration explains the memory allocation for the above declaration:

[Diagram

Description automatically generated](https://media.geeksforgeeks.org/wp-content/uploads/20200602104736/output_string_4.png)

# How is creation of a String with new() different from a literal?

|  |  |
| --- | --- |
| **String creation using new()** | **String creation using String literal** |
| If we create a String using new(), then a new object is created in the heap memory even if that value is already present in the heap memory. | If we create a String using String literal and its value already exists in the string pool, then that String variable also points to that same value in the String pool without the creation of a new String with that value. |
| It takes more time for the execution and thus has lower performance than using String literal. | It takes less time for the execution and thus has better performance than using new(). |
| Example:  String n1= new String(“Java”);  String n2= new String(“Java”);  String n3= new String(“Create”); | Example:  String s1=”Java”;  String s2=”Java”;  String s3=”Create”; |

# Lifecycle and States of a Thread in Java

A [thread](https://www.geeksforgeeks.org/multithreading-in-java/) in Java at any point of time exists in any one of the following states. A thread lies only in one of the shown states at any instant:

1. New
2. Runnable
3. Blocked
4. Waiting
5. Timed Waiting
6. Terminated

#### Life Cycle of a thread

1. **New Thread:** When a new thread is created, it is in the new state. The thread has not yet started to run when the thread is in this state. When a thread lies in the new state, its code is yet to be run and hasn’t started to execute.
2. **Runnable State:** A thread that is ready to run is moved to a runnable state. In this state, a thread might actually be running or it might be ready to run at any instant of time. It is the responsibility of the thread scheduler to give the thread, time to run.   
   A multi-threaded program allocates a fixed amount of time to each individual thread. Each and every thread runs for a short while and then pauses and relinquishes the CPU to another thread so that other threads can get a chance to run. When this happens, all such threads that are ready to run, waiting for the CPU and the currently running thread lie in a runnable state.
3. **Blocked/Waiting state:** When a thread is temporarily inactive, then it’s in one of the following states:
   * Blocked
   * Waiting
4. **Timed Waiting:** A thread lies in a timed waiting state when it calls a method with a time-out parameter. A thread lies in this state until the timeout is completed or until a notification is received. For example, when a thread calls sleep or a conditional wait, it is moved to a timed waiting state.
5. **Terminated State:** A thread terminates because of either of the following reasons:
   * Because it exits normally. This happens when the code of the thread has been entirely executed by the program.
   * Because there occurred some unusual erroneous event, like segmentation fault or an unhandled exception.

|  |
| --- |
| Java program to demonstrate thread states  **class** thread **implements** Runnable {  **public** **void** run()      {          // moving thread2 to timed waiting state  **try** {              Thread.sleep(1500);          }  **catch** (InterruptedException e) {              e.printStackTrace();          }            System.out.println(              "State of thread1 while it called join() method on thread2 -"              + Test.thread1.getState());  **try** {              Thread.sleep(200);          }  **catch** (InterruptedException e) {              e.printStackTrace();          }      }  }    **public** **class** Test **implements** Runnable {  **public** **static** Thread thread1;  **public** **static** Test obj;    **public** **static** **void** main(String[] args)      {          obj = **new** Test();          thread1 = **new** Thread(obj);            // thread1 created and is currently in the NEW          // state.          System.out.println(              "State of thread1 after creating it - "              + thread1.getState());          thread1.start();            // thread1 moved to Runnable state          System.out.println(              "State of thread1 after calling .start() method on it - "              + thread1.getState());      }    **public** **void** run()      {          thread myThread = **new** thread();          Thread thread2 = **new** Thread(myThread);            // thread1 created and is currently in the NEW          // state.          System.out.println(              "State of thread2 after creating it - "              + thread2.getState());          thread2.start();            // thread2 moved to Runnable state          System.out.println(              "State of thread2 after calling .start() method on it - "              + thread2.getState());            // moving thread1 to timed waiting state  **try** {              // moving thread1 to timed waiting state              Thread.sleep(200);          }  **catch** (InterruptedException e) {              e.printStackTrace();          }          System.out.println(              "State of thread2 after calling .sleep() method on it - "              + thread2.getState());    **try** {              // waiting for thread2 to die              thread2.join();          }  **catch** (InterruptedException e) {              e.printStackTrace();          }          System.out.println(              "State of thread2 when it has finished it's execution - "              + thread2.getState());      }  } |

**Output**

State of thread1 after creating it - NEW

State of thread1 after calling .start() method on it - RUNNABLE

State of thread2 after creating it - NEW

State of thread2 after calling .start() method on it - RUNNABLE

State of thread2 after calling .sleep() method on it - TIMED\_WAITING

State of thread1 while it called join() method on thread2 -WAITING

State of thread2 when it has finished it's execution - TERMINATED

**Multiple inheritance ininterface in java 8**

In **Java 8** you cannot implement multiple interfaces that having same signature, without explicitly[**overriding the methods**](https://javainterviewpoint.com/can-we-override-static-methods-in-java/) in the child class.

interface Car

{

public default void drive()

{

System.out.println("Car is driving");

}

}

interface Jeep

{

public default void drive()

{

System.out.println("Jeep is driving");

}

}

public class Vehicle implements Car,Jeep

{

@Override

public void drive()

{

Car.**super**.drive(); // or Jeep.sper.drive();

System.out.println("Vehicle is driving");

}

public static void main(String args[])

{

Vehicle v = new Vehicle();

v.drive();

}

}

**Output :**

Car is driving  
***Vehicle is driving***

In **Java 8** Interface can also have method definition using Default Methods, then obviously it should also result in ambiguity isn’t it ? **Yes**, but**Java 8** can handle this type of compatible issues. Lets look into the below example.

package com.javainterviewpoint;

interface Car

{

public default void drive()

{

System.out.println("Car is driving");

}

}

interface Jeep

{

public default void drive()

{

System.out.println("Jeep is driving");

}

}

public class Vehicle implements Car,Jeep

{

public static void main(String args[])

{

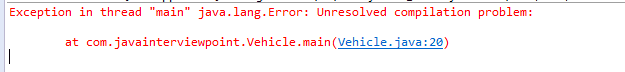
Vehicle v = new Vehicle();

v.drive();

}

}

when we try to execute above class, we will be getting the ***“Unresolved compilation problem”***

**[](https://javainterviewpoint.com/wp-content/uploads/2016/10/Multiple-Inheritance-in-Java-8.png)**

### ****Explicit Overriding :****