Class: A class can be understood as a template or a blueprint, which contains some values, known as member data or member, and some set of rules, known as behaviors or functions.

Interface: An interface refers to a special type of class, which contains methods, but not their definition. Only the declaration of methods is allowed inside an interface.

Object: An instance of a particular class or an entity that has state and Behavior is known as an object e.g., chair, bike,marker, pen, table, car, etc.

Abstract class: An abstract class is a special class containing abstract methods. The significance of abstract class is that the abstract methods inside it is not implemented and only declared. when a subclass inherits the abstract class and needs to use its abstract methods, they need to define and implement them.

Method

A method is a block of code which only runs when it is called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.

API (Application programming interface)

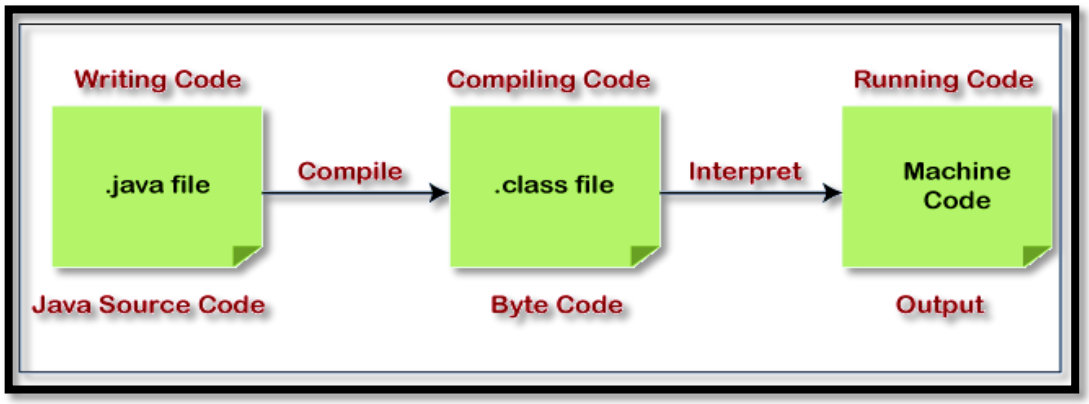
An API is the area of JDK. API includes classes, interfaces, packages and also their methods, fields, and constructors. All these built-in classes give benefits to the programmer.

REST API

It stands for Representational State Transfer API which is an architectural style that defines a set of rules to be used for creating web services. REST API is a way of accessing web services in a simple and flexible way without having any processing.

Thread

A thread is a lightweight process that helps program to perform multiple tasks concurrently. It shares the same memory space and resources but execute independently, and helps to achieve parallelism and responsiveness.



Framework

Frameworks are sets of classes and interfaces that provide a readymade architecture.

Collection Framework

The Collection Framework in Java is a set of classes and interfaces that provide a unified architecture for manipulating and storing groups of objects.

This

Keyword is a reference variable that refers to the current invoking object of a method or a constructor.

Super

We can use super keyword to access the data member or field of immediate parent class.

Applet

An applet is a java program that runs within the web browser. Applets use a graphical user interface and may have text, images, buttons, scrollbars, and sound.

Instance

An object of a particular class. An instance of a class is created using the new operator followed by the class name.

Java 8 Features

- Stream API
- Lambda Expressions
- Functional Interfaces
- Default Methods
- Date/Time API

Stream API

The Stream API processes collections of objects in Java. A stream is a sequence of objects supporting various methods that can be pipelined to produce the desired result.

There are two types of Operations in Streams:

- **Intermediate Operations**: Intermediate Operations are the types of operations in which multiple methods are chained in a row. Like → map(), filter(), sorted()
- **Terminate Operations**: Terminal Operations are the type of Operations that return the result. These Operations are not processed further just return a final result value. Like → collect(), forEach(), reduce()

Lambda expressions

Lambda Expressions are the short block of code that accepts input as parameters and returns a resultant value. it helps to create small, inline function implementations.

Functional Interfaces

Interfaces with a single abstract method, such as Runnable, Callable, and Comparator. Annotated with @FunctionalInterface is called Function interfaces.

Default Methods

Allow adding new methods to interfaces without breaking existing implementations. Defined with the default keyword.

Date/Time API

This provide flexible date/time library. Classes like LocalDate, LocalTime, LocalDateTime, ZonedDateTime, and Duration replace java.util.Date and java.util.Calendar.

Collectors

Part of the Stream API used to accumulate elements into collections, summaries, or other results. Methods like toList(), toSet(), toMap(), joining(), and groupingBy().

Java Enums

Java Enum is a data type which contains fixed set of constants.

Enterprise javabeans (EJB)

It is a serverside specification that handles business logic such as persistence, transactional integrity, and security in a standardized way.

Classloader

It is a part of the JVM that dynamically loads Java classes into the Java Virtual Machine

Generics

Java generics is a set of similar types of methods or related methods. Generics are mostly used by classes like hashset or hashmap.

Checked exception

An exception that is caught at the compilation time. For example, the java.io.ioexception is a checked exception.

New

The new keyword is used to allocate memory at runtime.

Constructor

A constructor in java is a special method that is used to initialize objects. The constructor is called when an object of a class is created.

Types of java constructors

1) Default constructor (no-arg constructor)
2) Parameterized constructor

Destructor

Destructors free up the resources and memory occupied by an object. Destructors are automatically called when an object is being destroyed.

Constant

A final variable in java, which means that the reference of it cannot be changed once

Continue

It is used to end the current loop process in a loop and continues to the next process.

Recursion

Recursion is a process in which a method calls itself continuously. That method called as recursive method. And its process called as recursion.

DOM

Document object model, defined by the w3c, that allows applications to dynamically access and update the content, structure, and style of documents.

Java garbage collection

It is the process by which java programs perform automatic memory management. It finds the unused objects (that are no longer used by the program) and delete or remove them to free up the memory.

Exception

An exception is an event that disrupts the normal flow of the program.

Extends

A keyword used to define the inheritance of classes or interfaces

Final

Final is a java keyword. When you define an entity once with 'final' and you cannot change it or derive from it later.

Finally

Finally defines a block of code we use along with the try keyword. It defines code that's always run after the try and any catch block, before the method is completed. It executed even java exception or runtime error occurred.

Global variable

A variable that is visible to all methods in the class.

Local variable

A variable defined in the method body, visible only inside it.

Hash code

A value used to provide an efficient way to map object and its location, returned by a hash function

Hash function

A hash function converts strings of different length into fixed length strings known as hash values or digests.

Hexadecimal

A number represented by the base of 16

Identifier

A name of a class, variable, method or interface defined in the code by the software developer

Immutable object

An object whose state or value is not changeable after creation.

Implements

A java keyword used to indicate which interfaces are implemented by the current class

Inheritance

It is a mechanism in java by which one class is allowed to inherit the features of another class.

Iteration

A single execution of a loop.

Jar stands for java archive. It's a file format and is used for aggregating many files into one.

War file

The war file contains the web application that can be deployed on any servlet /JSP container. The .war file contains JSP, html, javascript and other files necessary for the development of web applications.

Synchronized

The process of allowing only a single thread to access the shared data or resource at a particular point of time.

Wrapper

It is a class which contains the primitive data types.

Servlets

A servlet is an extension to a server that enhances the server's functionality.

Java core

Provides the main features of java, also named java standard edition

JAVA SE

SE stands for Standard Edition. It helps you to develop and deploy Java applications on desktops and servers. Java offers the rich user interface, performance, versatility, portability, and security that today's applications require.

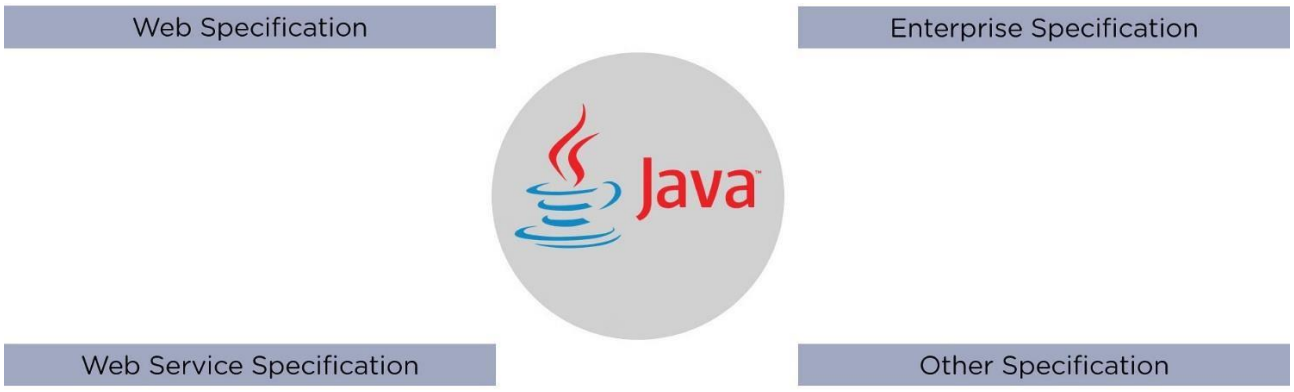
- It has has features like class libraries and deployment environments.
- We use Java SE for desktop and mobile application development.

JAVA EE

Earlier it is known as J2EE and currently known as Jakarta EE. It provides a platform for developers with enterprise features such as distributed computing and web services.

- It focuses on the highend and corporate type of Java Applications.
- It structured application with clients, business, and enterprise layers.
- We use mainly Java EE for web development.

Java EE Specification



JDK

JDK is a set of tools for developing Java applications. It is a development environment for building applications, applets, and components using the Java programming language. It provides the environment and core libraries used to write java programs.

Jvm

Java virtual machine, the abstract machine where the compiled java byte code is executed.

Jre

Jre (java™ runtime environment) is part of a jdk. Jre is a set of components to create and run a java application.

Module

A group of program components; in java, the term that's used for it is package

Nested class

A class, which is implemented inside the body of the other class

Null

A type indicating that object reference variable has no reference to any object existing in memory.

Package

A group of classes and interfaces.

Library a java library is just a collection of classes that have been written by somebody else already.

Primitive type

Boolean, byte, char, double, float, int, long or short.

Serialization

The process of encoding and decoding the objects to the stream of bytes.

JDBC

JDBC stands for java database connectivity is a java API to connect and execute the query with the database.

Return

A java keyword used to finish the execution of a method.

RPC

A RPC (remote procedure call) is an inter-process communication technique that is used for client-server-based applications.

Apache poi

Apache poi is an API provided by apache foundation which is collection of different java libraries. These library gives the facility to read, write and manipulate different Microsoft files such as excel sheet, power point, and word files.

Primitive datatype size

Byte 8 bits (1 byte) || Minimum Value 128 || Maximum Value 127

Short 16 bits (2 bytes) || Minimum Value 32,768 || Maximum Value 32,767

Int 32 bits (4 bytes) || Minimum Value 2,147,483,648 || Maximum Value 2,147,483,647

Long 64 bits (8 bytes) || Minimum Value 9,223,372,036,854,775,808 || Maximum Value 9,223,372,036,854,775,807

Float 32 bits (4 bytes) || Sufficient for storing 6 to 7 decimal digits

Double 64 bits (8 bytes) || Sufficient for storing 15 decimal digits

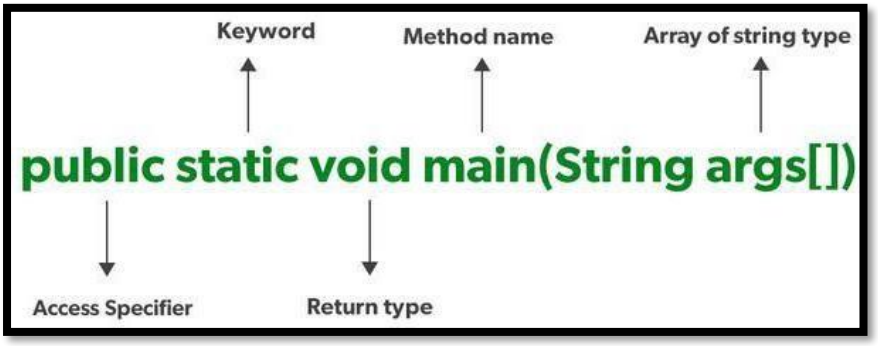
Char 16 bits (2 bytes)

Boolean not explicitly defined, but typically implemented as 8 bits (1 byte)

String pool

The string pool in Java is a special memory area where literal strings (strings which are enclosed between double quote ("")) are stored to optimize memory usage. When a string literal is encountered in code, Java checks if it already exists in the pool. If it does, the existing instance is reused; otherwise, a new instance is created and added to the pool. This helps reduce memory consumption by avoiding duplicate string objects.

Explain public static void main(String args[]) in Java.



What is a singleton class?

A class that restricts the instantiation to one object.

What is a nested class?

A class defined within another class.

What is an inner class?

A non-static nested class.

What is an anonymous inner class?

An inner class without a name, typically used for event handling.

What is a local inner class?

An inner class defined within a method.

What is a static nested class?

A static class defined within another class, can be instantiated without an instance of the outer class.

What is the difference between == and .equals() method in Java? When would you use one over the other?

Answer "==" compares the memory location of two objects, while "equals" compares the contents of two objects.

What is the purpose of the finalize() method in Java? Is it recommended to use it?

Answer it perform cleanup operations before an object is garbage collected.

How does exception handling work in Java? Explain the difference between checked and unchecked exceptions.

Answer Exception handling in Java allows developers to handle errors, exceptions, and abnormal conditions that may occur during the execution of a program. Java provide below mechanisms for exception handling.

1) Trycatchfinally Blocks

The try block is used to enclose the code that might throw an exception.

The catch block is used to handle specific exceptions that occur within the try block. Multiple catch blocks can be used to handle different types of exceptions.

The finally block is used to execute code that needs to be run regardless of whether an exception occurs or not. It's typically used for releasing resources like file handles or database connections.

2) Throw Statement

The throw statement is used to explicitly throw an exception from within a method.

3) Throws Clause

The throws clause is used in method declarations to indicate that the method might throw certain types of exceptions. It informs the caller of the method about the exceptions that need to be handled.

4) Custom Exception Classes

Developers can create their own exception classes by extending the Exception class or its subclasses.

Checked exceptions

- Checked exceptions happen at compile time when the source code is transformed into an executable code.
- The checked exception is checked by the compiler.
- Checked exceptions can be created manually.
- This exception is counted as a subclass of the class.
- Java Virtual Machine requires the exception to be caught or handled.

Unchecked exceptions

- Unchecked exceptions happen at runtime when the executable program starts running.
- These types of exceptions are not checked by the compiler.

- They can also be created manually.
- This exception happens in runtime, and hence it is not included in the exception class.
- Java Virtual Machine does not need the exception to be caught or handled.

Describe the concept of polymorphism in Java with an example.

Answer Polymorphism in Java is a concept by which we can perform a single action in different ways.

We can achieve polymorphism in Java using the following ways

1) Method Overriding Method overriding occurs when a subclass provides a specific implementation of a method that is already defined in its superclass

2) Method Overloading Method overloading allows multiple methods with the same name in a same class, but with different parameter lists.

Polymorphism in Java can be classified into two types, example

1) Static/compiletime Polymorphism It could be achieved through Overloading

2) Dynamic/Runtime Polymorphism It could be achieved through Overriding

What is the difference between hashmap and hashtable in Java? When would you use one over the other?

Answer hashmap and hashtable store key and value pairs in a hash table. Hashmap is nonsynchronized. It is not threadsafe and can't be shared between many threads without proper synchronization code whereas Hashtable is synchronized. It is threadsafe and can be shared with many threads.

Hashmap allows one null key and multiple null values whereas Hashtable doesn't allow any null key or value. Hashmap is generally preferred over hashtable if thread synchronization is not needed.

Explain the concept of multithreading in Java. How would you create and manage threads in Java?

Answer Multithreading in Java refers to the ability of a program to execute multiple threads concurrently. Multithreading enables programs to perform multiple tasks at the same time, it improves performance and responsiveness.

Multithreading can be achieved by either extending the Thread class or implementing the Runnable interface.

Thread States

Threads in Java can be in different states, including

New When a thread is created but not yet started.

Runnable When a thread is ready to run and waiting for its turn to be picked for execution by the thread scheduler.

Blocked When a thread is waiting for a monitor lock to enter a synchronized block or method.

Waiting When a thread is waiting indefinitely for another thread to perform a particular action.

Terminated When a thread has completed execution or has been terminated.

Thread Pooling

Thread pooling is a technique where a group of threads are created and reused to execute tasks rather than creating a new thread for each task.

Pros and cons

Advantages in terms of performance, responsiveness, and resource utilization. However, it also introduces complexity and potential issues related to synchronization and concurrency.

What is the difference between abstract class and interface in Java? When would you use one over the other?

Answer Abstract class and interface both are used to achieve abstraction.

Abstract class

- Abstract class can have abstract and nonabstract methods.
- Abstract class doesn't support multiple inheritance.
- Abstract class can have final, nonfinal, static and nonstatic variables.
- An abstract class can extend another Java class and implement multiple Java interfaces.

Interface

- Interface supports multiple inheritance.
- Interface has only static and final variables.
- An interface can extend another Java interface only.

You have to use an abstract class instead of an interface when you need more control over access modifiers, or are designing an API with evolving functionality.

How does Java handle memory management and garbage collection?

Answer objects are dynamically allocated memory from the heap using the new keyword. When the heap becomes full, garbage is collected. It makes the spaces for new objects.

List of garbage collector in java

- Serial Garbage Collector (serialgc)
- Parallel Garbage Collector (parallelgc)
- Concurrent marksweep (CMS) Garbage Collector
- G1 Garbage Collector
- Z Garbage Collector (ZGC)

Explain the difference between static and final keywords in Java. How is final used in variables, methods, and classes?

Answer

Static

It belongs to the class, shared by all objects. Makes a single copy of the variable in memory.

Final

It makes a variable constant; its value cannot be changed after initialization. The final keyword is a nonaccess modifier used for classes, attributes and methods, which makes them non-changeable (impossible to inherit or override)

Describe the principle of encapsulation in Java. Why is it important?

Answer Encapsulation refers to the bundling of fields and methods inside a single class. It prevents outer classes from accessing and changing fields and methods of a class. This also helps to achieve data hiding.

Encapsulation is important because it provides a powerful way to store, hide, and manipulate data while giving you increased control over it.

How does Java support concurrent programming? Discuss some classes/interfaces related to concurrent programming.

Answer Concurrent programming involves designing and implementing programs that can execute multiple tasks concurrently, allowing different parts of the program to run independently and simultaneously.

Java support concurrent programming through its builtin features, libraries, and language constructs. It allows threads and Synchronization mechanism

Java provides several classes and interfaces in the java.util.concurrent package to support concurrent programming.

Some interfaces like Executor, executor-service, Callable and Future, completion-service and class like Semaphore, countdown latch.

Explain the difference between arraylist and linkedlist in Java. When would you use one over the other?

Answer array list uses an array, which allows for fast random access but slow insertion and deletion. While LinkedList uses a doubly linked list, which allows for fast insertion and deletion but slow random access.

Usage Scenarios

Use arraylist when

- You need fast random access to elements by index.
- You frequently add or remove elements at the end of the list.
- Memory overhead is a concern.

Use linkedlist when

- You frequently add or remove elements at the beginning or end of the list.
- You need to efficiently insert or remove elements at arbitrary positions in the list.
- You do not require frequent random access to elements by index.

What is the purpose of the transient keyword in Java, and when would you use it?

Answer Transient is a variables modifier used in serialization. At the time of serialization, if we don't want to save value of a particular variable in a file, then we use transient keyword

Describe the Singleton design pattern in Java. How would you implement it, and what are the potential issues with different implementations?

Answer Singleton pattern ensures that only one instance of the class exists in the Java Virtual Machine and provides a global point of access to it

There are two forms of singleton design pattern

- Early Instantiation Creation of instance at load time.
- Lazy Instantiation Creation of instance when required.

Singleton class

A singleton class in Java ensures only one instance of itself exists. This is achieved by making the constructor private, so that no other instances of the class can be created, and providing a static method that returns the single instance of the class.

we can make singleton class with following points.

1) Make a constructor private.

2) Write a static method that has the return type object of this singleton class.

Why we create Singleton Class.

The primary purpose of a Single class is to restrict the limit of the number of object creation to only one.

What is Method hiding?

If you create a similar method with the same return type and same method arguments in child class then it will hide the superclass method, this is known as method hiding.

Java 2D

Java 2D API enables you to easily perform the following tasks: Draw lines, rectangles and any other geometric shape. It comes under JRE. JNDI (Java Naming and Directory Interface) JNDI (Java Naming and Directory Interface) enables Java platform-based applications to access multiple naming and directory services. it is Part of the Java Enterprise application programming interface (API) set, JNDI makes it possible for developers to create portable applications that are enabled for a number of different naming and directory services. It comes under JRE.

JRE (Java Runtime Environment)

The Java Runtime Environment, or JRE, is a software layer that runs on top of a computer's operating system software and provides the class libraries and other resources that a specific Java program needs to run. The JRE is one of three inter-related components for developing and running Java programs. The other two components are JDK, JVM. JRE also known as Java RTE. JRE includes the Java Virtual Machine (JVM), core classes, and supporting files.

The source Java code gets compiled and converted to Java bytecode. If you wish to run this bytecode on any platform, you require JRE. JRE acts as a layer on the top of the operating system.

JRE consists of the following components:

Deployment technologies such as deployment, Java plug-in, and Java Web Start.

User interface toolkits, including Abstract Window Toolkit (AWT), Swing, Java 2D, Accessibility, Image I/O, Print Service, Sound, drag, and drop (DnD) and input methods.

Integration libraries including Interface Definition Language (IDL), Java Database Connectivity (JDBC), Java Naming and Directory Interface (JNDI), Remote Method Invocation (RMI), Remote Method Invocation Over Internet Inter-Orb Protocol (RMI-IIOP) and scripting.

Other base libraries, including international support, input/output (I/O), extension mechanism, Beans, Java Management Extensions (JMX), Java Native Interface (JNI), Math, Networking, Override Mechanism, Security, Serialization and Java for XML Processing (XML JAXP).

Lang and util base libraries, including lang and util, zip, Java Archive (JAR), instrument, reflection, Collections, Concurrency Utilities, management, versioning, Logging, Preferences API, Ref Objects and Regular Expressions.

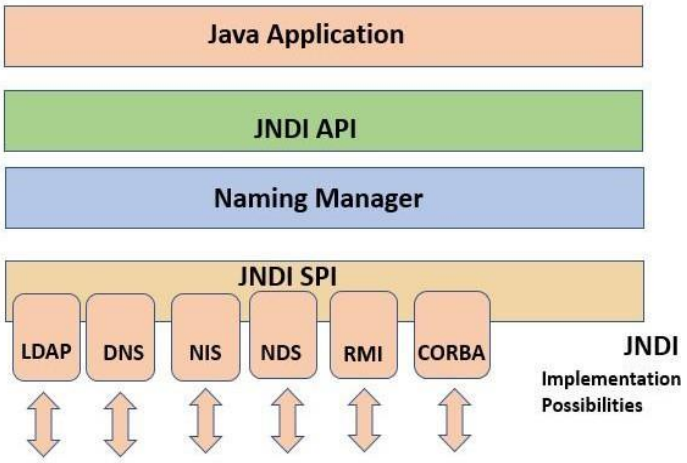
Java Virtual Machine (JVM), which comprise of Server Virtual Machine and Java HotSpot Client.

JDK (Java Development Kit)

JDK is a set of tools for developing Java applications. Developers choose JDKs by Java version and by package or edition—Java Enterprise Edition (Java EE), Java Special Edition (Java SE), or Java Mobile Edition (Java ME). Every JDK always includes a compatible JRE, because running a Java program is part of the process of developing a Java program.

JNDI (Java Naming and Directory Interface)

JNDI (Java Naming and Directory Interface) enables Java platform-based applications to access multiple naming and directory services. it is Part of the Java Enterprise application programming interface (API) set, JNDI makes it possible for developers to create portable applications that are enabled for a number of different naming and directory services. It comes under JRE.



JNDI comes with two things naming and directory service means

Naming service: It a place where we put objects and methods which is going to be used in our distributed application. And provide them a unique name. So that if we want to use that obj. or method so we can call it through their name directly.

Directory service. It a place where we put objects and methods which is going to be used in our distributed application. And provide them a unique name and a attribute. So that if we want to use that obj. or method so we can call it through their name or directory directly.

If we want to use Naming and Directory service in program we can use JNDI API. Which comes with two thing API & SPI.

JMX (Java Management Extensions)

JMX provide a standard way to managing resource, and monitoring applications and networks. It helps to manage application, objects, devices, service oriented networks.

What is the use of "System.out.printf" in java?

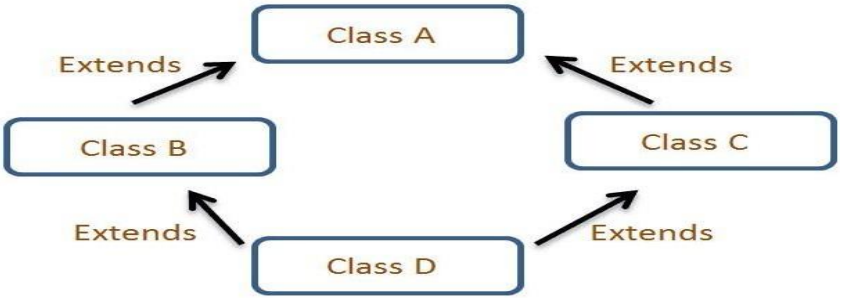
System. out. printf method is used to format the output of numbers and strings.

Example: System.out.printf(Locale.getDefault(),"my name is shubham jadon.",1);

output: my name is shubham jadon.

Why Java doesn't support multiple inheritance?

1) First reason is ambiguity around the Diamond problem, Consider a case where class B extends class A and Class C and both class A and C have the same method display(). Now java compiler cannot decide, which display method it should inherit. To prevent such situation, multiple inheritances is not allowed in java. This is also called the Diamond problem because the structure on this inheritance scenario is similar to 4 edge diamond, see below



2. Second and more convincing reason to me is that multiple inheritances does complicate the design and creates problem during casting, constructor chaining etc. like we call super() from the constructor. if there are two super class then compiler could be confuse which have to choose etc.

A class can implement any number of interfaces but can extend only one class.

The wrapper classes, Byte, Character, Short, Integer, Long, Float and Double are all immutable.

Why interfaces allowed multiple inheritance and does not allowed?

Because interfaces specify only what the class is doing, not how it is doing it.

What is immutable means?

The string is immutable means that we cannot change the object itself, but we can change the reference to the object.

We can make objects of a class immutable with multiple ways: like

- Final keyword.
- Make variable private.

Why does Java not support operator overloading?

Java doesn't supports operator overloading because it's just a choice made by its creators who wanted to keep the language more simple. If you allow programmer to do operator overloading they will come up with multiple meanings for same operator which will make the learning hard and things more confusing and messing.

Why String is final or immutable in Java?

1) The String is immutable in Java because of the security, synchronization and concurrency, caching, and class loading. The reason of making string final is to destroy the immutability and to not allow others to extend it.
2) Immutable string or objects that cannot be modified once it is created. But we can only change the reference to the object.
3) The String objects are cached in the String pool, and it makes the String immutable. The cached String literals are accessed by multiple clients. So, there is always a risk, where action performs by one client affects all other clients. that's why it is immutable. For example, if one client performs an action and changes the string value from Pressure to PRESSURE, all remaining clients will also read that value. For the performance reason, caching of String objects was important, so to remove that risk, we have to make the String Immutable.

What will happen if you put the return statement or System.exit () on the try or catch block? Will finally block execute?

If we use return statement inside try or catch block, finally block will always execute but we use System.exit () inside try or catch block, finally block will not execute.

Note: 1) finally block will always execute even an exception occurred or not in Java.

2) There are few situations where the finally will not be executed like JVM crash, power failure, software crash and etc.

Can you override a private method in Java?

No, we cannot override private or static methods in Java.

Private methods in Java are not visible to any other class which limits their scope to the class in which they are declared.

Can you overload a private method in Java?

Yes, we can overload private methods in Java but, you can access these from the same class due to their scope.

Can you overload static method in Java?

Yes, we can.

Can you override static method in Java?

No, we cannot override static methods because method overriding is based on dynamic binding at runtime and the static methods are bonded using static binding at compile time.

Can we override or overload final method in Java?

No, the Methods that are declared as final cannot be Overridden or hidden but we can override them.

What do the expression 1.0 / 0.0 will return?

The answer to this question is that 1.0 / 0.0 will compile successfully. And it will not throw ArithmeticException. It will just return Double. INFINITY

If a method throws NullPointerException in the superclass, can we override it with a method that throws RuntimeException.

Yes, we can override the method that throws runtimeException. because NullPointerException sub-class of RuntimeException **stringbuffer v/s stringbuilder**

- A string buffer is thread-safe whereas string builder is not thread-safe.
- StringBuffer is synchronized. This means that multiple threads cannot call the methods of StringBuffer simultaneously.
- StringBuilder is asynchronized. This means that multiple threads can call the methods of StringBuilder simultaneously.
- StringBuffer was introduced in Java 1.0 whereas string builder was introduced in Java 1.5 **Can we access non static variable in static method**

Yes, we can throw the obj of class.

```java
public class Snippet
{
   private String instanceVariable;
   private static String staticVariable;

   public String instanceMethod()
   {
      return "instance";
   }

   public static String staticMethod()
   {
      return "static";
   }

   public static void main(String[] args)
   {
      System.out.println(staticVariable); // ok
      System.out.println(Snippet.staticMethod()); // ok

      System.out.println(new Snippet().instanceMethod()); // ok
      System.out.println(new Snippet().instanceVariable); // ok

      System.out.println(Snippet.instanceMethod()); // wrong
      System.out.println(instanceVariable);         // wrong
   }
}
```

ArrayList vs VectorList in java

S.No.	ArrayList	Vector
1.	ArrayList is a resizable or dynamic array.	Vectors are also a form of dynamic array.
2.	ArrayList is not synchronised.	Vector is synchronised.
3.	Syntax of ArrayList: ArrayList<T> al = new ArrayList<T>();	Syntax of Vector: Vector<T> v = new Vector<T>();
4.	If the number of elements overextends its capacity, ArrayList can increase the 50% of the present array size.	If the number of elements overextends its capacity, Vector can increase the 100% of the present array size, which means double the size of an array.
5.	It is not an inheritance class.	It is an inheritance class.
6.	It is faster than Vector.	It is slow as compared to the ArrayList.
7.	It is not a legacy class.	It is a legacy class.
8.	It prefers the Iterator interface to traverse the components.	It prefers an Enumeration or Iterator interface to traverse the elements.

What will happen if we put a key object in a HashMap which is already there ?

It will replace old mapping because Hash Map doesn't allow duplicate values.

Q11. What makes a HashSet different from a TreeSet?

HashSet	TreeSet
It is implemented through a hash table.	TreeSet implements SortedSet Interface that uses trees for storing data.
It permits the null object.	It does not allow the null object.
It is faster than TreeSet especially for search, insert, and delete operations.	It is slower than HashSet for these operations.
It does not maintain elements in an ordered way.	The elements are maintained in a sorted order.
It uses equals() method to compare two objects.	It uses compareTo() method for comparing two objects.
It does not permit a heterogenous object.	It permits a heterogenous object.

Q12. What are the differences between HashMap and HashTable in Java?

HashMap	Hashtable
It is non synchronized. It cannot be shared between many threads without proper synchronization code.	It is synchronized. It is thread-safe and can be shared with many threads.
It permits one null key and multiple null values.	It does not permit any null key or value.
is a new class introduced in JDK 1.2.	It was present in earlier versions of java as well.
It is faster.	It is slower.
It is traversed through the iterator.	It is traversed through Enumerator and Iterator.
It uses fail fast iterator.	It uses an enumerator which is not fail fast.
It inherits AbstractMap class.	It inherits Dictionary class.

How does Garbage Collection prevent a Java application from going out of memory?

Java Garbage Collector does not prevent a Java application from going out of memory. It simply cleans the unused memory when an object is out of scope and no longer needed. As a result, garbage collection is not guaranteed to prevent a Java app from going out of memory.

Is Java "pass-by-reference" or "pass-by-value"

Java is always "pass-by-value". However, when we pass the value of an object, we pass the reference to it because the variables store the object reference, not the object itself. But this isn‟t "pass-by-reference." This could be confusing for beginners.

Reflection in Java

Reflection is an API that is used to examine or modify the behavior of methods, classes, and interfaces at runtime. The required classes for reflection are provided under java.lang.reflect package.

Difference between Exception and Error in Java

Exceptions and errors both are subclasses of Throwable class. The error indicates a problem that mainly occurs due to the lack of system resources and our application should not catch these types of problems. Some of the examples of errors are system crash error and out of memory error. Errors mostly occur at runtime that's they belong to an unchecked type.

Exceptions are the problems which can occur at runtime and compile time. It mainly occurs in the code written by the developers. Exceptions are divided into two categories such as checked exceptions and unchecked exceptions.

Note: An exception is an unwanted or unexpected event, which occurs during the execution of a program that disrupts the normal flow of the program‟s instructions.

- **Checked Exceptions**

They occur at compile time. compiler checks for a these exception. and These exceptions can be handled at the compilation time.

Example of Checked exception- *„File Not Found Exception‟*

- **Unchecked Exceptions**

These exceptions occur at runtime. compiler doesn't check for a these kinds of exception. These kinds of exceptions can‟t be caught or handled during compilation time.

Example of Unchecked Exceptions- *'No Such Element Exception'*

Serialization and Deserialization

Serialization in Java allows us to convert an Object to stream that we can send over the network or save it as file or store in DB for later usage. Deserialization is the process of converting Object stream to actual Java Object to be used in our program. Serialization in Java seems very easy to use at first but it comes with some trivial security and integrity issues. **Serializable in Java**

If you want a class object to be serializable, all you need to do it implement the java.io.Serializable interface. Serializable in java is a marker interface and has no fields or methods to implement. It‟s like an Opt-In process through which we make our classes serializable.

Serialization in java is implemented by *ObjectInputStream* and *ObjectOutputStream*, so all we need is a wrapper over them to either save it to file or send it over the network. Let‟s see a simple Serialization in java program example.

Points to Note About Serialization in Java?

- Serialization is a marker interface with no method or data member
- You can serialize an object only by implementing the serializable interface
- All the fields of a class must be serializable; otherwise, use the transient keyword
- The child class doesn‟t have to implement the Serializable interface, if the parent class does
- The serialization process only saves non-static data members, but not static or transient data members
  - By default, the String and all wrapper classes implement the Serializable interface.

How to Serialize an Object?

You must use the writeObject() method of the ObjectOutputStream class for serialization and readObject() method of the InputObjectStream class for deserialization purpose.

```
package com.rohitShukla.serialization; import

java.io.Serializable; public class Employee

implements Serializable {                private

String name;    private int id;  transient private

int salary;

        @Override

        public String toString(){

        return "Employee{name="+name+",id="+id+",salary="+salary+"}";

        }

        public String getName() {

                return name;

        }

        public void setName(String name) {

                this.name = name;

        }

        public int getId() {

                return id;

        }

        public void setId(int id) {

                this.id = id;

        }

        public int getSalary() {

                return salary;

        }

        public void setSalary(int salary) {

                this.salary = salary;

        }
```

}

Notice that it‟s a simple java bean with some properties and getter-setter methods. If you want an object property to be not serialized to stream, you can use **transient** keyword like I have done with salary variable.

During serialization, JVM ignores all transient fields. If we need to exclude specific object fields during serialization, mark them as transient.

***Note*:** If we serialize an object which didn‟t implement the Serializable interface, Java throws java.io.NotSerializableException.

Externalizable interface in Java

Externalization serves the purpose of custom Serialization, where we can decide what to store in stream.

Externalizable interface present in java.io, is used for Externalization which extends Serializable interface. It consist of two methods which we have to override to write/read object into/from stream which are-

Key differences between Serializable and Externalizable

- **Implementation :** Unlike Serializable interface which will serialize the variables in object with just by implementing interface, here we have to explicitly mention what fields or variables you want to serialize.
- **Methods :** Serializable is marker interface without any methods. Externalizable interface contains two methods: writeExternal() and readExternal().
- **Process:** Default Serialization process will take place for classes implementing Serializable interface. Programmer defined Serialization process for classes implementing Externalizable interface.
- **Backward Compatibility and Control:** If you have to support multiple versions, you can have full control with Externalizable interface. You can support different versions of your object. If you implement Externalizable, it's your responsibility to serialize super class.
- **public No-arg constructor:** Serializable uses reflection to construct object and does not require no arg constructor. But Externalizable requires public no-arg constructor.

Below is the example for Externalization-

// to read object from stream

void readExternal(ObjectInput in)

// to write object into stream

void writeExternal(ObjectOutput out)

Read it for more: https://www.coderscampus.com/java-serialization/

Cloneable in Java

Cloneable is an interface that is used to create the exact copy of an object. It exists in java.lang package. A class must implement the Cloneable interface if we want to create the clone of the class object.

The clone() method of the Object class is used to create the clone of the object. However, if the class doesn't support the cloneable interface, then the clone() method generates the CloneNotSupportedException.

We can also create a copy of an object by using the new keyword, but it will take a lot of processing time. Therefore, using the clone() method is efficient for this purpose. Consider the following example to create a copy of an object using the clone() method.

Memory Allocations available in Java?

Java has five significant types of memory allocations.

1. Class Memory
2. Heap Memory
3. Stack Memory
4. Program Counter-Memory
5. Native Method Stack Memory

What are the differences between Heap and Stack Memory in Java?

Stack is generally used to store the order of method execution and local variables. In contrast, Heap memory is used to store the objects. After storing, they use dynamic memory allocation and deallocation.

Check out: https://www.simplilearn.com/tutorials/java-tutorial/java-interview-questions

Java String Function

String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created.

- in java, strings are treated as objects.

There are two ways to create string in Java:

- String literal

String s = “rohitShukla”;

- Using new keyword

String s = new String (“rohitShukla”);

Post: https://www.interviewbit.com/java-string-interview-questions/

Post: https://www.programiz.com/java-programming/library/string

String Methods

split()

The split() method divides the string at the specified regex and returns an array of substrings. We can pass the parameter in it. Example:

string.split(String regex, int limit)

The string split() method can take two parameters:

1. regex - the string is divided at this regex (can be strings)
2. limit (optional) - controls the number of resulting substrings
3. If the limit parameter is not passed, split() returns all possible substrings.

Note: If the regular expression passed to split() is invalid, the split() method raises *PatternSyntaxExpression* exception.

```
public static void main(String[] args) {
    String txt = "Hello My name is Rohit shukla";
    String[] m = txt.split(" ",3);
    for (String k : m) {
        System.out.println(k+", ");
    }
}
```

More: https://www.programiz.com/java-programming/library/string/split

compareTo() / compareToIgnoreCase

The compareTo() method compares two strings lexicographically (in the dictionary order). The comparison is

based on the Unicode value of each character in the strings.

compareTo() Return Value

1. returns 0 if the strings are equal

2. returns a negative integer if the string comes before the str argument in the dictionary order 3. returns a positive integer if the string comes after the str argument in the dictionary order

```java
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    String str2 = "Learn Java";
    String str3 = "Learn Kolin";
    int result;

    // comparing str1 with str2
    result = str1.compareTo(str2);
    System.out.println(result);  // 0

    // comparing str1 with str3
    result = str1.compareTo(str3);
    System.out.println(result);  // -1

    // comparing str3 with str1
    result = str3.compareTo(str1);
    System.out.println(result);  // 1
  }
}
```

compareTo() takes the letter case into consideration. It means **learn** and **Learn** will be treat differently. But if

```java
class Main {
  public static void main(String[] args) {
    String str1 = "java is fun";

    // extract substring from index 0 to 3
    System.out.println(str1.substring(0, 4));
  }
}

// Output: java
```

you want to ignore this case, mean you want to treat **learn** and **Learn** same so you can use compareToIgnoreCase() and except this things as same as compareTo()

Sytax: string.compareTo(String str)

More: https://www.programiz.com/java-programming/library/string/compareto

More: https://www.programiz.com/java-programming/library/string/comparetoignorecase

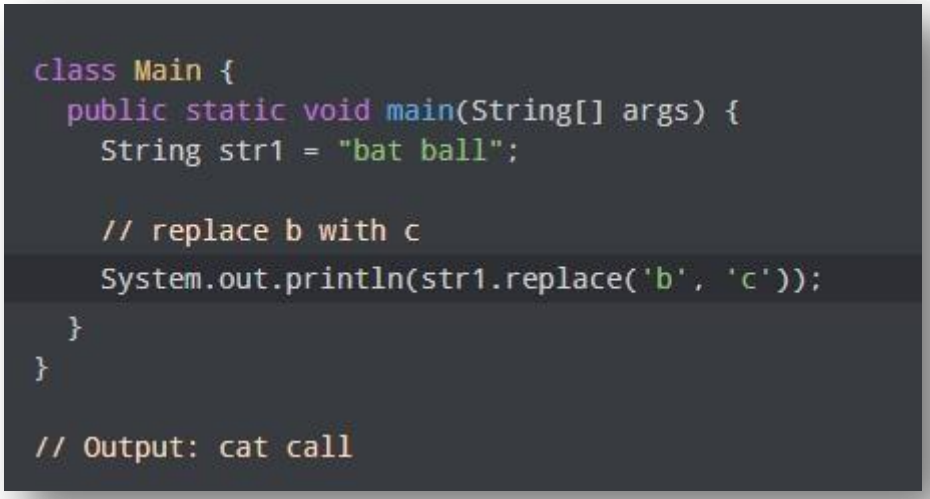
length()

The length() method returns the length of the string. this method doesn't take any parameters.

```
Syntax: string.length()
```

More: https://www.programiz.com/java-programming/library/string/length

replace() / replaceAll()



The replace() method replaces each matching occurrence of the old character/text in the string with the new character/text.

```
Syntax: string.replace(char oldChar, char newChar)
```

it can replace single character and substring both.

More: https://www.programiz.com/java-programming/library/string/replace

Replace() didn't support regular expression (regex) but replaceAll() supports. Syntax is same of both.

```
string.replaceAll(String regex, String replacement)
```

\\d use this regex to remove of numeric value from string.

\\s use this regex to remove of white space from string.

```java
class Main {
  public static void main(String[] args) {
    String str1 = "Learn\nJava \n\n   ";
    String result;

    // replace all whitespace characters with empty string
    result = str1.replaceAll("\\s", "");
    System.out.println(result);   // LearnJava
  }
}
```

replace() method is a bit faster than replaceAll() due to regex.

substring()

The substring() method return the substring on that correct index number. And we can also pass start Index & end index then substring then output will be substring between both indexes. We can also paas one index value. Java treats it like a start index.

```
string.substring(int startIndex, int endIndex)
```

```java
class Main {
  public static void main(String[] args) {
    String str1 = "program";

    // from the first character to the end
    System.out.println(str1.substring(0));  // program

    // from the 4th character to the end
    System.out.println(str1.substring(3));  // gram
  }
}
```

Example

--- and ---

More: https://www.programiz.com/java-programming/library/string/substring

equals()

The equals() method returns true if two strings are equal. If not, it returns false.

it returns true if the strings are equal and return false if strings are not equal and the str argument is null. It treats **learn** and **Learn** different. If you want to ignore this case then use **equalsIgnoreCase()** in place of **equals()** .

trim()

The trim() method removes any leading (starting) and trailing (ending) whitespaces from the specified string. It doesn't remove whitespace that appears in the middle. it doesn't take any parameters.

If you need to remove all whitespace characters from a string, you can use the String replaceAll() method with proper regex. Like this

```
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    Boolean result;

    // check if str1 contains "Java"
    result = str1.contains("Java");
    System.out.println(result);  // true

    // check if str1 contains "Python"
    result = str1.contains("Python");
    System.out.println(result);  // false

    // check if str1 contains ""
    result = str1.contains("");
    System.out.println(result);  // true
  }
}
```

contains()

The contains() method checks whether the specified string (sequence of characters) is present in the string or not. If string is present then it return “true” and string is not present then it return “False”. It treats **learn** and **Learn** different. **indexOf()**

The indexOf() method returns the index of the specified character or substring which is presented in the string. It return output in numeric.

We can pass two parameter in this. Character/substring and index number. Index number is optional but if we pass then searching will start from that index only. Character/substring is mandatory whose we want to search. It treats **learn** and **Learn** different. it return -1 when character/subString is not found. substring() is just opposite of it.

String s = ”Learn Share Learn”;

int output = s.indexOf(“Share”); // returns 6

String s = ”Learn Share Learn”;

int output = s.indexOf("ea",3);// returns 13

lastIndexOf()

we it work from last of the string and return the specific index of the substring/character.

charAt()

The charAt() method returns the character at the specified index. Here we pass the index number and it return the character which is located on that index in String.

```
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    String str2 = "Learn\nJava";

    // first character
    System.out.println(str1.charAt(0));  // 'L'

    // seventh character
    System.out.println(str1.charAt(6));  // 'J'

    // sixth character
    System.out.println(str2.charAt(5));  // '\n'
  }
}
```

toLowerCase()

The toLowerCase() method converts all characters in the string to lowercase characters.

```
class Main {
  public static void main(String[] args) {
    String str1 = "Learn Java";
    String str2 = "Java123";

    // convert to lowercase letters
    System.out.println(str1.toLowerCase());  // "learn java"
    System.out.println(str2.toLowerCase());  // "java123"
  }
}
```

concat()

The concat() method concatenates (joins) two strings and returns it.

The concat() method takes a single parameter. In Java, you can also use the + operator to concatenate two strings.

```
class Main {
  public static void main(String[] args) {
    String str1 = "Java";
    String str2 = "Programming";
    // concatenate str1 and str2

    System.out.println(str1.concat(str2));
```

```
  }
}
```

```
// Output: JavaProgramming
```

concat() Vs the + Operator for Concatenation

concat()	the + Operator
Suppose, **str1** is `null` and **str2** is `"Java"`. Then, `str1.concat(str2)` throws **NullPointerException**.	Suppose, **str1** is `null` and **str2** is `"Java"`. Then, `str1 + str2` gives **"nullJava"**.
You can only pass a String to the `concat()` method.	If one of the operands is a string and another is a non-string value. The non-string value is internally converted to a string before concatenation. For example, `"Java" + 5` gives `"Java5"`.

valueOf()

The valueOf() method returns the string representation of the argument passed. The valueOf() method takes a single parameter. we can convert char and char array to String by to value() method. we have to just pass array name in parameter.

```
class Main { public static void main(String[] args) {
int a = 5;     long l = -2343834L;     float f = 23.4f;
double d = 923.234d;

    // convert numbers to strings
    System.out.println(String.valueOf(a));  // "5"



    System.out.println(String.valueOf(l));  // "-2343834"
    System.out.println(String.valueOf(f));  // "23.4"
    System.out.println(String.valueOf(d));  // "923.234"
  }
}
```

we can also pass array, offset, length in parameter. syntax should be like this -->

syntax: valueOf**(**char[] data, int offset, int length**)** offerset: it will left this number

of elemnt from starting and continue to work after that.

length: it mean output/ string length

```
class Main {   public static void main(String[] args) {     char ch[]
= {'p', 'r', 'o', 'g', 'r', 'a', 'm'};     int offset = 2;
    int length = 4;
    String result;


    // subarray {'o', 'g', 'r', 'm'} is converted to string
    result = String.valueOf(ch, offset, length);

    System.out.println(result);  // "ogrm"
  }
}
```

matches()

The matches() method checks whether the string matches the given regular expression or not. and give the output in true /false

More: https://www.programiz.com/java-programming/library/string/matches

startsWith()

As it's name shows the startsWith() method checks whether the string begins with the specified string or not. and give the output in true and false. we can also pass the offset along with Stringt. **syntax**: string.startsWith**(**String str, int offset**)** offset mean as same here as I earlier told.

More: https://www.programiz.com/java-programming/library/string/startswith

endsWith()

The Java String endsWith() method checks whether the string ends with the specified string or not.

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

    String str = "Java Programming";

    System.out.println(str.endsWith("mming")); // true
    System.out.println(str.endsWith("a Programming")); // true
    System.out.println(str.endsWith("Java")); // false
  }
}
```

isEmpty()

The Java String isEmpty() method checks whether the string is empty or not. The isEmpty() method does not take any parameters. it give output in true and false.

intern()

Suppose we are intializing 2 variable and declaring same string in both of them. now they will capture the memory diffently or only one time bcoz value of both variable is same. obviously they will captue the memory

diffrently bcoz pool of both value is diffrent but this looks weird bcoz the value of both varibale is same to same then we should they are not covering onr space?

str1 and str2 have the same content. However, they are not equal because they don't share the same memory. you can manually use the intern() method so that the same memory is used for strings having the same content. Now they will use same string pool.

hashCode()

The Java String hashCode() method returns a hash code for the string. A hashcode is a number (object's memory address) generated from any object, not just strings. This number is used to store/retrieve objects quickly in a hashtable.

join()

The join() method returns a new string with the given elements joined with the specified delimiter.

Syntax: String.join**(**CharSequence delimiter, CharSequence...

elements**)** Here, ... signifies there can be one or more CharSequence.

```
class Main {
  public static void main(String[] args) {
    String str1 = "I";
    String str2 = "love";
    String str3 = "Java";

                                                    str2, str3);
    // join strings with space between them
    String joinedStr = String.join(" ", str1,

    System.out.println(joinedStr);
  }
}

// Output: I love Java
```

//++++++++++++++++++++

```
import java.util.ArrayList;

class Main {
  public static void main(String[] args) {
    ArrayList<String> text = new ArrayList<>();

    // adding elements to the arraylist
text.add("Java");     text.add("is");
text.add("fun");

    String result;

    result = String.join("-", text);

    System.out.println(result);  // Java-is-fun

  }
}
```

replaceFirst()

The Java String replaceFirst() method replaces the first substring that matches the regex of the string with the specified text.

Syntax: string.replaceFirst**(**String regex, String replacement**)**

subSequence()

The Java String subSequence() method returns a character sequence (a subsequence) from the

string. **Syntax:** string.subSequence**(**int startIndex, int endIndex**)**

```
class Main {
  public static void main(String[] args) {
    String str = "Java Programming";

    System.out.println(str.subSequence(3, 8)); // a Pro
  }
}
```

toCharArray()

The Java String toCharArray() method converts the string to a char array and returns it. this method doesn't take any parameters.

Syntax: string.toCharArray()

format()

The format() method returns a formatted string based on the argument passed.

More: https://www.programiz.com/java-programming/library/string/format

Important Points about String in java

Strings are derived data types in java. Derived data types are those that are made by using any other data type for example, arrays.

Java treats Strings as objects, not arrays. String objects are created using the java.lang.String class. String objects in Java are immutable; you cannot modify their contents. This means whenever we manipulate a String object, the new String is created rather than the original string being modified.

Strings are immutable in Java. Immutable objects mean they can't be changed or altered once they've been created. However, we can only modify the reference to the string object. The String is immutable in Java because of many reasons like security, caching, synchronization and concurrency, and class loading. we can use a string in the switch case in java.

In Java, every immutable object is thread-safe, which means String is also thread-safe.

String pool in Java.

String Pool, also known as SCP (String Constant Pool), is a special storage space in Java heap memory that is used to store unique string objects. Whenever a string object is created, it first checks whether the String object with the same string value is already present in the String pool or not, and if it is available, then the reference to the string object from the string pool is returned. Otherwise, the new string object is added to the string pool, and the respective reference will be returned. **intern() method do in Java?**

If you apply the intern() method to a few strings, you will ensure that all strings having the same content share the same memory in string pool

Difference between StringBuffer and StringBuilder in Java.

StringBuffer and StringBuilder are two Java classes for manipulating strings. These are mutable objects, i.e., they can be modified, and provide various methods such as insert(), substring(), delete(), and append(), for String manipulation.

StringBuffer: The StringBuffer class was created by the Java Team when they realized the need for an editable string object. Nevertheless, StringBuffer has all methods synchronized, meaning they are thread-safe. Therefore, StringBuffer allows only one thread to access a method at once, so it is not possible to call StringBuffer methods from two threads simultaneously, which means it takes more time to access. The StringBuffer class has synchronized methods, making it thread-safe, slower, and less efficient than StringBuilder. The StringBuffer class was introduced in Java 1.0.

Syntax:

StringBuffer var = new StringBuffer(str);

StringBuilder: It was at that point that the Java Team realized that making all methods of StringBuffer synchronized wasn't the best idea, which led them to introduce StringBuilder. The StringBuilder class has no synchronized methods. Unlike StringBuffer, StringBuilder does not offer synchronized methods, which makes it less thread-safe, faster, and more efficient. StringBuilder was introduced in Java 1.5 in response to StringBuffer's shortcomings.

Syntax:

StringBuilder var = new StringBuilder(str);

Difference between str1 == str2 and str1.equals(str2)?

Java offers both the equals() method and the "==" operator for comparing objects. However, here are some differences between the two:

1) Essentially, equals() is a method, while == is an operator.
2) The == operator can be used for comparing references (addresses) and the .equals() method can be used to compare content. To put it simply, == checks if the objects point to the same memory location, whereas .equals() compares the values of the objects.

```
public class StringComparison
{
   public static void main(String[] args)
   {
       String str1=new String("Scaler");
       String str2=new String("Scaler");
       System.out.println(str1 == str2);
       System.out.println(str1.equals(str2));
   }
}
```

Output:

```
false
true
```

How can a Java string be converted into a byte array?

The getBytes() method allows you to convert a string to a byte array

How do you convert a string to an integer and vice versa?

There is an Integer class in the Java lang package that provides different methods for converting strings to integers and vice versa.

```
public class StringtoInteger {
  public static void main(String args[])
  {
    String str1 = "1296";
```

```
    int i= Integer.parseInt(str1);
    System.out.println(i);
    String str2 = Integer.toString(i);
    System.out.println(str2);

}
}
```

How can we convert string to StringBuilder?

The append() method can be used to convert String to StringBuilder, and the toString() method can be used to convert StringBuilder to String.

```
public class StringToStringBuilder {

   public static void main(String args[]) {

      String strs[] = {"Scaler", "by", "InterviewBit!" };

      StringBuilder sb = new StringBuilder();

      sb.append(strs[0]);

      sb.append(" " + strs[1]);

      sb.append(" " + strs[2]);

      System.out.println(sb.toString());

   }

}
```

How do you check whether a String is empty in Java?

java provide isEmpty() method to check whether a string is empty. if output is true it means string is empty else false.

String Pool

String Pool is a pool of Strings stored in Java heap memory. We know that String is a special class in Java and we can create String object using new operator as well as providing values in double quotes.