Top 100+ Java Interview Questions and Answers in 2023

By Naveen 6.7 K Views 51 min read Updated on June 13, 2023

These are expert-created Java interview questions and answers to help you excel in your Java interview. Here you will learn Java advantages, inheritance, class, enumeration, iterator, JSON, abstract class, transient and volatile variables, HTTP tunneling, hibernate in Java, and more. Learn Java from Intellipaat **Java Training** to excel in your career!

Categories

Automation	3
Big Data	12
Business Intelligence	21
Cloud Computing	21
Cyber Security	2
Data Science	11
Database	8
Digital Marketing	3
Electric Vehicle	1
Investment Banking	1
Mobile Development	2
No-SQL	5
Programming	15
Project Management	6
Salesforce	3
Testing	4

UI UX	2
Website Development	8

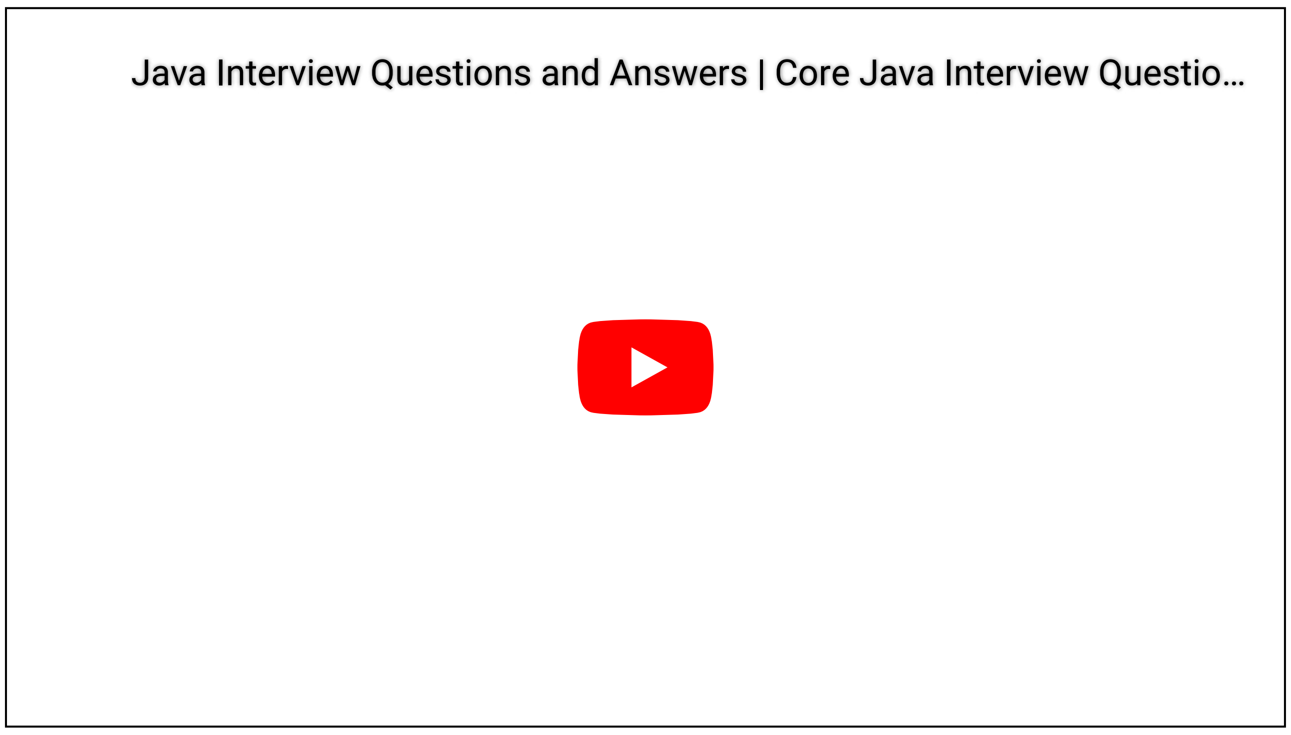
Top Java Interview Questions and Answers

Java is one of the widely used programming languages in the industry. There are more than 3 billion devices that use Java for their software development. A plethora of job opportunities are available for Java Developers. In this Core Java Interview Questions blog, you will go through the top Java interview questions that you will come across in Java interviews. Let us have a glance at a few of the important Java programming interview questions here:

Frequently Asked Java Interview Questions

This blog on Java Interview Questions is categorized into three parts as mentioned below:

Watch this Java Interview Questions video:


Java Interview Questions and Answers | Core Java Interview Questio…

Java Interview Questions and Answers for Freshers

These are the basic Java interview questions for freshers to crack their interview:

1. What do you understand by Java?

- Java is an object-oriented computer language.
- It is a high-level programming language developed by James Gosling in Sun Microsystems in the year 1995.
- Java is a fast, secure, and reliable language used for many games, devices, and applications.

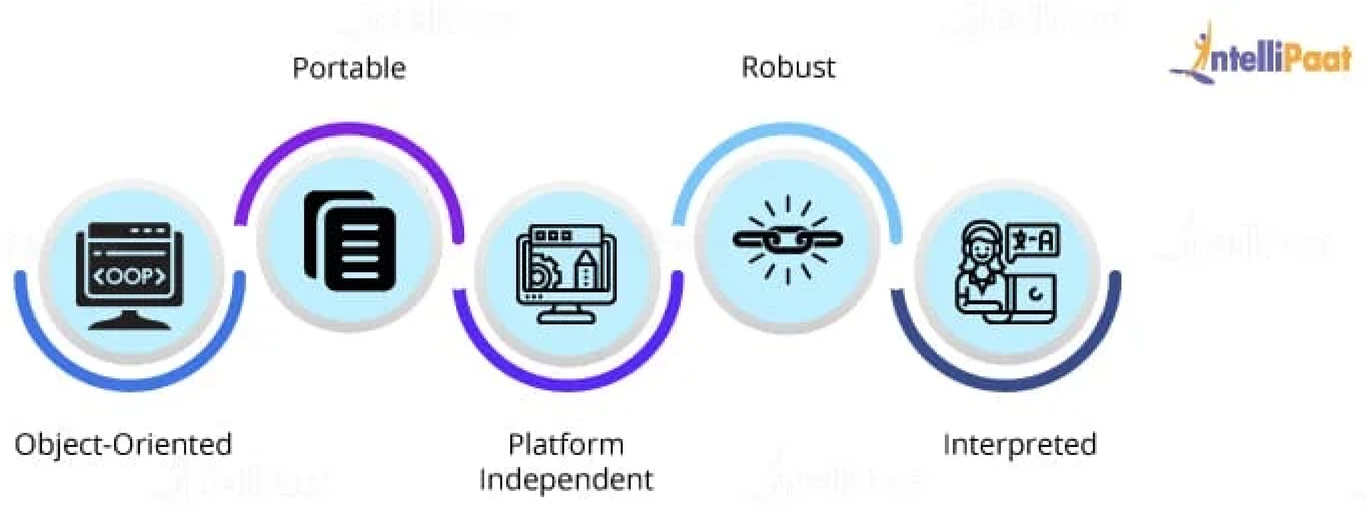
Go through this Java Tutorial to get a better understanding of the concept!

2. Compare between Java and Python.

Criteria	Java	Python
Ease of use	Good	Excellent
Speed of coding	Average	Excellent
Data types	Static typed	Dynamically typed
Data Science and Machine Learning applications	Average	Excellent

3. Outline the major Java features.

The major features of Java programming language are explained below:



- **Object-oriented:** Java is based on object-oriented programming where the class and methods describe the state and behavior of an object.
- **Portable:** A Java program gets converted into Java bytecodes that can be executed on any platform without any dependency.
- **Platform independent:** Java works on the 'write once, run anywhere' principle as it supports multiple platforms like Windows, Linux, Mac, Sun Solaris, etc.

- **Robust:** Java has strong memory management as there are no pointer allocations. It has an automatic garbage collection that prohibits memory leaks.
- **Interpreted:** As mentioned, Java compiler converts the codes into Java bytecodes which are then interpreted and executed by Java Interpreter.

Become a master of Java by enrolling in this online Java Course!

4. What do you mean by an object?

An object consists of methods and classes that depict its state and perform operations. A Java program contains a lot of objects instructing each other their jobs. This concept is part of core Java.

5. Distinguish between StringBuffer and StringBuilder in Java programming.

StringBuffer	StringBuilder
StringBuffer methods are synchronized.	StringBuilder is non-synchronized.
The storage area is heap and modified easily.	Storage is heap-based and can be modified.
StringBuffer is thread-safe.	StringBuilder is fast as it is not thread-safe.
The performance is very slow.	The performance is very fast.

Go through this tutorial to get a better understanding of Java String!

6. Differentiate between JDK, JRE, and JVM.

- **JVM** stands for Java Virtual Machine which provides the runtime environment for Java bytecodes to be executed.
- **JRE** (Java Runtime Environment) includes the sets of files required by JVM during runtime.
- **JDK** (Java Development Kit) consists of JRE along with the development tools required to write and execute a program.

Get a clear idea of why to get certified in Java!

7. Define inheritance.

Java includes the feature of inheritance which is an object-oriented programming concept. Inheritance lets a derived class inherit the methods of a base class.

8. Explain method overloading.

When a Java program contains more than one method with the same name but with different properties, then it is callec method overloading.

9. Compare overloading with overriding.

Overloading refers to the case of having two methods of the same name but different properties; whereas, overriding occurs when there are two methods of the same name and properties, but one is in the child class and the other is in the parent class.

Read this tutorial to get a clear understanding of Java!

10. Explain the creation of a thread-safe singleton in Java using double-checked locking.

Singleton is created with the double-checked locking as before Java 5 acts as a broker and it's been possible to have multiple instances of singleton when multiple threads create an instance of the singleton at the same time. Java 5 made it easy to create thread-safe singleton using Enum. Using a volatile variable is essential for the same.

11. What is a class in Java?

Java encapsulates codes in various classes that define new data types. These new data types are used to create objects.

12. Differentiate between an ArrayList and a Vector.

ArrayList	Vector
An ArrayList is not synchronized.	A vector is synchronized.
An ArrayList is fast.	A vector is slow as it is thread-safe.
If an element is inserted into an ArrayList, it increases its array size by 50 percent.	A vector defaults to doubling the size of its array.
An ArrayList does not define the increment size.	A vector defines the increment size.
An ArrayList can only use Iterator for traversing.	Except for hashtable, a vector are the only other class that uses both Enumeration and Iterator.

Career Transition

13. Mention the difference between Iterator and Enumeration.

Iterator	Enumeration
Iterator is an interface found in the java.util package.	Enumeration is an object that generates elements one at a time.
Uses three methods to interface: 1. hasNext() 2. next() 3. remove()	Uses two methods: 1. hasMoreElements() 2. nextElement()

Iterators allow removing elements from the given collection during the iteration with well-defined semantics.	It is used for passing through a collection, usually of unknown size.
Iterator method names have been improved.	The traversing of elements can only be done once per creation.

14. Explain the difference between the inner class and the subclass.

Inner Class	Subclass
An inner class is a class that is nested within another class.	A subclass is a class that inherits from another class called the superclass.
It provides access rights for the class, which is nesting it, which can access all variables and methods defined in the outer class.	It provides access to all public and protected methods and fields of its superclass.

15. Can we execute any code, even before the main method? Explain.

Yes, we can execute any code, even before the main method. We will be using a static block of code in the class when creating the objects at load time of the class. Any statements within this static block of code will get executed at once while loading the class, even before the creation of objects in the main method.

Prepare yourself for Java certification with our comprehensive online Java Training!

16. How can we restrict inheritance for a class?

We can restrict inheritance for a class by the following steps:

1. By using the final keyword
2. If we make all methods final, then we cannot override that
3. By using private constructors
4. By using the Javadoc comment (" // ")

17. Java doesn't support multiple inheritance. Why?

Java doesn't support multiple inheritance because we cannot use different methods in one class; it creates an ambiguity.
Example:

```
class Intellipaat1
{
void test()
{
system.out.println("test() method");
}
}class Intellipaat2
{
void test()
{
system.out.println("test() method");
}
}Multiple inheritance
class C extends Intellipaat1, Intellipaat2
{
     /* Code */
}
```

Intellipaat1 and Intellipaat2 test() methods are inheriting to class C. So, which test() method class C will take?

As Intellipaat1 and Intellipaat2 class test () methods are different, here we would face ambiguity.

18. Are constructors inherited? Can a subclass call the parent's class constructor?

We cannot inherit a constructor. We create an instance of a subclass using a constructor of one of its superclasses. Because overriding the superclass constructor is not our wish as if we override a superclass constructor, then we will destroy the encapsulation abilities of the language.

Check out this insightful tutorial to learn more about Java Constructors!

19. Define JSON.

The expansion of JSON is 'JavaScript Object Notation.' It is a much lighter and readable alternative to XML. It is independent and easily parse-able in all programming languages. It is primarily used for client–server and server–server communication.

20. What are the advantages of JSON over XML?

The advantages of JSON over XML are:

1. JSON is lighter and faster than XML.
2. It is easily understandable.
3. It is easy to parse and convert to objects for information consumption.
4. JSON supports multiple data types—string, number, array, or Boolean—but XML data are all strings.

Courses you may like

21. What is the difference between Java and C++?

Below are the key differences between Java and C++:

Java	C++
Java supports both compilers and interpreters	Supports only the compiler
Memory management is system controlled	Memory management can be accessed by the programmer
Doesn't support multiple inheritances	Support multiple inheritances
Offers limited support for the pointers	Fully supports the pointers
Shows heavy dependence on automatic garbage collection, but does not support the destructors	The user needs to perform manual management using the new and delete keywords

22. What is JIT Compiler?

The Just-In-Time compiler is an important part of the Java Runtime Environment that optimizes the performance of java based applications at run time. JIT parallelly compiles the bytecodes with similar functionality and reduces the compilation time.

A method in the java program can be called thousands of times at the startup. Therefore, instead of compiling every method, java OpenJ9 records the number of times a method is initiated. If the initiation count crosses a predefined threshold, the JIT compiler gets triggered.

Even learn about Queue in Java.

23. What is Classloader?

It is a part of the Java Runtime Environment which is used to dynamically load the class files into the Java Virtual Machine(JVM). The classloader triggers whenever a java file is executed.

There are three types of classloaders in Java:

- **Bootstrap Classloader:** The bootstrap loader is the superclass of extension classloader which loads the standard JDK files from rt.jar and other core classes. The rt.jar file contains all the package classes of java.net, java.lang, java.util, java.io, java.sql, etc.
- **Extension Classloader:** The extension classloader is known as the parent of the System classloaders and the child of the Bootstrap classloader. It only loads the files available in the extension classpath, that is '***ext.dirs/JRE/lib/ext*** directory. If the file is not available in the mentioned directory the extension classloader will delegate it to the System classloader.
- **System/Application Classloader:** It loads the classes of application type available in the environment variable such as ***-classpath, or -cp command-line option***. The application classloader generates a **ClassNotFoundException** if the class is not available in the application classpath.

24. Is an empty .java file name a valid source file name in java?

Yes, we can save a java file with an empty .java file name. You can compile it in the command prompt by 'javac .java' and run it by typing 'java classname'. Here is an example of such a program:

```
import java.util.*;
class A
{
        public static void main(String args[])
        {
                System.out.println("Hello World");
        }
}
```

After saving the code, you need to open the command prompt and go to the directory where the file is stored. Then, you need to type 'javac .java' to compile the program, and 'java A' to run the program.

25. What is the difference between Object-oriented and object-based programming language?

Following are the key differences between the object-oriented and object-based programming languages:

Object-oriented programming language	Object-based programming language
Object-oriented programming languages support all the features of OOPS, including polymorphism and inheritance	Object-based programming language does not support inheritance or polymorphism
Doesn't support built-in objects	Support built-in objects like JavaScript has the window object
Example: C#, C++, Java, Python, etc.	JavaScript and Visual Basic are examples of object-based programming language

26. How many types of constructors are there in the Java programming language?

There are two types of constructors in the Java programming language:

- **Parameterized Constructor**: A constructor that accepts arguments or parameters is known as the parameterized constructor. It initializes the instance variables will the values passed while creating an object.

Below is an example of the parameterized constructor:

```
import java.util.*;
class A
{
int roll_no;
String name;
A()
{
System.out.println("Constructor called");
}

public void fun()
{
System.out.println("Class A function called");
}
}
public class parameterized_constructor
{
public static void main(String args[])
{
A obj=new A();
obj.fun();
System.out.println("The value of instance variables are: ");
System.out.println("Name: "+obj.name);
System.out.println("Roll No: " +obj.roll_no);
}
}
The output of the above program would be:
Constructor called
Class A function called
The value of instance variables are:
Name: null
Roll No: 0
```

- **Non-parameterized/Default constructor**: The constructors with no parameters or arguments are known and non-parameterized constructors. In case you don't define a constructor of any type, the compiler itself creates a default constructor which will be called when the object is created.

Below is an example of a non-parameterized constructor:

```
import java.util.*;
class A
{
    int roll_no;
    String name;
    A(int x,String y)
    {
        this.roll_no=x;
        this.name=y;
        System.out.println("Constructor called");
    }

    public void fun()
    {
        System.out.println("Class A function called");
    }
}

public class non_parameterized_constructor
{
    public static void main(String args[])
    {
        A obj=new A(10,"Harry Williams");
        System.out.println("The value of instance variables");
        System.out.println("Name: "+obj.name);
        System.out.println("Roll No: "+obj.roll_no);
    }
}

The output of the above program will be:

Constructor called
The value of instance variables
Name: Harry Williams
Roll No: 10
```

27. What is the purpose of a Default constructor in Java?

If no constructor is available in the class, the java compiler creates a default constructor and assigns the default value to the instance variables.

For example:

```
import java.util.*;
class A
{
    String name;
    int id;
    void display()
    {
        System.out.println("Name: "+this.name);
        System.out.println("Id: "+this.id);
    }

}
public class B
{
    public static void main(String args[])
    {
        A object=new A();
        System.out.println("The values of the instance variables are:");
        System.out.println("Name: "+object.name);
        System.out.println("Name: "+object.id);
    }
}
The output of the above program:
The values of the instance variables are:
Name: null
Name: 0
```

28. What is the use of a copy constructor in Java?

Java does not support a copy constructor, however, you can copy the values of one instance to another by copying the instance variables of one object to another. Below is an example of such a method:

```
import java.util.*;
class demo
{
    String name;
    int roll_no;
    //constructor to initialize the name and roll number
    demo(int x, String y)
    {
        name=y;
        roll_no=x;
    }
    //constructor to initialize another object variables
    demo(demo ob)
    {
        roll_no=ob.roll_no;
        name=ob.name;
    }
    void display()
    {
        System.out.println("Name: "+name);
        System.out.println("Roll Number:"+roll_no);
    }
}

public class copy_constructor
{
    public static void main(String args[])
    {
        demo obj=new demo(10,"Alice Williams");
        demo obj2=new demo(obj);
        obj.display();
        obj2.display();
    }
}

The output of the following code is:
Name: Alice Williams
Roll Number:10
Name: Alice Williams
Roll Number:10
```

29. Why is the main method static in Java?

The main method is static in Java because to call the static methods, there is no need for the object. The Java Virtual Machine(JVM) has to create an object to call the non-static main() method, which will result in extra memory allocation.

30. Can we declare the static variables and methods in an abstract class?

Yes, we can declare the static variables and methods in an abstract class by inheriting the abstract class. In java, the static variables and methods can be accessed without creating an object. You can simply extend the abstract class and use the static context as mentioned below:

```
import java.util.*;
class A
{
    static int x=110;
    static void method1()
    {
        System.out.println("Class A static function called");
    }

}

public class prog1 extends A
{
    public static void main(String args[])
    {
        A.method1();
        System.out.println("Value of x="+A.x);
    }
}

The output of the above program will be:
Class A static function called
Value of x=110
```

31. What is Aggregation?

Aggregation in Java represents Has-A relationship where one class contains the reference of another class. In other words, it refers to a one-way relationship between two classes where the aggregate class has the instance of another class it owns.

For example, in the program below the aggregate class student have a one-way relationship with the class Address. Here each student having an address makes sense, but address having the student doesn't make any sense. So, instead of inheritance, the aggregation is based on the usage of the classes.

```
import java.util.*;
class Address
{
    String street, city, state, country;
    public Address(String street, String city, String state, String country)
    {
        this.street=street;
        this.city=city;
        this.state=state;
        this.country=country;
    }
}

public class student
{
    int reg_no;
    String name;
    Address address;
    public student(int id,String name,Address address)
    {
        this.reg_no=id;
        this.name=name;
        this.address=address;
    }
    public void display()
    {
        System.out.println("Student details: ");
        System.out.println("Name: "+name);
        System.out.println("Reg_no: "+reg_no);

        System.out.println("Address: "+address.street+", "+address.city+", "+address.state+",
"+address.country);
    }

    public static void main(String args[])
    {
        Address address1=new Address("Paper Mill Road","SRE","UP","India");
        Address address2=new Address("Service Road","BNG","KA","India");

        student ob1=new student(111,"Sam",address1);
        student ob2=new student(112,"Alex",address2);

        ob1.display();
        ob2.display();
    }
}
```

The output of the above program will be:

```
Student details:
Name: Sam
Reg_no: 111
Address: Paper Mill Road, SRE, UP, India
```

```
Student details:
Name: Alex
Reg_no: 112
Address: Service Road, BNG, KA, India
```

32. What is composition?

If an object contains another object and the contained can't exist without the existence of that particular object, then it's known as the composition. In simple words, the composition is a kind of aggregation used to describe the reference between two classes using the reference variable.

For example, a class student contains another class named address. So, if the class address cannot exist without the class student, then there exists a composition between them.

33. What is the difference between aggregation and composition?

Below are the key differences between aggregation and composition in Java:

Aggregation	Composition
The parent class has a one-way relationship with the child class.	In composition, the parent class owns the child class
Aggregation is denoted by an empty diamond in UML (Unified Modeling Language) notation	Denoted by a filled diamond in the UML notation
Child class has its own life cycle	Child class doesn't have a lifetime
Aggregation is a weak relationship. For example, a bike with an indicator is aggregation	Composition is a stronger relationship. For example, a bike with an engine is a composition

34. What is object cloning?

Object cloning means creating the exact copy of an existing object. In java programming, object cloning can be done using the clone() method. To create the object clone, you should implement the java.lang.Cloneable interface, otherwise the clone() method generates a CloneNotSupportException.

The syntax for the clone method is:

```
protected Object clone() throws CloneNotSupportedException
```

35. Can we overload the main method in Java?

Yes, you can overload the main() method in Java using the concept of method overloading.

Below is an example of main() method overloading:

```
import java.util.*;
public class main_overloading
{
   public static void main(String args)
   {
       System.out.println("Main with String");
   }
   public static void main(String args[])
   {
       System.out.println("Main with the String args[]");
   }
   public static void main()
   {
       System.out.println("main without String");
   }
}
```

The output of the above code will be:

```
Main with the String args[]
```

36. Explain public static void main(String argos[]) in Java?

The main() method is the entry point for any program in java language. In the expression **public static void main(String argos[])**, each keyword has significance as mentioned below:

- **Public:** It's an access specifier which is used to specify the accessibility of a method in java. Here, Public means anyone can access the method or the class.
- **static:** the static keyword in java identifies the main method as class-based, which means it can be accessed without creating an object/instance. Static methods can b3e directly called by the Java Virtual Machine(JVM), which in turn saves extra memory allocation.
- **void:** void is a return type in java, which means the main method will not return any value back.
- **String args[]:** String argos[] or String[] args is an array of String objects which stores the command line arguments. Here the argument is an array of types of java. Therefore, it's passed as the parameter in the main() method.

37. What are various exception handling keywords in Java?

Java language has three exception handling keywords:

- **try:** whenever a code segment has a chance of an error or any abnormality, it can be placed inside the try block. In case of any error, the try block will send the exception to the catch block.
- **catch:** catch handles the exception generated in the try block. It comes after the try block.
- **final:** The final keyword ensures that the segments execute despite the errors and exceptions processed by the try & catch block. The final keyword comes either after the try block or both the try and the catch block.

38. What is the NullPointerException?

The NullPointerException occurs when the user attempts to access or modify the fields of a null object. A null object is an instance defined with null values or without any reference.

39. What is the difference between a constructor and a destructor?

Below is the difference between a constructor and a destructor based on various parameters:

Base of Comparison	Constructor	Destructor
Purpose	A constructor is used to allocate the memory.	A destructor is used to deallocate or free up the memory.
Arguments	Mayor may not accept the arguments.	Destructors don't accept any arguments.

Declaration	class_name(arguments){
	
	}
		~class_name(){
		
		};

Calling	The constructor is called as soon as the object is created.	Called at the time of program termination or when the block is exited.
Order of execution	Constructors are executed in successive order, meaning from parent to child class.	Destructors are called in the reverse order, meaning from child to parent class.
Overloading	Can be overloaded.	Cannot be overloaded.

40. Can a Java interface have static methods?

Yes, a Java interface can have static methods, but we cannot override them as they're static. So, we have to define the static methods in the interface and only perform the calling party in the class.

Below is an example of a static method in Java interface:

```
import java.util.*;
public interface A
{
    static void staticmethod()
    {
        System.out.println("Static method of interface A called");
    }
    default void display()
    {
        System.out.println("Default function of interface A called");
    }
}
```

The above interface has a static method and a default method that can be executed without affecting the class that implements it. However, there is no use in creating the static methods in the Java interface as we cannot override the function by redefining it in the main() method.

```
import java.util.*;
the public class demo implements A
{
    public static void main(String args[])
    {
        demo obj =new demo();
        A.staticmethod();
        obj.display();
    }

}
```

So, the output of the above program would be:

Static method of interface A called
Default function of interface A called

41. How to manually throw an exception in Java programming?

The programmer has to use the throws keyword to create and throw a manual exception. Below is the program to manually throw a single IOException. If you want to throw:

```
import java.io.*;
class D
{
    void function1() throws ArithmeticException
    {
        throw new ArithmeticException("Arithmetic or zero value error");
    }
}

public class demoException
{
    public static void main(String args[]) throws ArithmeticException
    {
        D obj=new D();
        obj.function1();
        System.out.println("End of the program");
    }
}
```

The output of the above program will be:

```
java.lang.ArithmeticException: Arithmetic or zero value error
```

42. Differentiate between ‘==’ and equals()?

The main difference between ‘==’ and equals() is that ‘==’ is an operator, while equal() is a method in java. The == operator is used for reference or addresses comparison, meaning it checks if both the objects or variables are pointing to the same memory location.

Whereas, equals() method compares the content of two objects or variables. It checks if the value of two objects is the same or not.

```
import java.util.*;
public class B
{
    public static void main(String args[])
    {
           String str1=new String("Alex");
           String str2=new String("Alex");
           System.out.println("Output given by == operator: "+(str1==str2));
           System.out.println("Output by equals() method: "+(str1.equals(str2)));
    }
}
```

```
Output:
Output given by == operator: false
Output by equals() method: true
```

43. What are JAR files?

Java Archive(JAR) is a file format that aggregates many files into a single file similar in a format similar to a ZIP file. In other words, the JAR is a zipped file comprised of java lass files, metadata, and resources like text, images, audio files, directories, .class files, and more.
Following is the syntax to create a JAR file:

```
Jar cf file_name input files
```

44. What is a WAR file?

The Eb Archive file contains the files of a web application and reduces the time needed to transfer the files. War files can store java classes, XML, JSP, image, CSS, js, HTML, and JavaServer pages, which can be used to distribute the collection of Java classes, XML files, Java Servlets, Web pages, etc.

Following is the syntax to create a WAR file:

```
Jar -cvf file_name.war*
```

These are the Core Java basic interview questions for freshers, let us check out some of the top Java interview questions that come under the intermediate category.

Intermediate Java Interview Questions and Answers

45. Differentiate between this() and super() in Java.

this()	super()
Represents the present instance of a class	Represents the current instance of the parent class
Calls the default constructor	Calls the base class constructor
Used to point to the current class instance	Used to point to the instance of the superclass

46. Name the methods of an object class.

- **clone()**: This method helps create and return a copy of an object.
- **equals()**: This method helps compare.
- **finalize()**: It is called by the garbage collector on the object.
- **getClass()**: It allows us to return the runtime class of the object.
- **hashCode()**: This method helps return a hash code value for the object.
- **toString()**: It lets us return a string representation of the object.
- **notify(), notifyAll(), and wait()**: These help in synchronizing the activities of the independently running threads in a program.

Read this tutorial to learn more about Java Methods!

47. Define content negotiation.

If we have visited a website to search for information, we will get the information in different languages and in different formats. When a client makes an HTTP request to a server, the client can also specify the media types here. The client can specify what it can accept back from the host, and on the basis of availability the host will return to the client. This is known as content negotiation because the client and the server negotiate on the language and format of the content to be shared.

48. Can we import the same package/class twice? Will the JVM load the package twice at runtime?

A package or class can be inherited multiple times in a program code. JVM and compiler will not create any issue. Moreover, JVM automatically loads the class internally once, regardless of times it is called in the program.

Watch this Java Tutorial for Beginners video:

Java Tutorial for Beginners | Java Programming | Intellipaat

49. Define an abstract class.

A class that contains the abstract keyword in its declaration is known as an abstract class. It can have abstract and non-abstract methods (method with a body).

1. This class can have public, private, protected, or constants and default variables.
2. It needs to be extended and its method needs to be implemented. It cannot be instantiated.
3. If a class has at least one abstract method, then the class must be declared abstract.

50. Describe annotations.

- Java annotation is a tag that symbolizes the metadata associated with class, interface, methods, fields, etc.
- Annotations do not directly influence operations.
- The additional information carried by annotations is utilized by Java compiler and JVM.

51. Java doesn't use pointers. Why?

Pointers are susceptible and slightly carelessness in their use which may result in memory problems, and hence Java basically manages their use.

This blog will help you get a better understanding of Mobile Technology for Android and iOS using Java!

52. Distinguish between static loading and dynamic class loading.

- **Static loading**: In static loading, classes are loaded statically with the operator 'new.'
- **Dynamic class loading**: It is a technique for programmatically invoking the functions of a class loader at runtime. The syntax for this is as follows:

```
Class.forName (Test className);
```

53. Struts 1 classes are not thread-safe, whereas Struts 2 classes are thread-safe. Why?

Struts 1 actions are singleton. So, all threads operate on the single action object and hence make it thread-unsafe.

Struts 2 actions are not singleton, and a new action object copy is created each time a new action request is made and hence it is thread-safe.

54. Define JAXP and JAXB.

JAXP: It stands for Java API for XML Processing. This provides a common interface for creating and using DOM, SAX, and XSLT APIs in Java regardless of which vendor's implementation is actually being used.

JAXB: It stands for Java API for XML Binding. This standard defines a system for a script out of Java objects as XML and for creating Java objects from XML structures.

Learn more about Java Multithreading from this tutorial!

55. Define an enumeration?

The term “enum” is commonly used to refer to an enumeration. An enumeration represents an interface that encompasses methods facilitating access to the underlying data structure from which the enumeration is derived. It enables a systematic, sequential retrieval of all the elements stored within the collection.

56. How can we find the actual size of an object on the heap?

In the Java programming language, you cannot accurately determine the exact size of an object located in the heap.

57. Which API is provided by Java for operations on a set of objects?

Java offers a comprehensive Collection API that presents a multitude of beneficial methods applicable to a group of objects. Noteworthy classes within the Collection API encompass ArrayList, HashMap, TreeSet, and TreeMap. These classes serve as valuable tools for handling, manipulating, and organizing data structures in Java programs.

58. What’s the base class of all exception classes?

Java.Lang.throwable: It is the superclass of all exception classes, and all exception classes are derived from this base class.

Learn all about arrays from this comprehensive Java tutorial!

59. Why do we use a vector class?

A vector class provides the ability to execute a growable array of objects. A vector proves to be very useful if you don’t know the size of the array in advance or if we need one that can change the size over the lifetime of a program.

60. What is the difference between transient and volatile variables in Java?

Transient: In Java, it is used to specify whether a variable is not being serialized. Serialization is a process of saving an object’s state in Java. When we want to persist the object’s state by default, all instance variables in the object are stored. In some cases, we want to avoid persisting a few variables because we don’t have the necessity to transfer across the network. So, we declare those variables as transient.

If the variable is confirmed as transient, then it will not be persisted. The transient keyword is used with the instance variable that will not participate in the serialization process. We cannot use static with a transient variable as they are part of the instance variable.

Volatile: The volatile keyword is used with only one variable in Java, and it guarantees that the value of the volatile variable will always be read from the main memory and not from the thread’s local cache; it can be static.

61. Why Map interface does not extend the Collection interface in the Java Collections Framework?

The Map interface is not compatible with the Collection interface, because Map requires a key as well as a value, for example, if we want to add a key–value pair, we will use put(Object key, Object value).

There are two parameters required to add an element to HashMap object. In Collection interface, add(Object o) has only one parameter.

The other reasons are: Map supports valueSet, keySet, and other suitable methods that have just different views from the Collection interface.

62. Mention the uses of the synchronized block.

We use the synchronized block because:

- It helps lock an object for every shared resource.
- The scope of the synchronized block is smaller than the method.

63. What are the functions of hashCode() method?

The hashCode() method returns a hash code value (an integer number) and also the same integer number if the two keys (by calling the equals() method) are the same.
But sometimes, the two hash code numbers can have different or the same keys.

64. What is the default size of the load factor in the hashing-based collection?

Default size = 0.75
Default capacity = initial capacity * load factor

65. What are the differences between the JSP custom tags and Java beans?

- Custom tags can manipulate JSP content, but beans cannot.
- Composite operations can be reduced to a considerably simpler form with custom tags than with beans.
- Custom tags require reasonably a bit more work to set up than beans do.

- Custom tags are available only in JSP 1.1 and so on, but beans can be used in all JSP 1.x versions.

66. What are wrapper classes in Java?

Wrapper Classes in Java offers a mechanism to convert the primitive data types into reference or object types and vice-versa. Each primitive data type in Java has a dedicated class known as the wrapper class, which wraps the primitive types into an object of that particular class. Converting primitive data types into objects is known as autoboxing and converting from an object to primitive data types called unboxing.

One of the major applications of wrapper classes is changing the value inside the methods. Java does not support the call by reference, which means changes done inside a method will not change the original value. So, after converting the types into objects, the original values will also change. Also, the wrapper classes help to perform serialization by converting objects into streams.

67. Difference between Stack and Heap Memory in Java?

The key differences between Heap and Stack memory in Java are:

Features	Stack Memory	Heap Memory
Memory Size	Smaller in size.	Larger in size.
LifeTime	Exists until the end of the thread.	Lives from the start till the end of application execution.
Access	Objects stored in Stack can't be accessed by other threads.	Globally accessible.
Memory Management	Follows LIFO(Last In First Out) order.	Dynamic memory allocation.
Scope or Visibility	The variables in Stack memory are only visible to the owner thread.	Heap memory is visible to all the threads.
Order of Allocation	Continuous memory allocation.	Random order memory allocation.
Flexibility	Less flexible, one cannot alter the allocation memory.	Users can alter the allocated memory.
Efficiency	Fast access, allocation, and deallocation.	Slow access, allocation, and deallocation.

68. What is Polymorphism?

Polymorphism is the ability of an object to get processed in more than one way. It is mainly used when a base/parent class is used to refer to a child/derived class reference. There are two types of polymorphism in Java:

- **Compile-time Polymorphism:** Compile-time polymorphism is method overloading where two or more methods have the same name with arguments/parameters and return type.
- **Run-time polymorphism:** Runtime polymorphism of the Dynamic Method Dispatch helps the user to resolve the overridden method at runtime rather than compile-time. The overridden method is called the object of a superclass.

69. What do you mean by the interface in Java?

Java interface is a completely abstract class referred to as a collection of abstract methods, static methods, and constants. It acts as a blueprint of a class that groups related methods or functions with empty bodies.

Below is an example of a java interface:

```
import java.io.*;
public interface x
{
   int p=10;
   void pr();
   default void show()
   {
       System.out.println("A default method in interface");
   }
}
public class impx implements x
{
    public void pr()
    {
        System.out.println("Hello World");

    }
    public static void main(String args[])
    {
        x ob=new impx();
        ob.pr();
        ob.show();
    }
}
```

Output:

```
Hello World
A default method in interface
```

70. What is the difference between abstract class and interfaces?

Below are the major differences between the abstract class and an interface:

Abstract Class	Interface
Can provide complete, default, or only the details in method overriding	Doesn’t provide any code, the user can only have the signature of the interface.
Have instance variables.	No instance variables.
An abstract class can be public, private, or protected.	The interface must be public.
An abstract class can have constructors.	No constructors are available in the interface.
A class can inherit only a single abstract class.	A class can implement one or more interfaces.
Abstract classes are fast.	Interfaces are slower as they have to locate the corresponding methods in the class.

71. Explain data encapsulation in Java?

Data encapsulation is a fundamental OOPs concept that wraps the data(variables) and code(methods) as a single unit. These variables and methods can only be accessed through the current class, as they'll be hidden from others. Encapsulation protects the data from unnecessary modification and can be achieved in two ways:

- By declaring the variable of a class as private
- Using public setter and getter methods to view and modify the values

72. What is a servlet in java?

It's a server-side technology used to create web servers and application servers. We can understand Servlets as the java programs that handle the request from the web servers, process the request, and reverts to the web servers after the response is generated.

Several packages like javax.servlet and javax.servlet.HTTP provides the appropriate interfaces and classes that help the user to create their servlet. All the servlets should implement the javax.servlet.Servlet interface, plus other interfaces and classes like the HttpServlet, Java Servlet API, and more.

The execution of a servlet involves six distinct steps:

- The clients initiate the request by sending it to the web server or application server.
- The web server receives the request from the clients.
- The servlet processes the request, carrying out the necessary computations, and generates the corresponding output.
- Subsequently, the servlet sends the processed request back to the web server.
- Finally, the web server dispatches the request to the clients, and the resulting output is displayed on the client's screen or any other designated device.

73. What is a Request Dispatcher?

The request dispatcher serves as an interface utilized to effectively redirect requests to various resources within the application itself. These resources may include Java Server Pages (JSP), images, HTML files, or other servlets. Moreover, it facilitates the integration of responses from one servlet into another, guaranteeing that the client receives output from both servlets. Additionally, it allows for the seamless forwarding of client requests to the next servlet in the specified sequence.

There are two methods defined in a Requestdispatcher interface:

- **void include()**: It includes the content of the resource before sending the response.
- **void forward()**: the method forwards the request from one servlet to another resource like the JSP, image, etc on the server.

74. How do cookies work in Servlets?

Cookies are the text files sent by the server and get stored in the client system to keep the information on the local machine and use it for tracking purposes. Java Servlet supports HTTP cookies through javax.servlet.http.Cookie class that implements cleanable and serializable interfaces.

The server sends a set of cookies to the client computer, for example, name, age, etc. Then, the browser stores the information in the local machine and sends these cookies to the server, if any request is sent to the webserver. This information can then be used to identify the user.

75. What is the difference between ServletContext vs. ServletConfig?

The difference between ServletContext and ServletConfig is as follows:

ServletContext	ServletConfig

Common for all the servlets and generally represents the whole web application.	represents a single servlet.
Similar to global parameters linked to the whole application.	Similar to local parameters associated with a particular servlet.
The ServletContext has an application-wide scope.	ServletConfig has the servlet wide scope.
To get the context object, getServletContext() method is used.	To get the config object, getServletConfig() is used.
Mainly used to get a MIME type of file or application.	Mainly used for specific users or tasks like the shopping card.

76. Explain the thread lifecycle in Java?

There are 5 states in a thread lifecycle and java can be at any one of them. The thread lies in any one of these states:

- **New:** The thread lies in the new state after it has been created. It remains in the new state until you start the execution process.
- **Running:** At runnable state, the thread is ready to run at any point in time or it might have started running already.
- **Running:** The scheduler picks up the thread and changes its state to running, the CPU starts executing the thread in the running state.
- **Waiting:** While being ready to execute, another thread might be running in the system. So, the thread goes to the waiting state.
- **Dead:** Once the execution of the thread is finished, its state is changed to the dead state, which means it's not considered active anymore.

77. How to distinguish processes from threads?

Below are the key differences between the processes and threads:

Process	Thread
The process is the set of instructions or programs in execution.	Thread is a segment or part of a process
Takes more time to execute.	Takes less time to execute.
Context switching is slower.	Context switching is faster.
Processes are generally isolated.	Threads share the memory.
The process is a heavy-weight process.	Lightweight process.
One blocked process will not affect the performance of other processes.	A blocked thread would block other threads of the same process.
The process consumes fewer more resources.	Threads consume fewer resources.

78. Explain the types of Exceptions in Java?

There are two types of Exceptions in Java Programming:

- **Built-in Exception:** These exceptions are the exceptions available in the standard package of java lang. They are used for certain errors with specific Exceptions as mentioned below:
    - AriththmeticException: thrown for the errors related to the arithmetic operations.

  - IOException: thrown for failed input-output operations.
  - FileNotFourndException: raised when the file is not accessible or does not exist.
  - ClassNotFoundException: exception is raised when the compiler is unable to found the class definition.
  - InterruptedEException: raised when a thread is interrupted.
  - NoSuchFieldException: raised when a class or method doesn't have a variable specified.
  - NullPointerException: thrown while referring to the variable or values of a null object.
  - RuntimeException: raised for the errors occurring during the runtime. For example: performing invalid type conversion.
  - IndexOutOfBoundsException: raised when the index of the collection like an array, string, or vector is out of the range or invalid.
- **User-defined Exceptions:** User-defined exceptions are the exceptions thrown by the user with custom messages. These are used for cases where the in-built Exception might not be able to define the error.

79. How to write multiple catch statements under a single try block?

Below is the program for multiple catch statements:

```
import java.util.*;
public class multiple_catch
{
   public static void main(String args[])
   {
       Scanner s=new Scanner(System.in);
       int x,y,z;
       try{
           System.out.println("Enter the elements for division: ");
           x=s.nextInt();
           y=s.nextInt();
           z=x/y;
           System.out.println("Value of z: "+z);
        }
        catch(ArithmeticException e)
        {
            System.out.println("Arithmetic Exception occured");
        }
        catch(IllegalArgumentException e)
        {
            System.out.println("Illegal argument");
        }
        catch(Exception e)
        {
            System.out.println("Any Exception in the third block");
        }
        System.out.println("End of the program");
   }
}
```

The output of the following program in different cases will be:

```
Case 1: Dividing an integer by zero
Enter the elements for division:
30
0
Arithmetic Exception occurred
End of the program
```

```
Case 2: Execution successful
Enter the elements for division:
15
3
Value of z: 5
End of the program
```

```
Case 3: Value entered is invalid
Enter the elements for division:
23
t
Any Exception in the third block
End of the program
```

80. Difference between the throw and throws keyword?

Following are the key differences between the throw and throws keyword:

Key	throw	throws
Definition	The throw keyword is used to explicitly throw an exception inside a program or a block of code.	Throws keyword is used to declare an exception in the method signature that might occur during the compilation or execution of a program.
Type of exception	Can throw only unchecked exceptions.	Can be used for both checked and unchecked exceptions.
Syntax	throw keyword followed by an instance variable.	throws keyword followed by the exception class names.
The implementation	throw keyword is used to throw only a single exception.	throws can be used to throw multiple exceptions.

81. Write the string reversal program without using the in-built function?

There are two ways we can reverse a string in java without using the in-built function:

- Using the loop and print string backward by one character each time.
- Use the recursive function and print the string backward.

In the following program, you can see both ways to print the spring in reverse order:

```
import java.util.*;
public class string_reversal
{
public void method1(String str)
{
for(int i=str.length()-1;i>=0;i--)
{
System.out.print(str.charAt(i));
}
System.out.println();
}
public void reverse(String str)
{
if(str.isEmpty())
{
return;
}
else
reverse(str.substring(1));
System.out.print(str.charAt(0));
}
public static void main(String args[])
{
String str="Intellipaat";
System.out.println("The oroginal string is "+str);
System.out.println("String reversal using the loop: ");
string_reversal obj=new string_reversal();
obj.method1(str);
System.out.println("String reversal using recursion: ");
obj.reverse(str);
}
}
```

The output of the above program will be:

```
The oroginal string is Intellipaat
String reversal using the loop:
taapilletnI
String reversal using recursion:
taapilletnI
```

82. Write a program to print Fibonacci Series using the recursion concept?

```
import java.util.*;
public class fib
{
    public int fibonacci(int x)
    {
        if(x==0)
        {
            return 0;
        }
        else if(x==1||x==2)
        {
            return 1;
        }
        else
        {
            return (fibonacci(x-2)+fibonacci(x-1));
        }
    }
    public static void main(String args[])
    {
        int t=10;
        fib ob=new fib();
        for(int i=0;i<t;i++)
        {
            System.out.print(ob.fibonacci(i)+" ");
        }
    }
}
```

The output of the above program is:

```
0 1 1 2 3 5 8 13 21 34
```

That is all in the section of intermediate core Java interview questions. Let's move on to the next section of advanced Java interview questions for experienced professionals.

Advanced Java Interview Questions and Answers for Experienced Professionals

83. Explain the expression language in JSP.

The expression language is used in JSP to simplify the accessibility of objects. It provides many objects that can be used directly like param, requestScope, sessionScope, applicationScope, request, session, etc.

84. What are implicit objects?

Implicit objects, also called pre-defined variables, are created by the JSP engine inside the service method so that it can be accessed directly without being declared explicitly.

85. Define a cookie. What are the differences between a session and a cookie?

A cookie is a small piece of information, which is sent to the browser by a web server. The browser stores the cookies for every web server in a local file. In a future request, the browser sends all stored cookies for that detailed web server.

The difference between cookies and sessions are:

- A session works regardless of the settings on the client browser. The client may have selected to disable cookies.
- Sessions and cookies also differ in storing the amount of information. The HTTP session is able to store any Java object, while a cookie can only store String objects.

86. What is HTTP Tunneling?

HTTP tunneling refers to encapsulating the private network data/information and transmitting it through a public network. Instead of sending data as a packet, tunneling encrypts the data and encapsulates it in a connection. This process allows the outside clients to collect all the data sent by the client-side Object Request Broker (ORB) that needs to be sent to the server-side ORB. HTTP tunneling masks other protocol requests as HTTP requests.

87. What is the function of the IOC container in spring?

The IOC container is responsible for:

- Creating an instance
- Configuring the instance
- Assembling the dependencies

88. What is lazy loading in hibernate?

Lazy loading is a kind of setting that decides whether to load the child entities along with the parent entities or not. When enabling this feature, the associated entities will be loaded only when they are requested directly. The default value of this setting is 'true' which stops the child entities from loading.

89. How can we fetch records using Spring JdbcTemplate?

We can fetch records from the database by the query method of JdbcTemplate. There are two interfaces to do this:

1. ResultSetExtractor
2. RowMapper

90. Which is the front controller class of Spring MVC?

The Dispatcher Servlet class works as the front controller in Spring MVC.

91. What are the states of an object in hibernate?

The states of an object in hibernate are:

1. **Transient:** The objects that are just created having no primary key are in a transient state. Here the objects are associated with any session.
2. **Persistent**: When the session of an object is just opened and its instance is just saved or retrieved, it is said to be in a persistent state.

3. **Detached:** When the session of an object is closed, it is said to be in a detached state.

92. How to make an immutable class in hibernate?

If we mark a class as mutable=”false”, the class will be treated as an immutable class. The default value of mutable is “true”.

93. What is hash-collision in a HashTable? How is it handled in Java?

In HashTable, if two different keys have the same hash value, then it leads to hash-collision. A bucket of type linked list is used to hold the different keys of the same hash value.

94. Write a syntax to convert a given collection to a SynchronizedCollection.

The following will convert a given collection to a synchronized collection:

```
Collections.synchronizedCollection(Collection collectionObj)
```

95. Write a code to make the Collections read-only.

We can make the Collections read-only by using the following lines code:

```
General : Collections.unmodifiableCollection(Collection c)Collections.unmodifiableMap(Map m)
Collections.unmodifiableList(List l)
Collections.unmodifiableSet(Set s)
```

96. What is meant by binding in RMI?

Binding is the process of associating or registering a name for a remote object, which can be used as a further, in order to look up that remote object. A remote object can be associated with a name using the bind/rebind methods of the Naming class.

97. What are latest features introduced in Java 8?

The below features are introduced in Java 8:

- Lambda Expressions
- Interface Default and Static Methods
- Method Reference
- Parameters Name
- Optional Streams
- Concurrency

98. Name a few Java 8 annotations.

@Functional Interface annotation: It was introduced in Java SE 8, which indicates that the type declaration is intended to be a functional interface as defined by the Java language specification.

@Repeatable annotation: Introduced in Java SE 8, @Repeatable annotation indicates that the marked annotation can be applied many times to the same declaration or type use.

99. Distinguish between a predicate and a function.

A predicate takes one argument and returns a Boolean value.

A function takes one argument and returns an object. Both are useful for evaluating lambda expressions.

100. Write a code to sort a list of strings using Java 8 lambda expression.

```
private void sortUsingJava8(List names){
Collections.sort(names, (p1, p2) -> p1.compareTo(p2));
}
```

101. What is Nashorn in Java 8?

With Java 8, Nashorn, a much-improved JavaScript engine, is introduced and it replaced the existing Rhino. It provides 2–10 times better performance, as it directly compiles the code in memory and passes the bytecode to JVM. Although, Nashorn uses the invoke dynamic feature, introduced in Java 7, to improve performance.

102. Define a StringJoiner and write a sample code.

StringJoiner is a class in Java that is used to join multiple strings together, optionally adding a delimiter between them. Here's an example code that demonstrates the usage of StringJoiner:

```
Sample CodeStringJoiner strJoiner = new StringJoiner(".");
strJoiner.add("AAA").add("BBB");
System.out.println(strJoiner);
OutPut:
AAA.BBB
```

103. Can you execute the JavaScript code from Java 8 code base?

Yes! We can execute the JavaScript code from Java 8 code base by using ScriptEngineManager. We call JavaScript code and interpret it in Java.

104. What is the use of batch processing in JDBC?

Batch processing is a technique that is used to group a set of related SQL queries and execute them in order. This is done to avoid the execution of a single query at a time.

Making use of batch processing ensures that the queries execute faster, thereby maintaining the high efficiency and overall speed of execution.

105. What are the types of statements supported by JDBC?

There are three types of statements supported by **JDBC** as shown below:

- Statement: Used to execute static queries and for general access to the database
- CallableStatement: Provides access to stored procedures and runtime parameters
- PreparedStatement: Provides input parameters to queries while they are executing

106. What is a collection in Java?

A collection is a framework that is used to store and manipulate a container of objects in Java.

Java Collections can be used for performing a variety of operations such as:

- Search
- Sort
- Manipulate
- Delete

Below is the chart that represents the hierarchy of collections in Java:

107. What is the meaning of constructor overloading in Java?

Constructor overloading is a technique that is popular among programmers who have a requirement of adding multiple constructors to a single class with a different parameter list.

The following denotes an example of constructor overloading:

```
class Hello
{
int i;
public Hello(int x)
{
i=k;
}
public Hello(int x, int y)
{
/* Set of Instruction */
}
}
```

To clear your doubts you can visit Intellipaat's Java Community!

108. What is the JDBC Driver?

Java Database Connectivity(JDBC) is an API that allows the client to access different data sources from spreadsheets to flat files and relational databases. It's a middle layer interface that establishes a connection between java applications and databases. In addition to the connection of data sources, JDBC is used for sending queries and updates to the database, retrieving and processing data received from the database.

There are four types of JDBC drivers:

- **JDBC-ODBC bridge Drivers:** Uses Open Database Connectivity(ODBC) to connect with the database. It's also known as the Universal Driver as it can be used to connect with any database.
- **Native-API Drivers:** It's a partial java Driver that uses client-side libraries of the database to convert JDBC calls into native ones of the database programming interface. The Network-API needs to be installed on individual client machines separately.
- **Network protocol Drivers:** The driver uses the middleware or application server to convert the JDBC calls into the vendor-specific protocol. It doesn't require individual client-side installation as all the connectivity drivers are present in one driver.
- **Thin Drivers:** These are portable drivers that won't require any installation from any client or server-side installation. They directly interact with the database and don't require any database library to do so.

109. What is the difference between execute, executeQuery, and executeUpdate?

Below are the key differences between execute, executeQuery, and executeUpdate:

execute	executeQuery	executeUpdate
Used to execute any SQL query and get the result in boolean format (True or False).	executeQuery is used to execute the SQL queries and retrieve some data from the database.	executeUpdate is used to execute DML (Data Manipulation Language) statements like updating or modifying the database.
Returns a boolean value, where TRUE indicates the query returned is an object of the ResultSet, and FALSE indicates that nothing or an integer value has returned.	Returns the ResultSet object that contains the result generated and returned by the query.	Returns an integer value representing the number of rows updated or altered by the query.
Used to execute both select and non-select queries.	Used to execute select queries.	Used to execute non-select queries.
Ex-any SQL statement.	Ex- SELECT	Ex- INSERT, CREATE, ALTER, DELETEDDL, UPDATE, etc