I'm not a bot

Find the comprehensive guide to Java interview questions and answers for both freshers and experienced professionals. Here, we have covered the top interview questions on Java that are likely to be asked when you appear for your job interview. Java has been ruling the world of application development for more than a couple of decades now. Many new programming languages have hit the market over the years, but Java has still been a dominating language because of its unique features and robust functionalities. If you are someone who is looking to kickstart a career as a Java Programmer or Java Developer, or looking to switch your current job as an experienced professional, then knowing what are the top Java interview questions is crucial. It is because there will be several concepts that you need to know and answer well to crack the interview and get that dream job. To make things easier for you, we have curated everything here in detail. So, this article is for you if you are searching for: Java interview questions and answers for freshers Java interview questions and answers for experienced professionals Java interview questions PDF (Free Download) Advanced Java interview questions for 3-10 years experience. In addition to this, we have also included some expert tips to prepare for a Java developer interview, the most important programs that can be asked in the interview, and much more. Lets get started! In this section, we have covered some basic Java interview questions and answers for freshers. If you have done a Java course and are looking to begin your career in this field, then ensure to go through these questions below. It is one of the most popular programming languages today, with its role in the development of web applications, mobile apps, software, gaming systems, as well as server-side technologies. Java programming has been here for over 20 years now and has been nothing less than a boon for programmers and developers. Famous as an object-oriented programming language, Java is not only multi-purpose, but also a secure, high-performing, and trusted coding language. This is the reason behind its usage in even enterprise-grade software development. Upskill Yourself With Live Training C and C++ programming are the base of the Java syntax. Among the list of Java interview questions, you can be asked why Java is called platform independent. You must be ready with the correct answer. The meaning of platform-independent is that you can write a Java program on one machine and execute it on other machines or platforms. It has become possible because of the bytecode, and VMs can handle this code accordingly. So, there will be no issues related to the hardware when it comes to running the code. Java was developed in the year 1991. Java was developed by James Gosling. He was a computer scientist based in Canada, and is famous as the founder of Java programming. When he invented Java, he was working at Sun Microsystems, which Oracle later acquired. Write once, run anywhere, or WORA in Java means that it is a coding language where you write a program for some purpose only once and then use it or run it across multiple operating systems. For instance, you can write a program and run it on Windows, macOS, Android, Linux, etc. Javas write once, run anywhere term was first initiated by Sun Microsystems, where the founder of this language used to work. This characteristic makes Java a portable programming language. Also Read: Top 50+ HTML Interview Questions and Answers There is a wide range of use cases of Java programming language. Below are its main applications: Despite the introduction of Kotlin, Java is still used as a reliable programming language for Android app development. This coding language has the software development kits (SDKs) and libraries that are required to develop mobile apps. Another use of Java is in chatbot development. Smart chatbots that use natural language processing (NLP) can be built using this programming language. One of the most important applications of Java is in building games or gaming apps. Some of the world-famous games like Minecraft, Spiral Knights, SimCity, Saints Row 2, Asphalt 3, FIFA 11, Wakfu, Tokyo City Nights and many more are built on Java. The write once, use anywhere characteristic of Java makes it a highly applicable language for cloud applications as well. Plenty of cloud platforms rely on this programming for a decentralized experience. Big data platforms or tools heavily depend on Java, and it is considered a language on which the future of big data relies. This is because of its features that enable faster processing of large sets of data. Enterprise-level apps that are mission-critical are developed using Java programming. Even top brands like Wipro, Google, Infosys, and HCL use it to develop enterprise apps. It is because of its high performance it enables and supports a wide range of server-side technologies. Some of the most popular web apps built on Java include LinkedIn, IRCTC, and AliExpress. In IoT technology, sensors and hardware devices process the data. These things are mostly run using programs written in Java programming language. As one of the most suitable programming languages for artificial intelligence (AI) projects, Java can be used to develop intelligent solutions. For instance, it is great for building search algorithms, neural networks, ML-based services, deep learning applications, etc. JVM in Java, as the name suggests, is a virtual machine that plays a crucial part in the execution of source code. It works as an abstraction layer between the runtime environment and the hardware. JRE in Java is simply an environment that allows developers or programmers to run Java-based apps on operating systems. You can say that it facilitates the interaction between OS and the program. Java JRE provides several resources to programmers, such as libraries, JVM, Java Plug-in, Web Start, etc. It is available to download on Windows, Linux, macOS, and Oracle Solaris. Java Standard Edition, abbreviated as Java SE, is a computing platform on which programmers and developers build and deploy Java-based projects. This platform comes with plenty of Java libraries and APIs, including java.util, java.net, java.math, java.io, and many more. Java operators are simply the symbols used to perform a wide range of different operations. Every operator has its specific operation or functionality. For instance, you can use the + operator for the addition of two values, the operator for subtraction, the * operator for multiplication, and the / operator for division. You can classify the operators in Java into five categories, as mentioned below: These are used for mathematical calculations, or arithmetic operations, to be precise. +AdditionSubtraction*Multiplication/Division%Modulo These operators play a vital role in performing operations related to increasing or decreasing the values. Unary minus (-)To make a value negativeUnary plus (+)Generally not used as values are positive by defaultIncrement (++)To increase the value by 1Decrement ()To decrease the value by 1Inverting (!)To inverse the value These operators, as the name suggests, are used to assign values to variables in a Java program. OperatorPresentationMeaning=X = YX = Y+=X += YX = X + Y-=X -= YX = X = Y*=X *= YX = X * Y%=X %=X = X % Y If you want to see the relations between values, then the relational operators in Java are used. For instance, you can check whether the given values are equal to each other, greater/lesser than each other, greater than equal to, or lesser than equal to each other. If the boolean values dont meet the relational operator criteria, then accordingly, it returns true or false. OperatorMeaning==Is equal to!=Is not equal to>Is greater than=Is greater than or equal to { example.toggleFlag(); System.out.println("Flag set to true."); }); Thread readerThread = new Thread(() -> { while (!example.isFlag()) { // Busy-wait until the flag is set } System.out.println("Flag is true."); }); writerThread.start(); readerThread.start(); }}The 'volatile' keyword in Java is used to declare a variable as volatile. When a variable is declared as volatile, it ensures that any read or write operation on that variable is done directly to and from the main memory, rather than relying on a thread's local cache. This makes the variable's value visible to all threads, preventing potential issues like stale data. Above is an example.In this code, the 'volatile' keyword ensures that the 'flag' variable is consistently read and written to the main memory, preventing issues with thread visibility.How to Respond: Interviewers pose this to assess your knowledge of custom sorting in Java. Explain that the 'Comparator' interface is used to define custom comparison rules for objects. Provide a code example showing how to use a 'Comparator' for custom sorting.Sample Answer: import java.util.ArrayList;import java.util.Collections;import java.util.Comparator;import java.util.List; class Employee { private int id; private String name; public Employee(int id, String name) { this.id = id; this.name = name; } public int getId() { return id; } public String getName() { return name; }} public class ComparatorExample { public static void main(String[] args) { List employees = new ArrayList(); employees.add(new Employee(2, "Alice")); employees.add(new Employee(1, "Bob")); employees.add(new Employee(3, "Charlie")); // Custom sorting based on employee names Collections.sort(employees, Comparator.comparing(Employee::getName)); for (Employee employee : employees) { System.out.println(employee.getName()); } }} In Java, the 'Comparator' interface is used to define custom comparison rules for objects. It allows you to specify how objects should be compared and sorted based on your specific criteria. Above is an example:In this code, a custom 'Comparator' is used to sort a list of employees based on their names in alphabetical order.How to Respond: Interviewers use this to assess your understanding of object comparison in Java. Explain that 'equals' compares the content of objects, while '==' checks if two references point to the same object. Provide a code example illustrating the difference between 'equals' and '=='.Sample Answer:public class ObjectComparisonExample { public static void main(String[] args) { String s1 = new String("Hello"); String s2 = new String("Hello"); System.out.println("Using 'equals': " + s1.equals(s2)); // 'equals' compares content System.out.println("Using '==': " + (s1 == s2)); // '==' checks reference equality }}In Java, the 'equals' method and the '==' operator serve different purposes for comparing objects. 'equals' compares the content of objects to determine if they are equivalent, while '==' checks if two references point to the exact same object in memory.In this code, 's1.equals(s2)' will be true because the content of both strings is the same, while 's1 == s2' will be false because they are different objects in memory.Question 36: Explain the Concept of Polymorphism in Java. How Does It Improve Code Reusability and Maintainability?How to Respond: Interviewers ask this to assess your understanding of polymorphism. Explain that polymorphism allows objects of different classes to be treated as objects of a common superclass. Describe its significance in creating more versatile and maintainable code.Sample Answer: Polymorphism in Java allows objects of various classes to be treated as objects of a common superclass. For example, in a 'Shape' hierarchy, various shapes like 'Circle' and 'Rectangle' can have a common 'area()' method. This improves code reusability because you can create generic code that works with 'Shape' objects, regardless of the specific shape. It also enhances code maintainability since you can easily add new shapes without changing the existing code.Question 37: Write a Java Program to Count the Occurrence of Each Word in a Text.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to count the occurrence of each word in a given text.Sample Answer: import java.util.HashMap;import java.util.Map; public class WordCounter { public static void main(String[] args) { String text = "This is a sample text. This text contains sample words."; String[] words = text.split("\\s+"); // Split text into words Map wordCount = new HashMap(); for (String word : words) { wordCount.put(word, wordCount.getOrDefault(word, 0) + 1); } for (Map.Entry entry : wordCount.entrySet()) { System.out.println("Word: " + entry.getKey() + ", Count: " + entry.getValue()); } }} This code counts the occurrence of each word in the given text and displays the results.How to Respond: Interviewers use this to assess your knowledge of loop control in Java. Explain that the 'break' statement is used to exit a loop prematurely. Provide a code example illustrating the use of 'break' in a loop.Sample Answer: public class BreakStatementExample { public static void main(String[] args) { for (int i = 1; i The 'break' statement in Java is used to prematurely exit a loop when a certain condition is met. In this code, when 'i' equals 5, the 'break' statement is encountered, and the loop is exited prematurely.How to Respond: Interviewers pose this to assess your knowledge of exception handling in Java. Explain that exception propagation refers to the process of passing an exception from one method to another. Provide a code example illustrating exception propagation.Sample Answer:public class ExceptionPropagationExample { public static void main(String[] args) { try { method1(); } catch (Exception e) { System.out.println("Exception caught in main method: " + e.getMessage()); } } public static void method1() { method2(); } public static void method2() { int result = 10 / 0; // This line will throw an ArithmeticException }}Exception propagation in Java is the process of passing an exception from one method to another along the call stack. When an exception is thrown in a method and not caught, it will propagate up the call stack until it is caught or the program terminates. In this code, an exception is thrown in the 'method2,' and it propagates up to the 'main' method, where it is caught and handled.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to find the GCD of two numbers in Java.Sample Answer:public class GCDCalculator { public static void main(String[] args) { int num1 = 48; int num2 = 18; int gcd = findGCD(num1, num2); System.out.println("GCD of " + num1 + " and " + num2 + " is: " + gcd); } public static int findGCD(int a, int b) { while (b != 0) { int temp = b; b = a % b; a = temp; } return a; }}This code calculates the Greatest Common Divisor (GCD) of two numbers using the Euclidean algorithm and displays the result.How to Respond: Interviewers ask this to assess your understanding of the JVM. Explain that the JVM is an integral part of Java's platform independence. It interprets bytecode and executes it on the host platform. Describe how this allows Java applications to run on any platform with a compatible JVM.Sample Answer: The Java Virtual Machine (JVM) is an essential component of Java's platform independence. It interprets compiled Java bytecode and executes it on the host platform. This architecture allows Java applications to run on various platforms as long as they have a compatible JVM installed. The JVM abstracts the underlying hardware and operating system, making Java applications write once and run anywhere (WORA), a fundamental feature of the language.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to check whether a given number is prime in Java.Sample Answer: public class PrimeChecker { public static void main(String[] args) { int number = 17; // Change 'number' to the desired number boolean isPrime = isPrimeNumber(number); if (isPrime) { System.out.println(number + " is a prime number."); } else { System.out.println(number + " is not a prime number."); } } public static boolean isPrimeNumber(int num) { if (num This code checks whether a given number is prime and provides the result.How to Respond: Interviewers ask this to assess your understanding of access modifiers in Java. Explain the differences between 'public,' 'private,' and 'protected,' and discuss their use cases in class members.Sample Answer: In Java, access modifiers ('public,' 'private,' and 'protected') control the visibility and accessibility of class members.'public': Members marked as 'public' are accessible from any class and package. They have the widest visibility and can be accessed freely.'private': Members marked as 'private' are only accessible within the same class. They provide the highest level of encapsulation and are used to hide internal details.'protected': Members marked as 'protected' are accessible within the same class, its subclasses, and the same package. They are used for controlled access and allowing subclasses to access them.Each access modifier serves different purposes, ensuring proper encapsulation and access control in Java classes.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to find the Nth Fibonacci number in Java.Sample Answer: public class FibonacciCalculator { public static void main(String[] args) { int n = 10; // Change 'n' to the desired position in the Fibonacci sequence long nthFibonacci = calculateFibonacci(n); System.out.println("The " + n + "th Fibonacci number is: " + nthFibonacci); } public static long calculateFibonacci(int n) { if (n This code calculates the Nth Fibonacci number using dynamic programming and displays the result.How to Respond: Interviewers use this to assess your knowledge of the 'this' keyword. Explain that 'this' is used to refer to the current object and avoid name conflicts between instance variables and parameters. Provide a code example illustrating the use of 'this' to resolve a name conflict.Sample Answer:public class ThisKeywordExample { private int value; public ThisKeywordExample(int value) { this.value = value; // 'this' refers to the instance variable } public void displayValue(int value) { System.out.println("Instance variable value: " + this.value); System.out.println("Parameter value: " + value); } public static void main(String[] args) { ThisKeywordExample example = new ThisKeywordExample(42); example.displayValue(10); }}The 'this' keyword in Java is used to refer to the current object. It helps avoid name conflicts between instance variables and parameters when they have the same names. In this code, 'this.value' refers to the instance variable 'value,' and 'value' inside the method refers to the parameter. This avoids naming conflicts and ensures the correct variable is used.How to Respond: Interviewers ask this to assess your knowledge of OOP principles in Java. Explain key OOP concepts such as encapsulation, inheritance, polymorphism, and abstraction. Describe how these features enhance software development, making code more organized and maintainable.Sample Answer: Object-Oriented Programming (OOP) in Java is centered around key features like encapsulation, inheritance, polymorphism, and abstraction. These features enhance software development in the following ways:Encapsulation: Encapsulation groups related data and behaviors into classes, protecting data from unauthorized access and making code more organized.Inheritance: Inheritance allows the creation of new classes based on existing ones, promoting code reuse and structuring code hierarchies.Polymorphism: Polymorphism enables the use of objects of different classes through a common interface, making code more flexible and adaptable to changes.Abstraction: Abstraction simplifies complex systems by highlighting essential properties and behaviors, reducing complexity and improving maintainability. Together, these OOP features lead to cleaner, modular, and maintainable code, enhancing software development.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to sort an array of strings in alphabetical order in Java.Sample Answer:import java.util.Arrays;public class StringSorter { public static void main(String[] args) { String[] names = {"Alice", "David", "Bob", "Eve", "Carol"}; Arrays.sort(names); System.out.println("Sorted Names:"); for (String name : names) { System.out.println(name); } }}This code sorts an array of strings in alphabetical order using the Arrays.sort method and displays the sorted names.How to Respond: Interviewers use this to assess your knowledge of inheritance in Java. Explain that the 'super' keyword is used to call the parent class constructor and access parent class members. Provide a code example illustrating the use of 'super' to call the parent class constructor.Sample Answer:class Parent { int value; Parent(int value) { this.value = value; }}class Child extends Parent { Child(int value, int childValue) { super(value); // Call the constructor of the parent class // Additional child-specific initialization }}public class SuperKeywordExample { public static void main(String[] args) { Child child = new Child(10, 20); System.out.println("Child's value: " + child.value); }}The 'super' keyword in Java is used to call the constructor of the parent class. It is especially useful when a subclass wants to initialize the parent class's state. In this code, 'super(value)' is used in the 'Child' class to call the constructor of the 'Parent' class with a value argument, ensuring proper initialization of both parent and child classes.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to find the smallest and largest numbers in an array of integers in Java.Sample Answer:public class ArrayMinMaxFinder { public static void main(String[] args) { int[] numbers = {45, 78, 12, 93, 64, 101, 3}; int min = findMin(numbers); int max = findMax(numbers); System.out.println("Smallest number: " + min); System.out.println("Largest number: " + max); } public static int findMin(int[] numbers) { int min = numbers[0]; for (int number : numbers) { if (number < min) { min = number; } } return min; } public static int findMax(int[] numbers) { int max = numbers[0]; for (int number : numbers) { if (number > max) { max = number; } } return max; }}This code finds the smallest and largest numbers in an array of integers and displays the results.How to Respond: Interviewers ask this to assess your knowledge of exception handling in Java. Explain that the 'try-catch' block is used to handle exceptions gracefully. Provide a code example illustrating how a 'try-catch' block can be used to catch and handle exceptions.Sample Answer:public class TryCatchExample { public static void main(String[] args) { try { int result = divideByZero(); // This method will throw an ArithmeticException System.out.println("Result: " + result); } catch (ArithmeticException e) { System.out.println("An exception occurred: " + e.getMessage()); } } public static int divideByZero() { int numerator = 10; int denominator = 0; return numerator / denominator; }}In this code, the 'try' block contains code that might throw an ArithmeticException, and the 'catch' block handles the exception, preventing the program from crashing and providing an error message.How to Respond: Interviewers ask this to assess your knowledge of memory management in Java. Explain that the Java Garbage Collection is a process where the JVM automatically reclaims memory by identifying and removing objects that are no longer in use. Discuss how it helps in preventing memory leaks and simplifying memory management.Sample Answer: Java Garbage Collection is an automated process in the Java Virtual Machine (JVM) that helps manage memory by identifying and collecting objects that are no longer in use. It helps in memory management by:Reclaiming memory occupied by objects that are no longer reachable, preventing memory leaks.Simplifying memory management tasks by handling memory deallocation automatically.Ensuring efficient memory utilization, especially in long-running applications.Garbage Collection in Java significantly contributes to the stability and reliability of Java applications by preventing common memory-related issues.How to Respond: Interviewers use this to assess your knowledge of the 'final' keyword in Java. Explain that 'final' can be applied to classes, methods, and variables, making them immutable or unextendable. Discuss the implications of using 'final' in each context.Sample Answer: In Java, the 'final' keyword is used to make classes, methods, and variables immutable or unextendable:Final Classes: A 'final' class cannot be subclassed. This provides security and prevents unintended modifications to the class's behavior or structure.Final Methods: A 'final' method cannot be overridden by subclasses. It enforces specific behavior and prevents method modification in subclasses.Final Variables: A 'final' variable is a constant and cannot be reassigned. It ensures the variable's value remains unchanged, often used for constants or configuration settings. The 'final' keyword helps in enhancing code security, maintainability, and predictability.How to Respond: Interviewers ask this to assess your understanding of 'static' and 'instance' members in Java classes. Explain the difference between these types of members, and discuss their use cases in classes.Sample Answer: In Java, 'static' and 'instance' members differ in their association with class and instance objects:Static Members: These belong to the class rather than an instance. They are shared across all instances of the class. Commonly used for constants, utility methods, and variables shared by all objects of the class.Instance Members: These belong to individual instances of the class. Each instance has its set of instance variables and methods. They are used for properties and behaviors specific to each object.For example, a 'static' variable might be used to count the number of instances of a class, while an 'instance' variable would represent unique properties of each object. Understanding the distinction is crucial for proper data and behavior management in Java classes.How to Respond: Interviewers pose this to assess your coding skills. Provide a code example to check if a given string is a palindrome in Java.Sample Answer:public class PalindromeChecker { public static void main(String[] args) { String word = "racecar"; // Change 'word' to the desired string boolean isPalindrome = isPalindrome(word); if (isPalindrome) { System.out.println(word + " is a palindrome."); } else { System.out.println(word + " is not a palindrome."); } } public static boolean isPalindrome(String word) { word = word.toLowerCase(); // Convert to lowercase for case-insensitive comparison int start = 0; int end = word.length() - 1; while (start < end) { if (word.charAt(start) != word.charAt(end)) { return false; } start++; end--; } return true; }}This code checks if a given string is a palindrome in a case-insensitive manner and provides the result.When preparing for a Java interview, it's essential to know what to expect. Java interviews typically cover a range of topics to assess your knowledge and skills. Here's what you can anticipate:Core Java Concepts: Be ready to answer questions on fundamental Java concepts like data types, variables, operators, and control structures.Object-Oriented Programming (OOP): Expect questions about OOP principles, including classes, objects, inheritance, polymorphism, and encapsulation.Data Structures: You may be asked about common data structures like arrays, lists, and maps, and how to use them effectively.Algorithms: Brush up on algorithm basics, as you might need to solve problems or optimize code.Exception Handling: Understand how to handle exceptions gracefully in your Java code.Threading: Be prepared to discuss multi-threading, thread synchronization, and Java's concurrency utilities.Java APIs: Familiarize yourself with key Java APIs for I/O, collections, and more.Frameworks and Libraries: If the job requires it, you might be quizzed on popular Java frameworks like Spring or Hibernate.Coding Challenges: Expect to write code, solve problems, or demonstrate coding skills during technical interviews.Behavioral Questions: Interviews might also include behavioral questions to assess your teamwork, problem-solving, and communication skills.Project Discussion: Be ready to discuss your previous projects, your role in them, and any challenges you faced.Starting a Java interview on the right foot is crucial for a successful outcome. Follow these steps to make a positive first impression:Research the Company: Before the interview, learn about the company, its culture, products, and any recent news or achievements.Review Your Resume: Be ready to discuss your resume, highlighting your relevant skills and experiences.Understand the Job: Familiarize yourself with the job description and requirements, ensuring you can speak to how your skills align with the role.Prepare Questions: Prepare thoughtful questions to ask the interviewer. This shows your interest in the role and the company.Practice Problem-Solving: Practice solving coding challenges and problems. Consider using online coding platforms to sharpen your skills.Know Java Fundamentals: Review core Java concepts, data structures, and algorithms. This is the foundation of most Java interviews.Practice Soft Skills: Work on your communication skills, as clear and effective communication is important in interviews.Dress Appropriately: Choose professional attire that aligns with the company's dress code or industry standards.Arrive Early (Virtual or In-Person): If it's an in-person interview, arrive a bit early. For virtual interviews, ensure your setup is working properly.Stay Calm: Nervousness is natural, but try to stay calm and focused. Take your time to think before answering questions.Follow Up: After the interview, send a thank-you email to the interviewer, expressing your appreciation for the opportunity.Starting a Java interview with confidence and preparedness will help you perform your best and increase your chances of success.Preparing for a Java interview as a fresher can be a daunting but rewarding experience. This step-by-step guide will help you navigate the preparation process effectively, using the questions provided in the article.Research the CompanyStart by researching the company where you have an interview scheduled. Understand its culture, products, and values.Review Your ResumeCarefully review your resume, emphasizing any relevant coursework, projects, or internships you've completed.Understand the Job DescriptionThoroughly study the job description to identify the specific skills and knowledge required for the role.Prepare QuestionsCreate a list of questions you can ask the interviewer about the company, role, or industry. Thoughtful questions demonstrate your interest. You can refer to the above most common questionsReview Core Java ConceptsRevisit the core Java concepts. Use the article's "What to Expect in a Java Interview?" section to guide your study.Brush Up on Object-Oriented Programming (OOP)Study OOP principles, including classes, objects, inheritance, polymorphism, and encapsulation, as outlined in the article.Explore Data Structures and AlgorithmsFamiliarize yourself with common data structures and basic algorithms, as mentioned in the article.Practice Exception HandlingLearn how to handle exceptions gracefully. Understanding exception handling is vital for a Java developer.Dive into ThreadingGain a basic understanding of multi-threading, thread synchronization, and Java's concurrency utilities.Get to Know Java APIsReview key Java APIs for I/O, collections, and more, as outlined in the article.Code, Code, CodePractice coding challenges and problem-solving. Use online coding platforms to improve your coding skills.Develop Soft SkillsEnhance your soft skills, including communication and teamwork. Effective communication is valuable in interviews.Dress and PrepareChoose professional attire in line with the company's dress code. Ensure your interview setup is ready for virtual interviews.Stay CalmNervousness is natural. Prior to the interview, take a deep breath, and remember to stay calm and composed.The InterviewDuring the interview, discuss your resume, skills, and experiences. Be ready to answer questions and demonstrate your Java knowledge.Ask QuestionsDon't forget to ask thoughtful questions about the role and company. This shows your interest and engagement.By following this step-by-step guide, you'll be well-prepared to succeed in your Java interview as a fresher.1. Portfolio Presentation:Create an online portfolio showcasing your Java projects and code samples. A well-organized portfolio demonstrates your skills effectively.2. Online Java Communities:Join online Java communities, forums, or Stack Overflow. Engaging in discussions and helping others can improve your problem-solving abilities.3. Open-Source Contributions:Contribute to open-source Java projects. It's a practical way to gain real-world experience and collaborate with experienced developers.4. Side Projects:Develop side projects that solve practical problems. Having a diverse range of projects in your portfolio can make you stand out.5. Algorithm Complexity:Understand Big O notation and algorithm complexity. Interviewers might assess your ability to choose efficient algorithms.6. Software Design Patterns:Study common software design patterns like Singleton, Factory, and Observer. Demonstrating knowledge of these patterns can impress interviewers.7. Code Documentation:Practice documenting your code effectively. Well-documented code illustrates your professionalism and makes collaboration easier.8. Presentation Skills:Improve your ability to explain technical concepts clearly. This skill is invaluable when discussing your projects and solutions.9. Mobile Development:Explore mobile app development using Java for Android. Diversifying your skills can open up additional job opportunities.10. Mock Interviews: - Participate in mock interviews to simulate real interview scenarios. Seek feedback from mentors or peers to refine your interview skills.Is Java still a good career?Java remains a solid career choice. Its versatility, demand in various industries, and vast community support make it a reliable option for developers.Is it hard to get a job in Java?Getting a job in Java can vary based on your skills and location. Building a strong portfolio and mastering core concepts can make it easier.What is Java Freshers Salary?Java freshers' salaries vary by location and company. On average, it can range from $50,000 to $80,000 annually, with room for growth as experience accrues.Is Java growing or dying?Java is still evolving and remains a vital part of many systems. It's not dying but adapting to new technologies, including Android app development.Is Java highly paid?Java developers often receive competitive salaries. With experience and specialization, you can earn significantly higher incomes, particularly in fields like financial technology and enterprise software.Good Luck! with your Interview Do you have what it takes to ace a Java Interview? We are here to help you in consolidating your knowledge and concepts in Java. Before we begin, let's understand what Java is all about.What is Java?Java is a high-level programming language that was developed by James Gosling in the year 1982. It is based on the principles of object-oriented programming and can be used to develop large-scale applications.The following article will cover all the popular Core Java interview questions, String Handling interview questions, Java 8 interview questions, Java multithreading interview questions, Java OOPs interview questions, Java exception handling interview questions, collections interview questions, and some frequently asked Java coding interview questions.Go through all the important questions to enhance your chances of performing well in the Java Interviews. The questions will revolve around the basic, core & advanced fundamentals of Java.So, lets dive deep into the plethora of useful Java Technical Interview Questions and Answers categorised into the following sections:Join our community and share your recent Java interview experiences. Java language was developed so that it does not depend on any hardware or software because the compiler compiles the code and then converts it to platform-independent byte code which can be run on multiple systems.The only condition to run that byte code is for the machine to have a runtime environment (JRE) installed in it.Learn More Create a free personalised study plan Create a FREE custom study plan Get into your dream companies with expert guidance Get into your dream companies with expert.. Real-Life Problems Prep for Target Roles Custom Plan Duration Java supports primitive data types - byte, boolean, char, short, int, float, long, and double and hence it is not a pure object oriented language. Stack memory is the portion of memory that was assigned to every individual program. And it was fixed. On the other hand, Heap memory is the portion that was not allocated to the java program but it will be available for use by the java program when it is required, mostly during the runtime of the program.Java Utilizes this memory as -When we write a java program then all the variables, methods, etc are stored in the stack memory.And when we create any object in the java program then that object was created in the heap memory. And it was referenced from the stack memory.Example- Consider the below java program:class Main { public void printArray(int[] array){ for(int i : array) System.out.println(i); } public static void main(String args[]) { int[] array = new int[10]; printArray(array); }}For this java program. The stack and heap memory occupied by java is -Main and PrintArray is the method that will be available in the stack area and as well as the variables declared that will also be in the stack area.And the Object (Integer Array of size 10) we have created, will be available in the Heap area because that space will be allocated to the program during runtime. You can download a PDF version of Java Interview Questions. Download PDF Download PDF It is not wrong if we claim that Java is the complete object-oriented programming language because everything in Java is under the classes and we can access them by creating the objects.But we can even say that Java is not a completely object-oriented programming language because it has the support of primitive data types like int, float, char, boolean, double, etc.Now for the question: Is Java a completely object-oriented programming language? We can say that - Java is not a pure object-oriented programming language, because it has direct access to primitive data types. And these primitive data types don't directly belong to the Integer classes. C++ is only a compiled language, whereas Java is compiled as well as an interpreted language.Java programs are machine-independent whereas a c++ program can run only in the machine in which it is compiled.C++ allows users to use pointers in the program. Whereas java doesnt allow it. Java internally uses pointers.C++ supports the concept of Multiple inheritances whereas Java doesn't support this. And it is due to avoiding the complexity of name ambiguity that causes the diamond problem. Pointers are quite complicated and unsafe to use by beginner programmers. Java focuses on code simplicity, and the usage of pointers can make it challenging. Pointer utilization can also cause potential errors. Moreover, security is also compromised if pointers are used because the users can directly access memory with the help of pointers.Thus, a certain level of abstraction is furnished by not including pointers in Java. Moreover, the usage of pointers can make the procedure of garbage collection quite slow and erroneous. Java makes use of references as these cannot be manipulated, unlike pointers. Instance variables are those variables that are accessible by all the methods in the class. They are declared outside the methods and inside the class. These variables describe the properties of an object and remain bound to it at any cost.All the objects of the class will have their copy of the variables for utilization. If any modification is done on these variables, then only that instance will be impacted by it, and all other class instances continue to remain unaffected.Example:class Athlete {public String athleteName;public double athleteSpeed;public int athleteAge;}Local variables are those variables present within a block, function, or constructor and can be accessed only inside them. The utilization of the variable is restricted to the block scope. Whenever a local variable is declared inside a method, the other class methods dont have any knowledge about the local variable.Example:public void athlete() {String athleteName;double athleteSpeed;int athleteAge;} Advance your career with Mock Assessments Refine your coding skills with Mock Assessments Real-world coding challenges for top company interviews Real-world coding challenges for top companies Real-Life Problems Detailed reports There are no default values assigned to the variables in java. We need to initialize the value before using it. Otherwise, it will throw a compilation error of (Variable might not be initialized).But for instance, if we create the object, then the default value will be initialized by the default constructor depending on the data type.If it is a reference, then it will be assigned to null.If it is numeric, then it will assign to 0.If it is a boolean, then it will be assigned to false. Etc. Data Encapsulation is an Object-Oriented Programming concept of hiding the data attributes and their behaviours in a single unit.It helps developers to follow modularity while developing software by ensuring that each object is independent of other objects by having its own methods, attributes, and functionalities.It is used for the security of the private properties of an object and hence serves the purpose of data hiding. JIT stands for Just-In-Time and it is used for improving the performance during run time. It does the task of compiling parts of byte code having similar functionality at the same time thereby reducing the amount of compilation time for the code to run.The compiler is nothing but a translator of source code to machine-executable code. But what is special about the JIT compiler? Let us see how it works:First, the Java source code (.java) conversion to byte code (.class) occurs with the help of the javac compiler.Then, the .class files are loaded at run time by JVM and with the help of an interpreter, these are converted to machine understandable code.JIT compiler is a part of JVM. When the JIT compiler is enabled, the JVM analyzes the method calls in the .class files and compiles them to get more efficient and native code. It also ensures that the prioritized method calls are optimized.Once the above step is done, the JVM executes the optimized code directly instead of interpreting the code again. This increases the performance and speed of the execution. We are already aware of the (==) equals operator. That we have used this to compare the equality of the values. But when we talk about the terms of object-oriented programming, we deal with the values in the form of objects. And this object may contain multiple types of data. So using the (==) operator does not work in this case. So we need to go with the .equals() method.Both [(==) and .equals()] primary functionalities are to compare the values, but the secondary functionality is different.So in order to understand this better, lets consider this with the example -String str1 = "InterviewBit";String str2 = "InterviewBit";System.out.println(str1 == str2);This code will print true. We know that both strings are equals so it will print true. But here (==) Operators dont compare each character in this case. It compares the memory location. And because the string uses the constant pool for storing the values in the memory, both str1 and str2 are stored at the same memory location. See the detailed Explanation in Question no 73: Link.Now, if we modify the program a little bit with -String str1 = new String("InterviewBit");String str2 = "InterviewBit";System.out.println(str1 == str2);Then in this case, it will print false. Because here no longer the constant pool concepts are used. Here, new memory is allocated. So here the memory address is different, therefore (==) Operator returns false. But the twist is that the values are the same in both strings. So how to compare the values? Here the .equals() method is used..equals() method compares the values and returns the result accordingly. If we modify the above code with - System.out.println(str1.equals(str2));Then it returns true.equals()==This is a method defined in the Object class.It is a binary operator in Java.The .equals() Method is present in the Object class, so we can override our custom .equals() method in the custom class, for objects comparison.It cannot be modified. They always compare the HashCode.This method is used for checking the equality of contents between two objects as per the specified business logic.This operator is used for comparing addresses (or references), i.e checks if both the objects are pointing to the same memory location.Note:In the cases where the equals method is not overridden in a class, then the class uses the default implementation of the equals method that is closest to the parent class.Object class is considered as the parent class of all the java classes. The implementation of the equals method in the Object class uses the == operator to compare two objects. This default implementation can be overridden as per the business logic. Infinite loops are those loops that run infinitely without any breaking conditions. Some examples of consciously declaring infinite loop is:for (;;){ }while(true){ }do{ }while(true); Constructor overloading is the process of creating multiple constructors in the class consisting of the same name with a difference in the constructor parameters. Depending upon the number of parameters and their corresponding types, distinguishing of the different types of constructors is done by the compiler.class Hospital {int variable1, variable2;double variable3;public Hospital(int doctors, int nurses) { variable1 = doctors; variable2 = nurses;}public Hospital(int doctors) { variable1 = doctors;}public Hospital(double salaries) { variable3 = salaries}}Three constructors are defined here but they differ on the basis of parameter type and their numbers. Copy Constructor is the constructor used when we want to initialize the value to the new object from the old object of the same class.class InterviewBit{ String department; String service; InterviewBit(InterviewBit ib){ this.departments = ib.departments; this.services = ib.services; }}Here we are initializing the new object value from the old object value in the constructor. Although, this can also be achieved with the help of object cloning. Yes, It is possible to overload the main method. We can create as many overloaded main methods we want. However, JVM has a predefined calling method that JVM will only call the main method with the definition of -public static void main(string[] args)Consider the below code snippets:class Main { public static void main(String args[]) { System.out.println(" Main Method"); } public static void main(int[] args){ System.out.println("Overloaded Integer array Main Method"); } public static void main(char[] args){ System.out.println("Overloaded Character array Main Method"); } public static void main(double[] args){ System.out.println("Overloaded Double array Main Method"); } public static void main(float args){ System.out.println("Overloaded float Main Method"); }} In Java, method overloading is made possible by introducing different methods in the same class consisting of the same name. Still, all the functions differ in the number or type of parameters. It takes place inside a class and enhances program readability.The only difference in the return type of the method does not promote method overloading. The following example will furnish you with a clear picture of it.class OverloadingHelp { public int findarea (int l, int b) { int var1; var1 = l * b; return var1; } public int findarea (int l, int b, int h) { int var2; var2 = l * b * h; return var2; }}Both the functions have the same name but differ in the number of arguments. The first method calculates the area of the rectangle, whereas the second method calculates the area of a cuboid.Method overriding is the concept in which two methods having the same method signature are present in two different classes in which an inheritance relationship is present. A particular method implementation (already present in the base class) is possible for the derived class by using method overriding.Lets give a look at this example:class HumanBeing { public int walk (int distance, int time) { int speed = distance / time; return speed; }}class Athlete extends HumanBeing { public int walk(int distance, int time) { int speed = distance / time; speed = speed * 2; return speed; }}Both class methods have the name walk and the same parameters, distance, and time. If the derived class method is called, then the base class method walk gets overridden by that of the derived class. Yes, multiple catch blocks can exist but specific approaches should come prior to the general approach because only the first catch block satisfying the catch condition is executed. The given code illustrates the same:public class MultipleCatch {public static void main(String args[]) { try { int n = 1000, x = 0; int arr[] = new int[n]; for (int i = 0; i > Bitwise Right Shift Operator- This operator shifts each bit to its right position. And this maintains the signed bit.>>> Bitwise Right Shift Operator with trailing zero- This operator also shifts each bit to its right. But this doesnt maintain the signed bit. This operator makes the Most significant bit to 0.Example- Num1 = 8, Num2 = -8.So the binary form of these numbers are -Num1 = 00000000 00000000 00000000 00001000Num2 = 11111111 11111111 11111111 11111000>> Operator : 8 >> 1 (Shift by one bit) :Num1 = 00000000 00000000 00000000 00000100Num2 = 11111111 11111111 11111111 11111100>>> Operator : 8 >>> 1 (Shift by one bit) =Num1 = 00000000 00000000 00000000 00000100Num2 = 01111111 11111111 11111111 11111100 Composition, and Aggregation help to build (Has - A - Relationship) between classes and objects. But both are not the same in the end. Lets understand with the help of an example.Consider the University as a class that has some departments in it. So the university will be the container object. And departments in it will contain objects. Now in this case, if the container object destroys then the contained objects will also get destroyed automatically. So here we can say that there is a strong association between the objects. So this Strong Association is called Composition.Now consider one more example. Suppose we have a class department and there are several professors' objects there in the department. Now if the department class is destroyed then the professor's object will become free to bind with other objects. Because container objects (Department) only hold the references of contained objects (Professors). So here is the weak association between the objects. And this weak association is called Aggregation. When a String is formed as a literal with the assistance of an assignment operator, it makes its way into the String constant pool so that String Interning can take place. This same object in the heap will be referenced by a different String if the content is the same for both of them.public bool checking() {String first = "InterviewBit";String second = "InterviewBit";if (first == second) return true;else return false;}The checking() function will return true as the same content is referenced by both the variables.Conversely, when a String formation takes place with the help of a new() operator, interning does not take place. The object gets created in the heap memory even if the same content object is present.public bool checking() {String first = new String("InterviewBit");String second = new String("InterviewBit");if (first == second) return true;else return false;}The checking() function will return false as the same content is not referenced by both the variables. Both new and newInstance() operators are used to creating objects. The difference is- that when we already know the class name for which we have to create the object then we use a new operator. But suppose we dont know the class name for which we need to create the object, Or we get the class name from the command line argument, or the database, or the file. Then in that case we use the newInstance() operator.The newInstance() keyword throws an exception that we need to handle. It is because there are chances that the class definition doesnt exist, and we get the class name from runtime. So it will throw an exception. Yes, it is possible for the program to go out of memory in spite of the presence of a garbage collector. Garbage collection assists in recognizing and eliminating those objects which are not required in the program anymore, in order to free up the resources used by them.In a program, if an object is unreachable, then the execution of garbage collection takes place with respect to that object. If the amount of memory required for creating a new object is not sufficient, then memory is released for those objects which are no longer in the scope with the help of a garbage collector. The memory limit is exceeded for the program when the memory released is not enough for creating new objects.Moreover, exhaustion of the heap memory takes place if objects are created in such a manner that they remain in the scope and consume memory. The developer should make sure to dereference the object after its work is accomplished. Although the garbage collector endeavors its level best to reclaim memory as much as possible, memory limits can still be exceeded.Lets take a look at the following example:List example = new LinkedList();while(true){example.add(new String("Memory Limit Exceeded"));} Concurrent execution of different processes is made possible by synchronization. When a particular resource is shared between many threads, situations may arise in which multiple threads require the same shared resource.Synchronization assists in resolving the issue and the resource is shared by a single thread at a time. Lets take an example to understand it more clearly. For example, you have a URL and you have to find out the number of requests made to it. Two simultaneous requests can make the count erratic.No synchronization:package anonymous;public class Counting { private int increase_counter; public int increase() { increase_counter = increase_counter + 1; return increase_counter; }}If a thread Thread1 views the count as 10, it will be increased by 1 to 11. Simultaneously, if another thread Thread2 views the count as 10, it will be increased by 1 to 11. Thus, inconsistency in count values takes place because the expected final value is 12 but the actual final value we get will be 11.Now, the function increase() is made synchronized so that simultaneous accessing cannot take place.With synchronization:package anonymous;public class Counting { private int increase_counter; public synchronized int increase() { increase_counter = increase_counter + 1; return increase_counter; }}If a thread Thread1 views the count as 10, it will be increased by 1 to 11, then the thread Thread2 will view the count as 11, it will be increased by 1 to 12. Thus, consistency in count values takes place. public void fooBarMethod(String... variables){ }Ability to provide ... is a feature called varargs (variable arguments) which was introduced as part of Java 5.The function having ... in the above example indicates that it can receive multiple arguments of the datatype String.For example, the fooBarMethod can be called in multiple ways and we can still have one method to process the data as shown below:fooBarMethod("foo", "bar");fooBarMethod("foo", "bar", "boo");fooBarMethod(new String[]{"foo", "var", "boo"});public void myMethod(String... variables){ for(String variable : variables){ }} class InterviewBit{ int i; static int j; { System.out.println(" Instance Block 1. Value of i = "+i); } static{ System.out.println(" Static Block 1. Value of j = "+j); method_2(); } { i = 5; } static{ j = 10; } InterviewBit(){ System.out.println(" Welcome to InterviewBit "); } public static void main(String[] args){ InterviewBit ib = new InterviewBit(); } public void method_1(){ System.out.println(" Instance method. "); } static{ System.out.println(" Static Block 2. Value of j = "+j); } { System.out.println(" Instance Block 2. Value of i = "+i); method_1(); } public static void method_2(){ System.out.println(" Static method. "); }}The Output we get by executing this program will beStatic Block 1. Value of j = 0Static method.Static Block 2. Value of j = 10Instance Block 1. Value of i = 0Instance Block 2. Value of i = 5Instance method.Welcome to InterviewBitThis is a java tricky interview question frequently asked in java interviews for the experienced. The output will be like this because, when the java program is compiled and gets executed, then there are various steps followed for execution. And the steps are -Identification of Static Members from top to bottom.Execution of Static variable assignment and a Static block from top to bottom.Execution of the main method.Identification of Instance Members from top to bottom.Execution of Instance variable assignment and Instance block from top to bottom.Execution of Constructor.In above steps from 4 to 6, will be executed for every object creation. If we create multiple objects then for every object these steps will be performed.Now from the above code, the execution will happen like this -1. In the step of identification of static members. It is found that -static int j.static block.main method.static method_2.During identification, the JVM will assign the default value in the static int j variable. Then it is currently in the state of reading and indirectly writing. Because the original value is not assigned.2. In the next step, it will execute the static block and assign the value in static variables.First static block it will print and because execution from top to bottom and original value in j is not assigned. So it will print the default value of 0.After executing

static block 1. It will execute the static method_1 because it is called from the static block 1.Then it will assign the original value of 5 in the j variable. And executes the remaining static block.3. Now it will execute the main method. In which it will create an object for the class InterviewBit. And then the execution of instances will happen.4. Identify the instance variables and blocks from top to bottom.int i.Instance block 1.Instance method_1.Like a static variable, the instance variable also has been initialized with the default value 0 and will be in the state of reading and writing indirectly.5. It will execute the instance methods and assign the original value to the instance variable.Prints the Instance block 1. And the current value of i is not assigned till now, so it will print 0.Assign the original value to i. Then print instance block 2. And after that instance method will be called and printed because it is being called in the instance block.6. And at the last step, the constructor will be invoked and the lines will be executed in the constructor.This is how the java program gets executed. System.out.println() is used to print the message on the console. System - It is a class present in java.lang package. Out is the static variable of type PrintStream class present in the System class. println() is the method present in the PrintStream class.So if we justify the statement, then we can say that if we want to print anything on the console then we need to call the println() method that was present in PrintStream class. And we can call this using the output object that is present in the System class. Java thread life cycle is as follows:New When the instance of the thread is created and the start() method has not been invoked, the thread is considered to be alive and hence in the NEW state.Runnable Once the start() method is invoked, before the run() method is called by JVM, the thread is said to be in RUNNABLE (ready to run) state. This state can also be entered from the Waiting or Sleeping state of the thread.Running When the run() method has been invoked and the thread starts its execution, the thread is said to be in a RUNNING state.Non-Runnable (Blocked/Waiting) When the thread is not able to run despite the fact of its aliveness, the thread is said to be in a NON-RUNNABLE state. Ideally, after some time of its aliveness, the thread should go to a runnable state.A thread is said to be in a Blocked state if it wants to enter synchronized code but it is unable to as another thread is operating in that synchronized block on the same object. The first thread has to wait until the other thread exits the synchronized block.A thread is said to be in a Waiting state if it is waiting for the signal to execute from another thread, i.e it waits for work until the signal is received.Terminated Once the run() method execution is completed, the thread is said to enter the TERMINATED step and is considered to not be alive.The following flowchart clearly explains the lifecycle of the thread in Java. The main advantage of having an ordered array is the reduced search time complexity of O(log n) whereas the time complexity in an unordered array is O(n).The main drawback of the ordered array is its increased insertion time which is O(n) due to the fact that its element has to reordered to maintain the order of array during every insertion whereas the time complexity in the unordered array is only O(1).Considering the above 2 key points and depending on what kind of scenario a developer requires, the appropriate data structure can be used for implementation. It is possible to import a class or package more than once, however, it is redundant because the JVM internally loads the package or class only once. This is a big NO. We need to understand that the importing of the sub-packages of a package needs to be done explicitly. Importing the parent package only results in the import of the classes within it and not the contents of its child/sub-packages. NO. The control of the program post System.exit(0) is immediately gone and the program gets terminated which is why the finally block never gets executed. Marker interfaces, also known as tagging interfaces are those interfaces that have no methods and constants defined in them. They are there for helping the compiler and JVM to get run time-related information regarding the objects. This is a convenient means of initializing any collections in Java. Consider the below example.import java.util.HashSet;import java.util.Set; public class IBDoubleBraceDemo{ public static void main(String[] args){ Set stringSets = new HashSet() { { add("set1"); add("set2"); add("set3"); } }; doSomething(stringSets); } private static void doSomething(Set stringSets){ System.out.println(stringSets); }}In the above example, we see that the stringSets were initialized by using double braces.The first brace does the task of creating an anonymous inner class that has the capability of accessing the parent classs behavior. In our example, we are creating the subclass of HashSet so that it can use the add() method of HashSet.The second braces do the task of initializing the instances.Care should be taken while initializing through this method as the method involves the creation of anonymous inner classes which can cause problems during the garbage collection or serialization processes and may also result in memory leaks. The length method returns the number of Unicode units of the String. Let's understand what Unicode units are and what is the confusion below.We know that Java uses UTF-16 for String representation. With this Unicode, we need to understand the below two Unicode related terms:Code Point: This represents an integer denoting a character in the code space.Code Unit: This is a bit sequence used for encoding the code points. In order to do this, one or more units might be required for representing a code point.Under the UTF-16 scheme, the code points were divided logically into 17 planes and the first plane was called the Basic Multilingual Plane (BMP). The BMP has classic characters - U+0000 to U+FFFF. The rest of the characters- U+10000 to U+10FFFF were termed as the supplementary characters as they were contained in the remaining planes.The code points from the first plane are encoded using one 16-bit code unitThe code points from the remaining planes are encoded using two code units.Now if a string contained supplementary characters, the length function would count that as 2 units and the result of the length() function would not be as per what is expected.In other words, if there is 1 supplementary character of 2 units, the length of that SINGLE character is considered to be TWO - Notice the inaccuracy here? As per the java documentation, it is expected, but as per the real logic, it is inaccurate. public class InterviewBit{ public static void main(String[] args) { System.out.println('b' + 'i' + 't'); }}bit would have been the result printed if the letters were used in double-quotes (or the string literals). But the question has the character literals (single quotes) being used which is why concatenation wouldn't occur. The corresponding ASCII values of each character would be added and the result of that sum would be printed.The ASCII values of b, i, t are:b = 98i = 105t = 11698 + 105 + 116 = 319Hence 319 would be printed. First Approach: Set the object references to null once the object creation purpose is served.public class IBGarbageCollect { public static void main (String [] args){ String s1 = "Some String"; s1 = null; } }Second Approach: Point the reference variable to another object. Doing this, the object which the reference variable was referencing before becomes eligible for GC.public class IBGarbageCollect { public static void main(String [] args){ String s1 = "To Garbage Collect"; String s2 = "Another Object"; System.out.println(s1); s1 = s2; }}Third Approach: Island of Isolation Approach: When 2 reference variables pointing to instances of the same class, and these variables refer to only each other and the objects pointed by these 2 variables don't have any other references, then it is said to have formed an Island of Isolation and these 2 objects are eligible for GC.public class IBGarbageCollect { IBGarbageCollect ib; public static void main(String [] str){ IBGarbageCollect ibgc1 = new IBGarbageCollect(); IBGarbageCollect ibgc2 = new IBGarbageCollect(); ibgc1.ib = ibgc2; ibgc2.ib = ibgc1; ibgc1 = null; ibgc2 = null; }} class Main{ public static void main(String[] args){ int[][] num = new int[3][]; num[0] = new int[5]; num[1] = new int[2]; num[2] = new int[3]; num[2] = new int[5]; num[0] = new int[4]; num[1] = new int[3]; num = new int[2][]; }}In the above program, a total of 7 objects will be eligible for garbage collection. Lets visually understand what's happening in the code.In the above figure on line 3, we can see that on each array index we are declaring a new array so the reference will be of that new array on all the 3 indexes. So the old array will be pointed to by none. So these three are eligible for garbage collection. And on line 4, we are creating a new array object on the older reference. So that will point to a new array and older multidimensional objects will become eligible for garbage collection. There is no boundation for using a particular dependency injection. But the recommended approach is -Setters are mostly recommended for optional dependencies injection, and constructor arguments are recommended for mandatory ones. This is because constructor injection enables the injection of values into immutable fields and enables reading them more easily. A scope can be set by an annotation such as the @Scope annotation or the "scope" attribute in an XML configuration file. Spring Bean supports the following five scopes:SingletonPrototypeRequestSessionGlobal-session Java Design patterns are categorized into the following different types. And those are also further categorized asStructural patterns:AdapterBridgeFilterCompositeDecoratorFacadeFlyweightProxyBehavioral patterns:InterpreterTemplate method/ patternChain of responsibilityCommand patternIterator patternStrategy patternVisitor patternJ2EE patterns:MVC PatternData Access Object patternFront controller patternIntercepting filter patternTransfer object patternCreational patterns:Factory method/TemplateAbstract FactoryBuilderPrototypeSingleton The Java Garbage Collector (GC) typically removes unused objects when they are no longer required, but when they are still referenced, the unused objects cannot be removed. So this causes the memory leak problem. Example - Consider a linked list like the structure below -In the above image, there are unused objects that are not referenced. But then also Garbage collection will not free it. Because it is referencing some existing referenced object. So this can be the situation of memory leak.Some common causes of Memory leaks are -When there are Unbounded caches.Excessive page swapping is done by the operating system.Improper written custom data structures.Inserting into a collection object without first deleting it.etc. A thread that has a lock won't be released even after it calls sleep(). Despite the thread sleeping for a specified period of time, the lock will not be released. import java.util.*;public class InterviewBit { public static void main(String args[]) { Scanner s = new Scanner(System.in); String word = s.nextLine(); System.out.println("Is "+word+" palindrome? - "+isWordPalindrome(word)); } public static boolean isWordPalindrome(String word){ String reverseWord = getReverseWord(word); if(word.equals(reverseWord)){ return true; } return false; } public static String getReverseWord(String word){ if(word == null || word.isEmpty()){ return word; } return word.charAt(word.length()- 1) + getReverseWord(word.substring(0, word.length() - 1)); } } class InterviewBit { public static void printFibonacci(int val_1, int val_2, int num){ if(num == 0) return; System.out.print(val_1 + val_2 + " "); printFibonacci(val_2, val_1+val_2, --num); } public static void main(String args[]) { System.out.println(" *** Fibonacci Series *** "); System.out.print("0 1 "); printFibonacci(0, 1, 10); }}In the above code, we are printing the base 2 Fibonacci values 0 and 1. And then based on the length of Fibonacci to be printed, we are using the helper function to print that. The main idea is to validate the length of strings and then if found equal, convert the string to char array and then sort the arrays and check if both are equal.import java.util.Arrays;import java.util.Scanner;public class InterviewBit { public static void main(String[] args) { Scanner s = new Scanner(System.in); System.out.print("First String: "); String string1 = s.nextLine(); System.out.print("Second String: "); String string2 = s.nextLine(); if(string1.length() == string2.length()) { char[] characterArray1 = string1.toCharArray(); char[] characterArray2 = string2.toCharArray(); Arrays.sort(characterArray1); Arrays.sort(characterArray2); boolean isAnagram = Arrays.equals(characterArray1, characterArray2); System.out.println("Anagram: "+ isAnagram); }} public class FindFactorial { public static void main(String[] args) { int num = 10; long factorialResult = 1l; for(int i = 1; i 1+6+3 = 10Step 2: 10 => 1+0 = 1 => Hence 163 is a magic numberpublic class IBMagicNumber{ public static void main(String[] args) { int num = 163; int sumOfDigits = 0; while (num > 0 || sumOfDigits > 9) { if (num == 0) { num = sumOfDigits; sumOfDigits = 0; } sumOfDigits += num % 10; num /= 10; } if(sumOfDigits == 1) { System.out.println("Magic number"); }else { System.out.println("Not magic number"); } } } class InterviewBit { public static void main(String args[]) throws CustomException { throw new CustomException(" This is my custom Exception "); }} class CustomException extends Exception{ public CustomException(String message){ super(message); }}We have created the exception class named with CustomException and called the base exception constructor with the error message that we want to print. And to avoid handling exceptions in the main method, we have used the throws keyword in the method declaration. class InterviewBit{ public static void main(String[] args){ String str = "Welcome to InterviewBit"; int i = 0, j = str.length()-1; char[] revString = new char[j+1]; while(i < j){ revString[j] = str.charAt(i); revString[i] = str.charAt(j); i++; j--; } System.out.println("Reversed String = " + String.valueOf(revString)); }}In the above code, we are storing the last character from the string to the first and the first value to the last in the output character array. And doing the same thing in the loop for the remaining 2nd to n-1 characters. This is how the string will be reversed. mport java.util.Scanner;public class InterviewBit{ public static void main(String[] args) { Scanner sc = new Scanner(System.in); int no; System.out.print("Enter size of Array : "); no = sc.nextInt(); int[][] a = new int[no][no]; System.out.print("Enter "+ no*no+" Element Array : "); for(int i = 0; i