Java Programming Question and Answer

---

1. What is Java?

Java is a high-level, object-oriented programming language.

2. Explain the main features of Java.

Java features include platform independence, object-oriented, distributed computing, and multithreading.

3. What is the difference between JDK, JRE, and JVM?

JDK (Java Development Kit) is for development, JRE (Java Runtime Environment) is for running Java applications, and JVM (Java Virtual Machine) executes Java bytecode.

4. How does Java achieve platform independence?

Java achieves platform independence by compiling code into bytecode, which is then interpreted by the Java Virtual Machine (JVM) on different platforms.

5. What is the main difference between `==` and `.equals()` in Java?

`==` compares object references, while `.equals()` compares object contents.

6. Explain the significance of the `main` method in Java.

The `main` method is the entry point for Java applications, and execution starts from this method.

7. What is the Java API?

The Java API (Application Programming Interface) provides a set of classes and methods for Java development.

8. How are exceptions handled in Java?

Exceptions in Java are handled using `try`, `catch`, `finally`, and `throw` keywords.

9. What is the difference between `final`, `finally`, and `finalize` in Java?

`final` is a keyword used to define constants, `finally` is a block used in exception handling, and `finalize` is a method called by the garbage collector.

10. Explain the concept of object-oriented programming in Java.

Object-oriented programming in Java involves the creation and manipulation of objects, encapsulation, inheritance, and polymorphism.

11. How is multiple inheritance achieved in Java?

Multiple inheritance is achieved through interfaces in Java.

12. What is the purpose of the `this` keyword in Java?

`this` refers to the current instance of the class and is used to differentiate instance variables from local variables.

13. How are static and instance variables different?

Static variables belong to the class and are shared among all instances, while instance variables are specific to an instance of the class.

14. What is the purpose of the `super` keyword in Java?

`super` is used to call the superclass methods, access superclass fields, and invoke the superclass constructor.

15. Explain the concept of method overloading in Java.

Method overloading allows a class to have multiple methods with the same name but different parameter lists.

16. What is the difference between `public`, `private`, `protected`, and `default` access modifiers?

`public` is accessible from any class, `private` is only accessible within the same class, `protected` is accessible within the same package and subclasses, and the default (no modifier) is accessible within the same package.

17. How does garbage collection work in Java?

Garbage collection in Java automatically reclaims memory by identifying and collecting objects without references.

18. Explain the difference between the `String`, `StringBuilder`, and `StringBuffer` classes.

`String` is immutable, `StringBuilder` is mutable and not thread-safe, and `StringBuffer` is mutable and thread-safe.

19. What is the purpose of the `static` keyword in Java?

The `static` keyword is used to define class-level variables and methods.

20. How do you achieve multithreading in Java?

Multithreading in Java is achieved by extending the `Thread` class or implementing the `Runnable` interface.

21. What are the primitive data types in Java?

The primitive data types in Java are `byte`, `short`, `int`, `long`, `float`, `double`, `char`, and `boolean`.

22. Explain the difference between `float` and `double` in Java.

`float` is a 32-bit IEEE 754 floating-point, and `double` is a 64-bit IEEE 754 floating-point.

23. How do you convert a string to an integer in Java?

Use the `Integer.parseInt()` method or `Integer.valueOf()` method.

24.Explain the concept of autoboxing and unboxing in Java. Autoboxing is the automatic conversion of a primitive type to its corresponding wrapper class, and unboxing is the reverse process.

25. What is the `instanceof` operator used for in Java?

The `instanceof` operator is used to test if an object is an instance of a particular class or interface.

26. What is the ternary operator in Java?

The ternary operator (`? :`) is a shorthand way of writing an `if-else` statement.

27. Explain the difference between the `++i` and `i++` operators in Java.

`++i` is the pre-increment operator (increments before the value is used), and `i++` is the post-increment operator (increments after the value is used).

28. How does the `switch` statement work in Java?

The `switch` statement evaluates an expression and executes the code block corresponding to the matched case.

29. What are bitwise operators in Java?

Bitwise operators perform operations on individual bits of binary numbers. Examples include `&` (AND), `|` (OR), `^` (XOR), `~` (NOT), `<<` (left shift), and `>>` (right shift).

30. What is the purpose of the `Math` class in Java?

The `Math` class provides mathematical operations and functions in Java.

31. What is the `for` loop in Java?

The `for` loop is a control flow statement that repeatedly executes a block of code based on a condition.

32. How does the enhanced `for` loop (for-each loop) work in Java?

The enhanced `for` loop simplifies iterating over arrays or collections.

33. Explain the `while` loop in Java.

The `while` loop repeatedly executes a block of code as long as a given condition is true.

34. What is the purpose of the `do-while` loop in Java?

The `do-while` loop is similar to the `while` loop but ensures that the block of code is executed at least once before checking the condition.

35. How do you break out of a loop in Java?

The `break` statement is used to exit a loop prematurely.

36. What is the `continue` statement in Java?

The `continue` statement is used to skip the rest of the loop's code and move to the next iteration.

37. Explain the concept of labeled loops in Java.

Labeled loops allow specifying a label for a loop, and the `break` or `continue`

38. What is Spring Boot?

Spring Boot is an extension of the Spring framework that simplifies the process of building production-ready applications with the Spring framework.

39. Explain the key features of Spring Boot..

Key features include automatic configuration, embedded servers, production-ready defaults, and a wide range of built-in tools.

40.How does Spring Boot simplify the configuration of Spring applications?.

Spring Boot uses convention over configuration and provides default settings, reducing the need for explicit configuration.

41.What is the purpose of the `@SpringBootApplication` annotation?.

`@SpringBootApplication` is a combination of `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`. It marks the main class of a Spring Boot application.

42.Explain the significance of the `application.properties` or `application.yml` file in Spring Boot..

It is used to configure application properties, such as database connection details, server port, and other settings.

43. How does Spring Boot handle external configuration?.

Spring Boot supports multiple property sources, including application properties, YAML files, environment variables, and command-line arguments.

44. What is the role of the `pom.xml` file in a Spring Boot project?.

The `pom.xml` file is a Maven Project Object Model file that contains project configuration details, dependencies, and plugins.

45. Explain the purpose of the Spring Boot Starter dependencies..

Spring Boot Starter dependencies simplify project setup by providing a set of common dependencies for specific tasks, such as web development, data access, or messaging.

46. What is the difference between the `@RestController` and `@Controller` annotations in Spring Boot?.

`@RestController` is a specialized version of `@Controller` that is used for RESTful web services. It includes the `@Controller` and `@ResponseBody` annotations.

47. How do you enable cross-origin resource sharing (CORS) in a Spring Boot application?.

Use the `@CrossOrigin` annotation on the controller or configure CORS globally in the `WebMvcConfigurer` bean.

48.Explain the role of the `@RequestMapping` annotation in Spring Boot..

`@RequestMapping` is used to map web requests to specific controller methods, specifying the URI template and HTTP methods.

49. What is the purpose of the `@PathVariable` annotation in Spring Boot?

`@PathVariable` is used to extract values from URI templates in the request URL.

50. How do you handle form data in a Spring Boot application?.

Use the `@RequestParam` annotation to bind form data to method parameters in a controller.

51. What is the significance of the `@RequestBody` annotation in Spring Boot?.

`@RequestBody` is used to bind the request body to a method parameter in a controller, especially in RESTful services.

52. Explain the difference between `@RequestParam` and `@PathVariable` in Spring Boot.

`@RequestParam` is used for query parameters in the request URL, while `@PathVariable` is used for values embedded in the URI.

53. How do you handle exceptions in a Spring Boot application?

Use the `@ExceptionHandler` annotation to handle exceptions globally or use `try-catch` blocks in specific methods.

54. What is the purpose of the `@ResponseStatus` annotation in Spring Boot?

`@ResponseStatus` is used to specify the HTTP status code returned by a controller method.

55. Explain the concept of content negotiation in Spring Boot..

Content negotiation allows a Spring Boot application to respond with different representations (JSON, XML, etc.) based on the client's requested media type.

56. What is the role of the `Model` interface in Spring Boot?.

The `Model` interface is used to pass data from a controller to a view.

57. How do you implement method-level security in Spring Boot?.

Use the `@PreAuthorize` and `@Secured` annotations to enforce security at the method level.

58.Explain the purpose of the Spring Data JPA repository..

The Spring Data JPA repository provides a set of CRUD operations for working with JPA entities.

59.How do you define a JPA entity in a Spring Boot application?.

Use the `@Entity` annotation to define a JPA entity class.

60.What is the purpose of the `@Repository` annotation in Spring Boot?.

`@Repository` is used to indicate that a class is a pring Data repository.

71.How do you perform database queries in Spring Boot using JPA?.

Use method names following the naming conventions in Spring Data JPA, or use JPQL or native queries with the `@Query` annotation.

72.What is the role of the `@Transactional` annotation in Spring Boot?.

`@Transactional` is used to specify that a method is transactional, ensuring that either all operations within the method succeed or none do.

73.Explain the difference between an inner join and an outer join in Spring Boot data repositories..

In Spring Data JPA, inner joins fetch only matching records, while outer joins fetch matching and non-matching records.

74.How do you enable second-level caching in Spring Boot with Hibernate?

Configure caching in the `application.properties` file and annotate entities with `@Cacheable`.

75. What is Spring Data REST, and how does it simplify building RESTful APIs?

Spring Data REST is an extension of Spring Data that automatically exposes JPA repositories as RESTful endpoints.

76.Explain the purpose of the `@Query` annotation in Spring Boot Data JPA.?

`@Query` is used to define custom queries using JPQL or native SQL in Spring Data JPA repositories.

77.How do you perform pagination in Spring Boot Data JPA?

Use the `Pageable` parameter in repository methods, and Spring Data JPA automatically handles pagination.

78.What is Spring Security, and how does it enhance security in a Spring Boot application?

Spring Security is a powerful and customizable authentication and access control framework for Java applications.

79.How do you configure basic authentication in Spring Boot Security?

Use the `SecurityConfigurerAdapter` to configure basic authentication with a username and password.

80. What is the difference between an Inner Class and a Sub-Class?

An Inner class is a class which is nested within another class. An Inner class has access rights for the class which is nesting it and it can access all variables and methods defined in the outer class.

A sub-class is a class which inherits from another class called super class. Sub-class can access all public and protected methods and fields of its super class.

81. What are the various access specifiers for Java classes?

In Java, access specifiers are the keywords used before a class name which defines the access scope. The types of access specifiers for classes are:

1) Public: Class,Method,Field is accessible from anywhere.

2) Protected: Method,Field can be accessed from the same class to which they belong or from the sub-classes, and from the class of same package, but not from outside.

3) Default: Method,Field,class can be accessed only from the same package and not from outside of it's native package.

4) Private: Method,Field can be accessed from the same class to which they belong.

82.What's the purpose of Static methods and static variables?

When there is a requirement to share a method or a variable between multiple objects of a class instead of creating separate copies for each object, we use static keyword to make a method or variable shared for all objects.

83. What is data encapsulation and what's its significance?

Encapsulation is a concept in Object Oriented Programming for combining properties and methods in a single unit.

Encapsulation helps programmers to follow a modular approach for software development as each object has its own set of methods and variables and serves its functions independent of other objects. Encapsulation also serves data hiding purpose.

84. What is a singleton class? Give a practical example of its usage.

A singleton class in java can have only one instance and hence all its methods and variables belong to just one instance. Singleton class concept is useful for the situations when there is a need to limit the number of objects for a class.

The best example of singleton usage scenario is when there is a limit of having only one connection to a database due to some driver limitations or because of any licensing issues.

85. What are Loops in Java? What are three types of loops?

Looping is used in programming to execute a statement or a block of statement repeatedly. There are three types of loops in Java:

1) For Loops

For loops are used in java to execute statements repeatedly for a given number of times. For loops are used when number of times to execute the statements is known to programmer.

2) While Loops

While loop is used when certain statements need to be executed repeatedly until a condition is fulfilled. In while loops, condition is checked first before execution of statements.

3) Do While Loops

Do While Loop is same as While loop with only difference that condition is checked after execution of block of statements. Hence in case of do while loop, statements are executed at least once.

86. What is an infinite Loop? How infinite loop is declared?

An infinite loop runs without any condition and runs infinitely. An infinite loop can be broken by defining any breaking logic in the body of the statement blocks.

Infinite loop is declared as follows:

```
for (;;)
{
   // Statements to execute
```

```
    // Add any loop breaking logic
}
```

87. What is the difference between continue and break statement?

break and continue are two important keywords used in Loops. When a break keyword is used in a loop, loop is broken instantly while when continue keyword is used, current iteration is broken and loop continues with next iteration.

In below example, Loop is broken when counter reaches 4.

```
for (counter = 0; counter & lt; 10; counter++)
    system.out.println(counter);

if (counter == 4) {

    break;
}

}
```


In the below example when counter reaches 4, loop jumps to next iteration and any statements after the continue keyword are skipped for current iteration.

```
for (counter = 0; counter < 10; counter++)
    system.out.println(counter);

if (counter == 4) {

    continue;
}
system.out.println("This will not get printed when counter is 4");
}
```

88. What is the difference between double and float variables in Java?

In java, float takes 4 bytes in memory while Double takes 8 bytes in memory. Float is single precision floating point decimal number while Double is double precision decimal number.

89. What is Final Keyword in Java? Give an example.

In java, a constant is declared using the keyword Final. Value can be assigned only once and after assignment, value of a constant can't be changed.

In below example, a constant with the name const_val is declared and assigned avalue:

Private Final int const_val=100

When a method is declared as final, it can NOT be overridden by the subclasses. This method are faster than any other method, because they are resolved at complied time.

When a class is declares as final, it cannot be subclassed. Example String, Integer and other wrapper classes.

90. What is ternary operator? Give an example.

Ternary operator , also called conditional operator is used to decide which value to assign to a variable based on a Boolean value evaluation. It's denoted as ?

In the below example, if rank is 1, status is assigned a value of "Done" else "Pending".

```
public class conditionTest {
   public static void main(String args[]) {
      String status;
      int rank = 3;
      status = (rank == 1) ? "Done" : "Pending";
      System.out.println(status);
   }
}
```

91. How can you generate random numbers in Java?

Using Math.random() you can generate random numbers in the range greater than or equal to 0.1 and less than 1.0

Using Random class in package java.util

92. What is default switch case? Give example.

In a switch statement, default case is executed when no other switch condition matches. Default case is an optional case .It can be declared only once all other switch cases have been coded.

In the below example, when score is not 1 or 2, default case is used.

```
public class switchExample {
   int score = 4;
   public static void main(String args[]) {
      switch (score) {
         case 1:
            system.out.println("Score is 1");
            break;
         case 2:
            system.out.println("Score is 2");
            break;
         default:
            system.out.println("Default Case");
      }
   }
}
```


93. What's the base class in Java from which all classes are derived?

java.lang.object

94. Can main() method in Java can return any data?

In java, main() method can't return any data and hence, it's always declared with a void return type.

95. What are Java Packages? What's the significance of packages?

In Java, package is a collection of classes and interfaces which are bundled together as they are related to each other. Use of packages helps developers to modularize the code and group the code for proper re-use. Once code has been packaged in Packages, it can be imported in other classes and used.

96. Can we declare a class as Abstract without having any abstract method?

Yes we can create an abstract class by using abstract keyword before class name even if it doesn't have any abstract method. However, if a class has even one abstract method, it must be declared as abstract otherwise it will give an error.

97. What's the difference between an Abstract Class and Interface in Java?

The primary difference between an abstract class and interface is that an interface can only possess declaration of public static methods with no concrete implementation while an abstract class can have members with any access specifiers (public, private etc) with or without concrete implementation.

Another key difference in the use of abstract classes and interfaces is that a class which implements an interface must implement all the methods of the interface while a class which inherits from an abstract class doesn't require implementation of all the methods of its super class.

A class can implement multiple interfaces but it can extend only one abstract class.

98. What are the performance implications of Interfaces over abstract classes?

Interfaces are slower in performance as compared to abstract classes as extra indirections are required for interfaces. Another key factor for developers to take into consideration is that any class can extend only one abstract class while a class can implement many interfaces.

Use of interfaces also puts an extra burden on the developers as any time an interface is implemented in a class; developer is forced to implement each and every method of interface.

99. Does Importing a package imports its sub-packages as well in Java?

In java, when a package is imported, its sub-packages aren't imported and developer needs to import them separately if required.

For example, if a developer imports a package university.*, all classes in the package named university are loaded but no classes from the sub-package are loaded. To load the classes from its sub-package (say department), developer has to import it explicitly as follows:

Import university.department.*

100. Can we declare the main method of our class as private?

In java, main method must be public static in order to run any application correctly. If main method is declared as private, developer won't get any compilation error however, it will not get executed and will give a runtime error.

101. How can we pass argument to a function by reference instead of pass by value?

In java, we can pass argument to a function only by value and not by reference.

102. How an object is serialized in java?

In java, to convert an object into byte stream by serialization, an interface with the name Serializable is implemented by the class. All objects of a class implementing serializable interface get serialized and their state is saved in byte stream.

103. When we should use serialization?

Serialization is used when data needs to be transmitted over the network. Using serialization, object's state is saved and converted into byte stream .The byte stream is transferred over the network and the object is re-created at destination.

104. Is it compulsory for a Try Block to be followed by a Catch Block in Java for Exception handling?

Try block needs to be followed by either Catch block or Finally block or both. Any exception thrown from try block needs to be either caught in the catch block or else any specific tasks to be performed before code abortion are put in the Finally block.

Java Interview Questions and Answers for Experienced

105. Is there any way to skip Finally block of exception even if some exception occurs in the exception block?

If an exception is raised in Try block, control passes to catch block if it exists otherwise to finally block. Finally block is always executed when an exception occurs and the only way to avoid execution of any statements in Finally block is by aborting the code forcibly by writing following line of code at the end of try block:

System.exit(0);

106. When the constructor of a class is invoked?

The constructor of a class is invoked every time an object is created with new keyword.

For example, in the following class two objects are created using new keyword and hence, constructor is invoked two times.


```
public class const_example {

   const_example() {

      system.out.println("Inside constructor");
   }
   public static void main(String args[]) {

      const_example c1 = new const_example();

      const_example c2 = new const_example();
   }
}
```

107. Can a class have multiple constructors?

Yes, a class can have multiple constructors with different parameters. Which constructor gets used for object creation depends on the arguments passed while creating the objects.

108. Can we override static methods of a class?

We cannot override static methods. Static methods belong to a class and not to individual objects and are resolved at the time of compilation (not at runtime).Even if we try to override static method, we will not get an complitaion error,nor the impact of overriding when running the code.

109. In the below example, what will be the output?

```
public class superclass {

   public void displayResult() {

      system.out.println("Printing from superclass");

   }

}

public class subclass extends superclass {

   public void displayResult() {

      system.out.println("Displaying from subClass");

      super.displayResult();

   }

   public static void main(String args[]) {

      subclass obj = new subclass();

      obj.displayResult();
```

```
    }

}
```

Ans: Output will be:

```
Displaying from subClass

Printing from superclass
```

110. Is String a data type in java?

String is not a primitive data type in java. When a string is created in java, it's actually an object of Java.Lang.String class that gets created. After creation of this string object, all built-in methods of String class can be used on the string object.

111. In the below example, how many String Objects are created?

```
String s1="I am Java Expert";

String s2="I am C Expert";

String s3="I am Java Expert";
```

In the above example, two objects of Java.Lang.String class are created. s1 and s3 are references to same object.

33) Why Strings in Java are called as Immutable?

In java, string objects are called immutable as once value has been assigned to a string, it can't be changed and if changed, a new object is created.

In below example, reference str refers to a string object having value "Value one".

```
String str="Value One";
```

When a new value is assigned to it, a new String object gets created and the reference is moved to the new object.

str="New Value";

34) What's the difference between an array and Vector?

An array groups data of same primitive type and is static in nature while vectors are dynamic in nature and can hold data of different data types.

112. What is multi-threading?

Multi threading is a programming concept to run multiple tasks in a concurrent manner within a single program. Threads share same process stack and running in parallel. It helps in performance improvement of any program.

113. Why Runnable Interface is used in Java?

Runnable interface is used in java for implementing multi threaded applications. Java.Lang.Runnable interface is implemented by a class to support multi threading.

114. What are the two ways of implementing multi-threading in Java?

Multi threaded applications can be developed in Java by using any of the following two methodologies:

1) By using Java.Lang.Runnable Interface. Classes implement this interface to enable multi threading. There is a Run() method in this interface which is implemented.

2) By writing a class that extend Java.Lang.Thread class.

115. When a lot of changes are required in data, which one should be a preference to be used? String or StringBuffer?

Since StringBuffers are dynamic in nature and we can change the values of StringBuffer objects unlike String which is immutable, it's always a good choice to use StringBuffer when data is being changed too much. If we use String in such a case, for every data change a new String object will be created which will be an extra overhead.

116. What's the purpose of using Break in each case of Switch Statement?

Break is used after each case (except the last one) in a switch so that code breaks after the valid case and doesn't flow in the proceeding cases too.

If break isn't used after each case, all cases after the valid case also get executed resulting in wrong results.

117. How garbage collection is done in Java?

In java, when an object is not referenced any more, garbage collection takes place and the object is destroyed automatically. For automatic garbage collection java calls either System.gc() method or Runtime.gc() method.

118. How we can execute any code even before main method?

If we want to execute any statements before even creation of objects at load time of class, we can use a static block of code in the class. Any statements inside this static block of code will get executed once at the time of loading the class even before creation of objects in the main method.

119. Can a class be a super class and a sub-class at the same time? Give example.

If there is a hierarchy of inheritance used, a class can be a super class for another class and a sub-class for another one at the same time.

In the example below, continent class is sub-class of world class and it's super class of country class.

```
public class world {

.....

}
public class continenet extends world {

.....
```

```
}
public class country extends continent {

.....

}
```

120. How objects of a class are created if no constructor is defined in the class?

Even if no explicit constructor is defined in a java class, objects get created successfully as a default constructor is implicitly used for object creation. This constructor has no parameters.

121. In multi-threading how can we ensure that a resource isn't used by multiple threads simultaneously?

In multi-threading, access to the resources which are shared among multiple threads can be controlled by using the concept of synchronization. Using synchronized keyword, we can ensure that only one thread can use shared resource at a time and others can get control of the resource only once it has become free from the other one using it.

122. Can we call the constructor of a class more than once for an object?

Constructor is called automatically when we create an object using new keyword. It's called only once for an object at the time of object creation and hence, we can't invoke the constructor again for an object after its creation.

123. There are two classes named classA and classB. Both classes are in the same package. Can a private member of classA can be accessed by an object of classB?

Private members of a class aren't accessible outside the scope of that class and any other class even in the same package can't access them.

124. Can we have two methods in a class with the same name?

We can define two methods in a class with the same name but with different number/type of parameters. Which method is to get invoked will depend upon the parameters passed.

For example in the class below we have two print methods with same name but different parameters. Depending upon the parameters, appropriate one will be called:

```
public class methodExample {

   public void print() {

      system.out.println("Print method without parameters.");

   }

   public void print(String name) {

      system.out.println("Print method with parameter");

   }

   public static void main(String args[]) {

      methodExample obj1 = new methodExample();

      obj1.print();

      obj1.print("xx");

   }

}
```


125.How can we make copy of a java object?

We can use the concept of cloning to create copy of an object. Using clone, we create copies with the actual state of an object.

Clone() is a method of Cloneable interface and hence, Cloneable interface needs to be implemented for making object copies.

125. What's the benefit of using inheritance?

Key benefit of using inheritance is reusability of code as inheritance enables sub-classes to reuse the code of its super class. Polymorphism (Extensibility) is another great benefit which allow new functionality to be introduced without effecting existing derived classes.

126. What's the default access specifier for variables and methods of a class?

Default access specifier for variables and method is package protected i.e variables and class is available to any other class but in the same package, not outside the package.

127. Give an example of use of Pointers in Java class.

There are no pointers in Java. So we can't use concept of pointers in Java.

128.How can we restrict inheritance for a class so that no class can be inherited from it?

If we want a class not to be extended further by any class, we can use the keyword Final with the class name.

In the following example, Stone class is Final and can't be extend

```
public Final Class Stone {
        // Class methods and Variables
}
```

129. What's the access scope of Protected Access specifier?

When a method or a variable is declared with Protected access specifier, it becomes accessible in the same class, any other class of the same package as well as a sub-class.

Modifier	Class	Package	Subclass	World
public Y	Y	Y	Y	y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private Y	N	N	N	N

130 What's difference between Stack and Queue?

Stack and Queue both are used as placeholder for a collection of data. The primary difference between a stack and a queue is that stack is based on Last in First out (LIFO) principle while a queue is based on FIFO (First In First Out) principle.

131. In java, how we can disallow serialization of variables?

If we want certain variables of a class not to be serialized, we can use the keyword transient while declaring them. For example, the variable trans_var below is a transient variable and can't be serialized:

```
public class transientExample {
        private transient trans_var;
   // rest of the code
}
```

132. How can we use primitive data types as objects?

Primitive data types like int can be handled as objects by the use of their respective wrapper classes. For example, Integer is a wrapper class for primitive data type int. We can apply different methods to a wrapper class, just like any other object.

133. Which types of exceptions are caught at compile time?

Checked exceptions can be caught at the time of program compilation. Checked exceptions must be handled by using try catch block in the code in order to successfully compile the code.

134. Describe different states of a thread.

A thread in Java can be in either of the following states:

Ready: When a thread is created, it's in Ready state.

Running: A thread currently being executed is in running state.

Waiting: A thread waiting for another thread to free certain resources is in waiting state.

Dead: A thread which has gone dead after execution is in dead state.

59) Can we use a default constructor of a class even if an explicit constructor is defined?

Java provides a default no argument constructor if no explicit constructor is defined in a Java class. But if an explicit constructor has been defined, default constructor can't be invoked and developer can use only those constructors which are defined in the class.

135. Can we override a method by using same method name and arguments but different return types?

The basic condition of method overriding is that method name, arguments as well as return type must be exactly same as is that of the method being overridden. Hence using a different return type doesn't override a method.

136. What will be the output of following piece of code?

```
public class operatorExample {
   public static void main(String args[]) {
      int x = 4;
      system.out.println(x++);
         }
}
```


In this case postfix ++ operator is used which first returns the value and then increments. Hence it's output will be 4.

137. A person says that he compiled a java class successfully without even having a main method in it? Is it possible?

main method is an entry point of Java class and is required for execution of the program however; a class gets compiled successfully even if it doesn't have a main method. It can't be run though.

138. Can we call a non-static method from inside a static method?

Non-Static methods are owned by objects of a class and have object level scope and in order to call the non-Static methods from a static block (like from a static main method), an object of the class needs to be created first. Then using object reference, these methods can be invoked.

139. What are the two environment variables that must be set in order to run any Java programs?

Java programs can be executed in a machine only once following two environment variables have been properly set:

PATH variable

CLASSPATH variable

140. Can variables be used in Java without initialization?

In Java, if a variable is used in a code without prior initialization by a valid value, program doesn't compile and gives an error as no default value is assigned to variables in Java.

141. Can a class in Java be inherited from more than one class?

In Java, a class can be derived from only one class and not from multiple classes. Multiple inheritances is not supported by Java.

142. Can a constructor have different name than a Class name in Java?

Constructor in Java must have same name as the class name and if the name is different, it doesn't act as a constructor and compiler thinks of it as a normal method.

143. What will be the output of Round(3.7) and Ceil(3.7)?

Round(3.7) returns 4 and Ceil(3.7) returns 4.

144. Can we use goto in Java to go to a particular line?

In Java, there is not goto keyword and java doesn't support this feature of going to a particular labeled line.

145. Can a dead thread be started again?

In java, a thread which is in dead state can't be started again. There is no way to restart a dead thread.

146. Is the following class declaration correct?

```
public abstract final class testClass {
        // Class methods and variables
}
```

Ans: The above class declaration is incorrect as an abstract class can't be declared as Final.

147. Is JDK required on each machine to run a Java program?

JDK is development Kit of Java and is required for development only and to run a Java program on a machine, JDK isn't required. Only JRE is required.

148. What's the difference between comparison done by equals method and == operator?

In Java, equals() method is used to compare the contents of two string objects and returns true if the two have same value while == operator compares the references of two string objects.

In the following example, equals() returns true as the two string objects have same values. However == operator returns false as both string objects are referencing to different objects:

```
public class equalsTest {

   public static void main(String args[]) {

      String str1 = new String("Hello World");

      String str2 = new String("Hello World");

      if (str1.equals(str2))

      { // this condition is true
```

```
        System.out.println("str1 and str2 are equal in terms of values");

    }

    if (str1 == str2) {

        //This condition is true

        System.out.println("Both strings are referencing same object");

    } else

    {

        // This condition is NOT true

        System.out.println("Both strings are referencing different objects");

    }

  }

}
```


149. Is it possible to define a method in Java class but provide it's implementation in the code of another language like C?

Yes, we can do this by use of native methods. In case of native method based development, we define public static methods in our Java class without its implementation and then implementation is done in another language like C separately.

150. How are destructors defined in Java?

In Java, there are no destructors defined in the class as there is no need to do so. Java has its own garbage collection mechanism which does the job automatically by destroying the objects when no longer referenced.

151. Can a variable be local and static at the same time?

No a variable can't be static as well as local at the same time. Defining a local variable as static gives compilation error.

152. Can we have static methods in an Interface?

Static methods can't be overridden in any class while any methods in an interface are by default abstract and are supposed to be implemented in the classes being implementing the interface. So it makes no sense to have static methods in an interface in Java.

153. In a class implementing an interface, can we change the value of any variable defined in the interface?

No, we can't change the value of any variable of an interface in the implementing class as all variables defined in the interface are by default public, static and Final and final variables are like constants which can't be changed later.

154. Is it correct to say that due to garbage collection feature in Java, a java program never goes out of memory?

Even though automatic garbage collection is provided by Java, it doesn't ensure that a Java program will not go out of memory as there is a possibility that creation of Java objects is being done at a faster pace compared to garbage collection resulting in filling of all the available memory resources.

So, garbage collection helps in reducing the chances of a program going out of memory but it doesn't ensure that.

155. Can we have any other return type than void for main method?

No, Java class main method can have only void return type for the program to get successfully executed.

Nonetheless , if you absolutely must return a value to at the completion of main method , you can use System.exit(int status)

156. I want to re-reach and use an object once it has been garbage collected. How it's possible?

Once an object has been destroyed by garbage collector, it no longer exists on the heap and it can't be accessed again. There is no way to reference it again.

157. In Java thread programming, which method is a must implementation for all threads?

Run() is a method of Runnable interface that must be implemented by all threads.

158. I want to control database connections in my program and want that only one thread should be able to make database connection at a time. How can I implement this logic?

This can be implemented by use of the concept of synchronization. Database related code can be placed in a method which hs synchronized keyword so that only one thread can access it at a time.

159. How can an exception be thrown manually by a programmer?

In order to throw an exception in a block of code manually, throw keyword is used. Then this exception is caught and handled in the catch block.

```
public void topMethod() {
   try {
      excMethod();
   } catch (ManualException e) {}
}
```

```
public void excMethod {
   String name = null;
   if (name == null) {
      throw (new ManualException("Exception thrown manually ");
      }
   }
```

160. I want my class to be developed in such a way that no other class (even derived class) can create its objects. How can I do so?

If we declare the constructor of a class as private, it will not be accessible by any other class and hence, no other class will be able to instantiate it and formation of its object will be limited to itself only.

161. How objects are stored in Java?

In java, each object when created gets a memory space from a heap. When an object is destroyed by a garbage collector, the space allocated to it from the heap is re-allocated to the heap and becomes available for any new objects.

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

In java, there is no way to find out the exact size of an object on the heap.

163. Which of the following classes will have more memory allocated?

Class A: Three methods, four variables, no object

Class B: Five methods, three variables, no object

Memory isn't allocated before creation of objects. Since for both classes, there are no objects created so no memory is allocated on heap for any class.

164. What happens if an exception is not handled in a program?

If an exception is not handled in a program using try catch blocks, program gets aborted and no statement executes after the statement which caused exception throwing.

165. I have multiple constructors defined in a class. Is it possible to call a constructor from another constructor's body?

If a class has multiple constructors, it's possible to call one constructor from the body of another one using this().

166. What's meant by anonymous class?

An anonymous class is a class defined without any name in a single line of code using new keyword.

For example, in below code we have defined an anonymous class in one line of code:

```
public java.util.Enumeration testMethod()
{
   return new java.util.Enumeration()
   {
      @Override
      public boolean hasMoreElements()
      {
         // TODO Auto-generated method stub
         return false;
      }
      @Override
      public Object nextElement()

      {
         // TODO Auto-generated method stub
         return null;
      }
   }
```

167. Is there a way to increase the size of an array after its declaration?

Arrays are static and once we have specified its size, we can't change it. If we want to use such collections where we may require a change of size (no of items), we should prefer vector over array.

168. If an application has multiple classes in it, is it okay to have a main method in more than one class?

If there is main method in more than one classes in a java application, it won't cause any issue as entry point for any application will be a specific class and code will start from the main method of that particular class only.

169. I want to persist data of objects for later use. What's the best approach to do so?

The best way to persist data for future use is to use the concept of serialization.

170. What is a Local class in Java?

In Java, if we define a new class inside a particular block, it's called a local class. Such a class has local scope and isn't usable outside the block where its defined.

171. String and StringBuffer both represent String objects. Can we compare String and StringBuffer in Java?

Although String and StringBuffer both represent String objects, we can't compare them with each other and if we try to compare them, we get an error.

172. What is the difference between == and .equals() in Java?

== is used for comparing object references, checking if they point to the same memory location. .equals() is a method used to compare the content or values of objects.

173. What are the access modifiers in Java and what do they signify?

Access modifiers in Java are: public, private, protected, and default (no modifier). They control the accessibility of classes, variables, methods, etc. public allows access from anywhere, private restricts access to within the class, protected allows access within the package and subclasses, and the default modifier allows access within the package.

174. Explain the difference between ArrayList and LinkedList.

ArrayList uses a dynamic array for storing elements and is good for accessing elements randomly. LinkedList uses a doubly linked list, which provides better performance in insertion and deletion operations, especially for large data sets.

175. What is a Java interface?

An interface in Java is a blueprint of a class that specifies a set of methods that a class must implement. It provides a way to achieve abstraction and multiple inheritance in Java.

176. What is the difference between an abstract class and an interface?

An abstract class can have abstract and non-abstract methods, whereas an interface can only have abstract methods. A class can implement multiple interfaces, but it can inherit only one abstract class.

177. What is method overloading and method overriding in Java?

Method overloading is the process of having multiple methods in the same class with the same name but different parameters. Method overriding occurs when a subclass provides a specific implementation of a method that is already defined in its superclass.

178. Explain the concept of multithreading in Java.

Multithreading in Java allows concurrent execution of two or more parts of a program for maximum utilization of CPU. It enhances responsiveness and allows for better performance by executing multiple threads simultaneously.

179. What is the difference between final, finally, and finalize in Java?

final is used to restrict further modification, finally is a block used in exception handling to ensure execution whether an exception is handled or not, and finalize is a method used for garbage collection, called before an object is destroyed.

180. How does exception handling work in Java?

Java uses try-catch blocks for handling exceptions. Code that might throw exceptions is placed inside the try block, and the handling mechanism is defined in the catch block. Additionally, there's an optional finally block for code that needs to be executed regardless of whether an exception occurs or not.

181. Explain the concept of inheritance in Java.

Inheritance is a mechanism in Java where a new class (subclass) is derived from an existing class (superclass). The subclass inherits the properties and behaviors (methods and fields) of the superclass, allowing code reusability and promoting the concept of hierarchy.

182. What are the different types of inheritance in Java?

Java supports single, multilevel, hierarchical, and multiple inheritances through interfaces.

183. What is a constructor, and what is its purpose in Java?

A constructor is a special method in Java that is used to initialize objects. Its purpose is to initialize the state of an object when it's created.

184. What is method overriding? Can you provide an example?

Method overriding occurs when a subclass provides a specific implementation of a method that is already defined in its superclass. Example:

```
class Animal {
    void makeSound() {
        System.out.println("Some sound");
    }
}
class Dog extends Animal {
    @Override
    void makeSound() {
        System.out.println("Bark");
    }
}
```


185. Explain the difference between static and final keywords in Java.

static is used to create variables and methods that belong to the class rather than to instances of the class. final is used to restrict further modification of classes, methods, and variables.

186. What is the purpose of the this keyword in Java?

The this keyword refers to the current instance of the class and is used to differentiate between instance variables and parameters with the same name within a method.

187. What is a package in Java? Why is it used?

A package in Java is a mechanism to encapsulate a group of classes, interfaces, enumerations, and sub-packages. It helps in organizing and managing Java classes, preventing naming conflicts, and providing access control.

188. What is method overloading? Provide an example.

Method overloading is the process of having multiple methods in the same class with the same name but different parameters. Example:

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }
    double add(double a, double b) {
        return a + b;
    }
}
```


189. Explain the static keyword in Java.

static keyword is used to create variables and methods that belong to the class rather than to instances of the class. Static variables are shared among all instances of the class, and static methods can be called without creating an instance of the class.

190. What is the difference between break and continue statements in Java?

break statement is used to terminate the loop or switch statement, while continue statement is used to skip the current iteration of a loop and continue with the next iteration.

191. Can you explain the super keyword in Java?

The super keyword in Java is used to refer to the immediate parent class object. It is used to access methods, variables, and constructors of the superclass.

192. What is the difference between String, StringBuffer, and StringBuilder?

String is immutable (cannot be changed), StringBuffer is synchronized (thread-safe) but slower, while StringBuilder is not synchronized (not thread-safe) but faster.

193. What is the purpose of the transient keyword in Java?

The transient keyword is used to indicate that a variable should not be serialized when the object containing that variable is serialized.

195. Explain the concept of method chaining in Java.

Method chaining involves invoking multiple methods on the same object in a single line, where each method returns the object itself. It enhances code readability and conciseness.

196. What is the difference between HashMap and HashTable in Java?

HashMap is not synchronized and allows null values and one null key, while HashTable is synchronized and doesn't allow null keys or values.

197. What is the purpose of the finalize() method?

The finalize() method is called by the garbage collector before an object is destroyed. It can be overridden to perform cleanup operations before the object is garbage collected.

198. What is the difference between the throw and throws keywords in Java?

throw is used to explicitly throw an exception, while throws is used in the method signature to declare the exceptions that a method might throw.

199. What is a lambda expression in Java?

A lambda expression is an anonymous function that allows the concise representation of a method interface. It enables the writing of shorter and more readable code.

200. Explain the difference between the stack and the heap in Java.

The stack is used for static memory allocation, containing method calls, local variables, etc., and follows a last-in, first-out (LIFO) approach. The heap is used for dynamic memory allocation, containing objects, and follows no particular order for accessing elements.

201. What is the assert keyword used for in Java?

The assert keyword is used to perform assertion testing during development. It helps in debugging by testing assumptions about the program.

202. What are checked and unchecked exceptions in Java?

Checked exceptions are checked at compile time and need to be handled using try-catch or declaring them in the method using throws. Unchecked exceptions are not checked at compile time and include runtime exceptions.

203. Explain the clone() method in Java.

The clone() method is used to create a copy of an object. To use clone(), a class must implement the Cloneable interface and override the clone() method.

204. How does Java handle memory management and garbage collection?

Java uses automatic memory management through garbage collection. The JVM identifies and removes objects that are no longer reachable to free up memory.

205. What is the difference between ArrayList and LinkedList in terms of performance?

ArrayList provides better performance for accessing elements randomly, while LinkedList is better for frequent insertion and deletion operations.

206. What is the purpose of the volatile keyword in Java?

The volatile keyword is used to indicate that a variable may be modified asynchronously by multiple threads, ensuring that changes are visible to all threads.

207. Explain the concept of polymorphism in Java.

Polymorphism refers to the ability of a variable, method, or object to take multiple forms. It includes method overriding and method overloading in Java.

208. What is the difference between an interface and an abstract class?

An interface can only have abstract methods and cannot have method implementations, while an abstract class can have both abstract and concrete methods.

209. What is the purpose of the try-with-resources statement in Java?

try-with-resources is used to automatically close resources that implement AutoCloseable or Closeable interfaces, ensuring proper resource management and preventing resource leaks.

210. Explain the concept of JavaBeans.

JavaBeans are reusable software components that follow specific naming conventions, enabling easy integration and extension in various Java applications.

211. How can you prevent a Java class from being inherited?

You can prevent a class from being inherited by using the final keyword in the class declaration.

212. What is the purpose of the System.arraycopy() method in Java?

System.arraycopy() is used to copy elements from one array to another with more efficiency than a loop. It provides native implementation for array copying.

213. What is the difference between compareTo() and equals() methods in Java?

compareTo() is used to compare two objects based on their natural ordering. equals() is used to check if two objects are equal.

214. Explain the super() keyword in Java.

The super() keyword is used to call the constructor of the superclass. It is used to initialize the superclass part of an object.

215. What is the purpose of the instanceof operator in Java?

The instanceof operator is used to test whether an object is an instance of a particular class, subclass, or interface.

216. What are the access specifiers used for methods in an interface?

All methods in an interface are implicitly public and abstract. Since Java 8, default and static methods can also have implementations in interfaces.

217. Explain the concept of method references in Java.

Method references provide a way to refer to methods or constructors without invoking them. They can be used to make code more concise, especially when working with lambda expressions.

218. What is the purpose of the java.lang.Math class in Java?

The java.lang.Math class provides methods for performing basic numeric operations like trigonometric, logarithmic, exponential functions, etc.

219. What is the purpose of the strictfp keyword in Java?

The strictfp keyword is used to force floating-point calculations to adhere to the IEEE 754 standard, ensuring consistent results across different platforms.

220. What is a marker interface in Java? Can you provide an example?

A marker interface in Java is an interface with no methods or fields. Its sole purpose is to mark or tag a class to provide some useful information to the compiler or runtime. An example is the Serializable interface.

221. What is the purpose of the java.lang.Object class in Java?

The java.lang.Object class is the root class for all Java classes. It provides basic functionalities like toString(), equals(), hashCode(), etc., which are inherited by all Java classes.

222. Explain the Enum class in Java.

Enums in Java are special data types that consist of a fixed set of constants. They can contain methods, constructors, and instance variables.

223. What is the purpose of the static block in Java?

The static block in Java is used to initialize static variables or to perform one-time initialization tasks for a class.

224. What is the difference between this() and super() in Java constructors?

this() is used to call another constructor within the same class, while super() is used to call a constructor from the superclass.

225. What is the purpose of the java.lang.StringBuilder class?

StringBuilder is used to create mutable (modifiable) string objects in Java, allowing manipulation of characters in a string.

226. What is the diamond operator (<>) in Java generics?

The diamond operator (<>) is used for type inference in Java generics. It allows you to instantiate a generic class without explicitly specifying the type.

227. What is a try-with-resources statement in Java and why is it used?

try-with-resources is used to automatically close resources that implement AutoCloseable interfaces, providing automatic resource management.

228. What is the purpose of the java.lang.System class in Java?

The java.lang.System class provides access to system resources such as input and output streams, system properties, environment variables, etc.

229. Explain the StringTokenizer class in Java.

StringTokenizer is used to break a string into tokens based on specified delimiters. It's a legacy class and has been mostly replaced by String.split() or regex.

230. What is the purpose of the assert keyword in Java?

The assert keyword is used for assertion testing during development to check assumptions about the program and catch logical errors.

231. What is the purpose of the break statement in Java?

The break statement is used to terminate a loop or switch statement prematurely, transferring control out of the loop or switch.

232. Explain the purpose of the Comparator interface in Java.

The Comparator interface in Java is used for custom sorting of objects. It defines methods to compare objects based on specific criteria.

233. What is the purpose of the ObjectOutputStream and ObjectInputStream classes in Java?

ObjectOutputStream and ObjectInputStream classes are used to serialize and deserialize Java objects respectively, allowing objects to be written to a stream and reconstructed later.

234. Explain the difference between method hiding and method overriding in Java.

Method hiding occurs when a subclass defines a static method with the same signature as a static method in its superclass, while method overriding occurs when a subclass provides a specific implementation of a method from its superclass.

235. What is the purpose of the instanceof operator in Java?

The instanceof operator is used to check if an object is an instance of a particular class or interface. It returns true if the object is an instance; otherwise, it returns false.

236. What is the purpose of the finalize() method in Java?

The finalize() method is called by the garbage collector before an object is garbage collected. It can be overridden to perform cleanup operations or release resources.

237. What are the different types of exceptions in Java?

Exceptions in Java are broadly categorized into checked exceptions (compile-time checked) and unchecked exceptions (runtime exceptions).

238. Explain the difference between the continue and break statements in Java.

The continue statement is used to skip the current iteration of a loop and continue with the next iteration, while the break statement is used to terminate the loop or switch statement prematurely.

239. What is the purpose of the default method in Java interfaces?

The default method in Java interfaces provides a default implementation for a method. It allows adding new methods to interfaces without breaking existing implementations.

240. What is the purpose of the Class class in Java?

The Class class in Java is used to represent classes and interfaces at runtime. It provides methods to examine the runtime properties of classes.

241. What is the purpose of the Thread class in Java?

The Thread class in Java is used to create and manage threads. It allows concurrent execution of multiple parts of a program to achieve better performance.

242. Explain the purpose of the synchronized keyword in Java.

The synchronized keyword is used to control access to critical sections of code in a multithreaded environment, ensuring that only one thread can access the synchronized block at a time.

243. What is the purpose of the Math.random() method in Java?

Math.random() is used to generate random numbers between 0.0 (inclusive) and 1.0 (exclusive).

244. What is the purpose of the try block in exception handling?

The try block is used to enclose the code that might throw exceptions. It's followed by catch or finally blocks to handle exceptions or execute cleanup code.

245. Explain the difference between ArrayList and Vector in Java.

ArrayList is not synchronized, while Vector is synchronized. This makes ArrayList more efficient in single-threaded scenarios, but Vector is safer in multithreaded environments.

246. What is a static import in Java?

Static import is a feature that allows static members (fields and methods) of a class to be used in the code without specifying the class name.

247. Explain the purpose of the System.exit() method in Java.

System.exit() is used to terminate the currently running Java Virtual Machine (JVM) with an exit status. It immediately stops the program.

248. What is the purpose of the ClassCastException in Java?

ClassCastException is thrown when an object is cast to an incompatible class.

249.What is the purpose of the String.format() method in Java?

String.format() is used to format strings based on specified formatting patterns.

248. Explain the forEach method in Java streams.

The forEach method in Java streams is used to iterate through elements of a stream and perform an action for each element.

249. What is the purpose of the System.arraycopy() method in Java?

System.arraycopy() is used to efficiently copy data from one array to another.

250. What is the purpose of the volatile keyword in Java?

The volatile keyword is used to declare a variable as a shared variable among multiple threads, ensuring that its value is always read from and written to the main memory.

Cleancode