Computer Science
P-303 (Core Java)

Fill in the Blanks

1. Java was developed by _____ at _____ in the year _____. Answer: James Gosling, Sun Microsystems, 1995
2. Java is known for its _____ which allows programs to run on different platforms without modification. Answer: Platform independence
3. The Java programming language uses a _____ architecture. Answer: Write Once, Run Anywhere (WORA)
4. In Java, source code is compiled into _____ code, which is then executed by the Java Virtual Machine (JVM). Answer: Bytecode
5. The main method in Java has the following signature: _____. Answer: **public static void main(String[] args)**
6. In Java, variables are declared using the _____ keyword. Answer: **int**, **String**, etc.
7. Constants in Java are declared using the _____ keyword. Answer: **final**
8. The _____ keyword is used to refer to the current object instance within a class. Answer: **this**
9. The _____ keyword is used to call a method or constructor of the parent class. Answer: **super**
10. An _____ class cannot be instantiated, and it can only be used as a base for other classes. Answer: **abstract**
11. The _____ keyword is used to declare a method, variable, or class as accessible without needing an instance of the class. Answer: **static**
12. In Java, inheritance is implemented using the _____ keyword. Answer: **extends**
13. Interfaces are implemented using the _____ keyword. Answer: **implements**
14. A class can implement multiple interfaces using _____. Answer: **,** (comma)
15. Java supports various data types, such as _____ and _____. Answer: **int**, **double**
16. _____ are classes that wrap primitive data types, allowing them to be used as objects. Answer: Wrapper classes
17. Arithmetic operators in Java include _____ and _____. Answer: **+**, **-**
18. _____ operators in Java include **&&**, **||**, and **!**. Answer: Logical
19. _____ operators in Java include **&**, **|**, and **^**. Answer: Bitwise
20. An _____ is a combination of variables, operators, and method calls that evaluates to a single value. Answer: Expression
21. Java comments are used for _____ and are not executed by the compiler. Answer: Documentation
22. You can print output to the console in Java using the _____ method. Answer: **System.out.println()**
23. Java provides _____ statements for decision making. Answer: **if**, **else if**, **else**
24. _____ is used for looping in Java. Answer: **while**, **for**, **do-while**
25. Nesting _____ within _____ is a common practice in Java. Answer: loops, loops
26. In Java, methods are defined using the _____ keyword. Answer: **void**, **int**, etc.
27. The _____ of a variable determines where it can be accessed in a program. Answer: Scope
28. Arguments passed to a method are known as _____. Answer: Parameters
29. The _____ keyword is used to return a value from a method. Answer: **return**
30. Java provides several built-in class methods, such as _____ for input and _____ for mathematical operations. Answer: **Scanner**, **Math**

1. In object-oriented programming, the fundamental concept is _____. Answer: Objects
2. A class in Java is a _____ for creating objects. Answer: Blueprint
3. _____ are used to represent the attributes and behaviors of an object in a class. Answer: Fields (or instance variables)
4. _____ methods are associated with the class itself, rather than with individual objects. Answer: Class
5. An object is an _____ of a class. Answer: Instance
6. Object references in Java store the memory address of the _____. Answer: Object
7. When you pass an object as a parameter to a method, you are passing the _____ of the object. Answer: Reference
8. _____ classes cannot be extended or inherited. Answer: Final

9. The process of automatically deallocating memory occupied by unreachable objects is called _____. Answer: Garbage Collection
10. In Java, a _____ is a special type of method that initializes an object. Answer: Constructor
11. The _____ keyword is used to call a constructor from another constructor in the same class. Answer: **this**
12. The _____ keyword is used to call a constructor in the superclass. Answer: **super**
13. Method overloading allows multiple methods in the same class with the same _____ but different parameters. Answer: Name
14. Constructor overloading is having multiple constructors in a class with different _____. Answer: Parameter lists
15. _____ is a relationship where one class contains an object of another class. Answer: Aggregation
16. _____ is a relationship where one class inherits properties and behaviors from another class. Answer: Inheritance
17. In Java, the keyword _____ is used to establish inheritance between classes. Answer: **extends**
18. In Java, a class can implement multiple interfaces using the _____ keyword. Answer: **implements**
19. Types of inheritance in Java include _____ and _____. Answer: Single inheritance, Multiple inheritance
20. An _____ is a contract that specifies a set of methods that a class must implement. Answer: Interface
21. _____ is the process of casting an object reference to a superclass reference. Answer: Up-Casting
22. _____ is the process of casting a superclass reference to a subclass reference. Answer: Down-Casting
23. _____ is the automatic conversion of a primitive data type to its corresponding wrapper class object. Answer: Auto-Boxing
24. _____ provide a way to represent a fixed set of values as named constants. Answer: Enumerations (Enums)
25. Polymorphism allows objects of different classes to be treated as objects of a _____ class. Answer: Common superclass
26. Method overriding occurs when a subclass provides a specific implementation of a method that is already defined in the _____ class. Answer: Superclass
27. In Java, you cannot override a _____ method. Answer: Final
28. Java uses packages to organize classes and avoid _____. Answer: Naming conflicts
29. Java has several _____ packages for common tasks like input/output and data structures. Answer: Pre-defined
30. To create custom packages in Java, you use the _____ statement. Answer: **package**

1. An array is a collection of _____ of the same data type. Answer: Elements
2. In Java, arrays are zero-based, meaning the index of the first element is _____. Answer: 0
3. To create a one-dimensional array in Java, you declare it like this: _____[] arrayName. Answer: DataType
4. A two-dimensional array is an array of _____ arrays. Answer: One-dimensional
5. A three-dimensional array is an array of _____ arrays. Answer: Two-dimensional
6. A jagged array is an array of _____ arrays, where each sub-array can have a different length. Answer: One-dimensional
7. To create an array of objects in Java, you use the _____ keyword. Answer: **new**
8. You can dynamically reference array elements using square brackets and the _____ operator. Answer: Index

Strings and I/O:

9. In Java, strings are represented using the _____ class. Answer: **String**
10. To create a string object in Java, you can use _____. Answer: Double quotes or the **new** keyword
11. String objects in Java are _____. Answer: Immutable
12. To compare two strings for equality in Java, you should use the _____ method. Answer: **equals()**
13. You can concatenate strings in Java using the _____ operator. Answer: **+**
14. When passing a string to a method, Java uses _____ to pass it by reference. Answer: Reference (or Pass by reference)

15. The _____ class is used to efficiently manipulate and modify string data. Answer: **StringBuilder**
16. The _____ class is similar to **StringBuilder** but is thread-safe. Answer: **StringBuffer**
17. Java I/O operations are handled through the _____ package. Answer: **java.io**
18. The _____ class provides methods for file-related operations in Java. Answer: **File**
19. To create a file object in Java, you use the _____ constructor. Answer: **File(String pathname)**
20. To read from a file, you can use _____ streams like **FileInputStream** and **FileReader**. Answer: Character
21. To write to a file, you can use _____ streams like **FileOutputStream** and **FileWriter**. Answer: Character
22. The _____ class can be used to read character data from an input stream. Answer: **InputStreamReader**
23. The _____ class can be used to write character data to an output stream. Answer: **OutputStreamWriter**
24. To read and write binary data, you can use _____ streams like **FileInputStream** and **FileOutputStream**. Answer: Byte
25. The _____ class provides formatted output capabilities in Java. Answer: **PrintStream**
26. The _____ class is used to write formatted text to a file. Answer: **PrintWriter**
27. Compressing and uncompressing files in Java can be done using the _____ and _____ classes. Answer: **ZipOutputStream**, **ZipInputStream**

Java Exception Handling:

1. Exception handling in Java is used to manage _____ situations that can occur during program execution. Answer: Error
2. An exception is an _____ event that occurs during program execution. Answer: Unusual
3. The process of transferring control from the normal flow of execution to an exception handler is called _____. Answer: Exception propagation
4. An uncaught exception in Java results in the program _____. Answer: Terminating abnormally
5. The _____ keyword is used to explicitly throw an exception in Java. Answer: **throw**
6. Java provides a set of _____ exceptions, such as **NullPointerException** and **ArrayIndexOutOfBoundsException**. Answer: Built-in
7. You can create your own custom exceptions by extending the _____ class. Answer: **Exception** or a subclass of **Exception**

Java Threading:

8. In Java, multi-threading is achieved by using the _____ class and implementing the _____ interface. Answer: **Thread**, **Runnable**
9. To create a thread in Java, you can extend the _____ class and override the **run()** method. Answer: **Thread**
10. Alternatively, you can create a thread by passing an instance of a class that implements the _____ interface to the **Thread** constructor. Answer: **Runnable**
11. Thread prioritization in Java is done using _____. Answer: Priority values (integer values)
12. Synchronization in Java is used to prevent _____ among threads. Answer: Race conditions
13. In Java, you can use the _____ keyword to mark a method as synchronized. Answer: **synchronized**
14. Communication between threads can be achieved using _____ methods like **wait()** and **notify()**. Answer: Object monitor
15. Suspending a thread in Java can be done using the _____ method. Answer: **suspend()**
16. Resuming a suspended thread is accomplished with the _____ method. Answer: **resume()**

Java Networking:

17. The Java networking functionality is provided by the _____ package. Answer: **java.net**
18. TCP/IP is a _____-layered protocol used for internet communication. Answer: Four
19. _____ sockets provide a reliable, stream-oriented communication channel in Java. Answer: TCP
20. _____ sockets provide an unreliable, datagram-oriented communication channel in Java. Answer: UDP
21. The _____ class in Java is used for creating server sockets. Answer: **ServerSocket**
22. The _____ class in Java is used for creating client sockets. Answer: **Socket**

Database Connectivity with JDBC:

23. JDBC stands for _____. Answer: Java Database Connectivity
24. To connect to a database using JDBC, you need a _____ driver. Answer: Database-specific

25. The _____ interface in JDBC is used for establishing a connection to a database. Answer: **Connection**
26. _____ objects in JDBC are used to execute SQL queries. Answer: **Statement**
27. The _____ interface allows you to execute precompiled SQL statements in JDBC. Answer: **PreparedStatement**
28. _____ in JDBC allows you to retrieve and manipulate the results of a query. Answer: **ResultSet**
29. To close resources like connections and statements in JDBC, you should use the _____ block. Answer: **try-catch-finally**
30. _____ is the standard API for database access in Java. Answer: JDBC

Short Type

Java History, Architecture, and Features:

1. **Q:** Who developed the Java programming language? **A:** Java was developed by James Gosling and his team at Sun Microsystems.
2. **Q:** In which year was the first version of Java released? **A:** The first version of Java was released in 1995.
3. **Q:** What is the Java Virtual Machine (JVM)? **A:** JVM is a part of the Java Runtime Environment (JRE) responsible for executing Java bytecode.
4. **Q:** Name one key feature of Java's platform independence. **A:** Java programs can run on any platform with a compatible JVM.

Understanding the Semantic and Syntax Differences Between C++ and Java:

5. **Q:** What is a major difference between C++ and Java regarding memory management? **A:** In C++, programmers must explicitly manage memory, while Java uses automatic garbage collection.
6. **Q:** Can you use pointers in Java like you can in C++? **A:** No, Java does not have explicit pointers like C++.
7. **Q:** How does Java handle multiple inheritance differently from C++? **A:** Java uses interfaces to achieve multiple inheritance, while C++ supports it through multiple base classes.

Compiling and Executing a Java Program:

8. **Q:** What is the file extension for Java source code files? **A:** Java source code files have a **.java** extension.
9. **Q:** How do you compile a Java program from the command line? **A:** You can use the **javac** command followed by the Java source file's name.
10. **Q:** What command runs a compiled Java program? **A:** You can use the **java** command followed by the class name with the **main** method.

Variables, Constants, Keywords:

11. **Q:** Define a variable in Java. **A:** A variable is a named storage location in Java used to hold data of a particular type.
12. **Q:** What is the purpose of the **final** keyword in Java? **A:** The **final** keyword is used to make a variable, method, or class immutable or unchangeable.
13. **Q:** Name a keyword used for inheritance in Java. **A:** The **extends** keyword is used for inheritance in Java.

Data Types and Wrapper Classes:

14. **Q:** How many primitive data types are there in Java? **A:** There are 8 primitive data types in Java.
15. **Q:** What is autoboxing and unboxing in Java? **A:** Autoboxing is the automatic conversion of a primitive type to its corresponding wrapper class, and unboxing is the reverse.
16. **Q:** Which wrapper class represents integers? **A:** **Integer** is the wrapper class for integers.

Operators and Expressions:

17. **Q:** What is the result of **5 / 2** in Java? **A:** The result is **2** because it performs integer division.
18. **Q:** What operator is used for logical AND in Java? **A:** The **&&** operator is used for logical AND.
19. **Q:** Explain the purpose of the ternary operator **? :** in Java. **A:** It is used for conditional expressions, returning one of two values based on a boolean expression.

Comments and Basic Program Output:

20. **Q:** How do you write a single-line comment in Java? **A:** Single-line comments are written using **//**.
21. **Q:** What is the purpose of the **System.out.println()** method? **A:** It is used to print output to the console in Java.
22. **Q:** How can you write a multi-line comment in Java? **A:** Multi-line comments are enclosed between **/*** and ***/**.

Decision Making Constructs:

23. **Q:** What are the three types of decision-making constructs in Java? **A:** They are **if**, **switch**, and **ternary** (conditional operator).
24. **Q:** What is the purpose of the **break** statement in a **switch** statement? **A:** It is used to exit the **switch** statement after a case is matched.
25. **Q:** In a loop, what does the **continue** statement do? **A:** It skips the current iteration and continues with the next iteration of the loop.

Java Methods:

26. **Q:** What is the scope of a local variable in a Java method? **A:** Local variables are accessible only within the method where they are declared.
27. **Q:** What is method overloading? **A:** Method overloading is when a class has multiple methods with the same name but different parameters.
28. **Q:** What is the difference between a method parameter and an argument? **A:** A method parameter is a variable in the method's definition, while an argument is the actual value passed when calling the method.

Input through Keyboard:

29. **Q:** How can you read user input from the command line using Java's **Scanner** class? **A:** You create a **Scanner** object and use its methods, such as **nextLine()** or **nextInt()**, to read input.
30. **Q:** What is the purpose of the **BufferedReader** class when reading input? **A:** **BufferedReader** is used for efficient reading of characters, lines, or strings from an input stream.

Principles of Object-Oriented Programming:

1. **Q:** What are the four main principles of Object-Oriented Programming? **A:** The four main principles of OOP are encapsulation, inheritance, polymorphism, and abstraction.
2. **Q:** What is encapsulation in OOP? **A:** Encapsulation is the bundling of data (attributes) and methods (functions) that operate on the data into a single unit called a class.
3. **Q:** Explain the concept of inheritance. **A:** Inheritance is a mechanism in OOP where a new class (subclass or derived class) can inherit properties and behaviors from an existing class (superclass or base class).

Defining & Using Classes:

4. **Q:** What is a class in Java? **A:** A class in Java is a blueprint or template for creating objects.
5. **Q:** How do you define a class in Java? **A:** You define a class using the **class** keyword followed by the class name and its body enclosed in curly braces.
6. **Q:** What is an object in Java? **A:** An object is an instance of a class, representing a real-world entity.

Class Variables & Methods:

7. **Q:** What are class variables? **A:** Class variables, also known as static variables, belong to the class rather than a specific instance of the class. They are shared among all instances of the class.
8. **Q:** How do you define a class method (static method) in Java? **A:** You use the **static** keyword before the method declaration.
9. **Q:** Can you access a class method without creating an object of the class? **A:** Yes, class methods can be called using the class name without creating an object.

Objects & References:

10. **Q:** What is an object reference in Java? **A:** An object reference is a variable that holds the memory address of an object.
11. **Q:** How do you create an object in Java? **A:** You create an object using the **new** keyword followed by the class constructor.
12. **Q:** What is the difference between an object and an object reference? **A:** An object is the actual instance of a class, while an object reference is a variable that points to that instance.

Objects as Parameters:

13. **Q:** Can you pass objects as parameters to methods in Java? **A:** Yes, you can pass objects as method parameters in Java.
14. **Q:** What happens when you pass an object as a parameter to a method? **A:** You are passing a reference to the object, allowing the method to operate on the object's data.

Constructor & Constructor Overloading:

15. **Q:** What is a constructor in Java? **A:** A constructor is a special method used for initializing objects. It has the same name as the class.
16. **Q:** How does constructor overloading work? **A:** Constructor overloading is when a class has multiple constructors with different parameter lists.

17. **Q:** What is the purpose of the **this** keyword in a constructor? **A:** The **this** keyword is used to refer to the current instance of the class within a constructor.

Inheritance:

18. **Q:** What is inheritance in Java? **A:** Inheritance is a mechanism where one class inherits the properties and behaviors of another class.
19. **Q:** Explain the difference between **extends** and **implements** in inheritance. **A:** **extends** is used for class inheritance, while **implements** is used to implement interfaces.
20. **Q:** Name two types of inheritance. **A:** Two types of inheritance are single inheritance (one class inherits from one superclass) and multiple inheritance (one class inherits from multiple superclasses).

Interface & Polymorphism:

21. **Q:** What is an interface in Java? **A:** An interface defines a contract of methods that a class implementing the interface must provide.
22. **Q:** What is polymorphism in OOP? **A:** Polymorphism allows objects of different classes to be treated as objects of a common superclass.
23. **Q:** How does method overriding relate to polymorphism? **A:** Method overriding is a form of polymorphism where a subclass provides a specific implementation of a method defined in its superclass.

Package & Enumeration:

24. **Q:** What is a package in Java? **A:** A package is a way to organize classes into namespaces for better code organization and reuse.
25. **Q:** Can you create custom packages in Java? **A:** Yes, you can create custom packages by defining your package structure and placing your classes within it.
26. **Q:** What is an enumeration (enum) in Java? **A:** An enumeration is a data type that consists of a fixed set of named values, often used for representing a collection of constants.

Garbage Collection & Final Classes:

27. **Q:** What is garbage collection in Java? **A:** Garbage collection is the process of automatically reclaiming memory occupied by objects that are no longer reachable.
28. **Q:** What is a final class in Java? **A:** A final class is a class that cannot be extended (inherited) by other classes.

Up-Casting & Down-Casting:

29. **Q:** What is up-casting in Java? **A:** Up-casting is the casting of a subclass object to a superclass reference.
30. **Q:** What is down-casting in Java? **A:** Down-casting is the casting of a superclass reference to a subclass reference, which requires explicit type casting.

Arrays:

1. **Q:** What is an array in Java? **A:** An array is a data structure that stores a fixed-size sequence of elements of the same data type.
2. **Q:** How do you declare a one-dimensional array in Java? **A:** You declare a one-dimensional array using the following syntax: **dataType[] arrayName;**
3. **Q:** How do you create and initialize a one-dimensional array in Java? **A:** You can create and initialize a one-dimensional array like this: **int[] numbers = {1, 2, 3, 4, 5};**
4. **Q:** What is a two-dimensional (2D) array? **A:** A 2D array is an array of arrays, allowing you to store data in rows and columns.
5. **Q:** How do you declare and initialize a 2D array in Java? **A:** You can declare and initialize a 2D array like this: **int[][] matrix = {{1, 2}, {3, 4}};**
6. **Q:** What is a jagged array? **A:** A jagged array is an array of arrays where each sub-array can have a different length.

Array of Objects:

7. **Q:** Can you create an array of objects in Java? **A:** Yes, you can create an array of objects where each element is an instance of a class.
8. **Q:** How do you initialize an array of objects? **A:** You need to create instances of the class and assign them to the array elements.
9. **Q:** What is a dynamic array in Java? **A:** A dynamic array is an array that can change in size during runtime using classes like **ArrayList** or **LinkedList**.

Strings:

10. **Q:** What is the Java **String** class? **A:** The **String** class represents a sequence of characters and provides various methods for manipulating strings.

11. **Q:** How do you create a **String** object in Java? **A:** You can create a **String** object using the constructor or by simply assigning a string literal to a variable.
12. **Q:** Why are strings in Java immutable? **A:** Strings are immutable in Java to ensure their content cannot be changed after creation, which improves security and performance.
13. **Q:** How do you check if two strings are equal in Java? **A:** You can use the **equals()** method to compare the contents of two strings.
14. **Q:** What is the **StringBuilder** class used for? **A:** The **StringBuilder** class is used for creating mutable strings, allowing efficient concatenation and modification of strings.

I/O Package:

15. **Q:** What is the Java I/O package used for? **A:** The Java I/O package is used for input and output operations, including reading from and writing to files.
16. **Q:** What is a stream in Java I/O? **A:** A stream is a sequence of data elements that can be read from or written to, such as files, network connections, or memory buffers.
17. **Q:** What is the **File** class used for? **A:** The **File** class in Java is used to represent files and directories in the file system and provides methods for file manipulation.
18. **Q:** How do you read data from a file in Java? **A:** You can use classes like **FileInputStream** or **FileReader** to read data from a file.
19. **Q:** How do you write data to a file in Java? **A:** You can use classes like **FileOutputStream** or **FileWriter** to write data to a file.
20. **Q:** What is the purpose of the **PrintStream** and **PrintWriter** classes? **A:** These classes provide convenient methods for printing formatted data to output streams, including files.

Compressing and Uncompressing Files:

21. **Q:** What is file compression? **A:** File compression is the process of reducing the size of a file to save storage space or reduce transmission time.
22. **Q:** What classes are commonly used for file compression in Java? **A:** Java provides classes like **ZipInputStream** and **ZipOutputStream** for working with compressed files in ZIP format.
23. **Q:** How do you create a compressed ZIP file in Java? **A:** You can use **ZipOutputStream** to create a ZIP file and add entries to it.
24. **Q:** How do you extract data from a compressed ZIP file in Java? **A:** You can use **ZipInputStream** to read and extract data from a compressed ZIP file.

Exception Handling:

1. **Q:** What is an exception in Java? **A:** An exception is an event or error that occurs during the execution of a program and disrupts its normal flow.
2. **Q:** What is the purpose of exception handling? **A:** Exception handling allows a program to gracefully handle errors and exceptions, preventing crashes.
3. **Q:** What are some common built-in exceptions in Java? **A:** Common exceptions include **NullPointerException**, **ArrayIndexOutOfBoundsException**, and **IOException**.
4. **Q:** What is the difference between checked and unchecked exceptions? **A:** Checked exceptions must be declared in the method's signature or handled using **try-catch**, while unchecked exceptions (e.g., **RuntimeExceptions**) do not need this.
5. **Q:** How can you create custom exceptions in Java? **A:** You can create custom exceptions by extending the **Exception** class or its subclasses.
6. **Q:** What is the purpose of the **throw** keyword in Java? **A:** The **throw** keyword is used to manually throw an exception within a method.

Multi-Threading:

7. **Q:** What is multi-threading in Java? **A:** Multi-threading is the concurrent execution of multiple threads within a Java program.
8. **Q:** How do you create a thread in Java? **A:** You can create a thread by extending the **Thread** class or implementing the **Runnable** interface.
9. **Q:** What is the **Runnable** interface used for? **A:** The **Runnable** interface is used for creating threads by defining a **run()** method that contains the thread's code.
10. **Q:** How do you prioritize threads in Java? **A:** Threads can be assigned priorities using the **setPriority()** method, ranging from 1 (lowest) to 10 (highest).
11. **Q:** What is thread synchronization in Java? **A:** Thread synchronization is the process of controlling access to shared resources to prevent data corruption and race conditions.
12. **Q:** How can you suspend and resume threads in Java? **A:** You can use the **suspend()** and **resume()** methods, but they are deprecated. It's recommended to use **wait()** and **notify()**.

Java Networking:

13. **Q:** What is the **java.net** package used for? **A:** The **java.net** package provides classes for network communication, including client-server interactions.
14. **Q:** What is TCP/IP in networking? **A:** TCP/IP (Transmission Control Protocol/Internet Protocol) is a suite of communication protocols used for internet and network communication.
15. **Q:** How do you establish a client-server connection in Java using sockets? **A:** You can use the **Socket** class for client-side communication and the **ServerSocket** class for server-side communication.
16. **Q:** What is Datagram programming? **A:** Datagram programming uses the **DatagramSocket** and **DatagramPacket** classes to send and receive data in packets (datagrams).

Database Connectivity using JDBC:

17. **Q:** What is JDBC in Java? **A:** JDBC (Java Database Connectivity) is a Java API for connecting and interacting with databases.
18. **Q:** What are the steps to connect to a database using JDBC? **A:** The steps include loading the JDBC driver, establishing a connection, creating a statement, executing SQL queries, and handling results.
19. **Q:** What is the purpose of the JDBC **Connection** interface? **A:** The **Connection** interface represents a database connection and is used to create and manage database connections.
20. **Q:** How can you prevent SQL injection in JDBC? **A:** You should use prepared statements with placeholders instead of concatenating user input into SQL queries.
21. **Q:** What is connection pooling in JDBC? **A:** Connection pooling is a technique for reusing and efficiently managing database connections to improve performance.

LONG TYPE

Java History, Architecture, and Features:

1. How did Java evolve from its origins at Sun Microsystems to its current state?
2. Describe the key architectural components of the Java platform.
3. What are the primary features that distinguish Java from other programming languages?
4. Explain the concept of platform independence in Java.
5. Discuss the significance of the Java Virtual Machine (JVM) in the Java architecture.
6. How does Java handle memory management and garbage collection?
7. Describe the role of the Java Standard Library in Java development.

Understanding the Semantic and Syntax Differences Between C++ and Java:

8. Compare and contrast the syntax of C++ and Java with specific examples.
9. What are the major semantic differences between C++ and Java?
10. Explain how memory management differs in C++ and Java.
11. Discuss the role of pointers in C++ and how it differs from Java's approach.

Compiling and Executing a Java Program:

12. Walk through the steps involved in compiling and executing a Java program.
13. Explain the purpose of the **javac** and **java** commands in Java development.

Variables, Constants, Keywords:

14. Define variables in Java and explain their types.
15. How are constants defined in Java, and why are they useful?
16. Discuss the significance and usage of the following keywords: **super**, **this**, **final**, **abstract**, **static**, **extends**, **implements**, and **interface**.

Data Types:

17. List and explain the various data types available in Java.
18. What is the difference between primitive data types and reference data types in Java?

Wrapper Classes:

19. What are wrapper classes in Java, and why are they needed?
20. Provide examples of when and how to use wrapper classes.

Operators (Arithmetic, Logical, and Bitwise) and Expressions:

21. Explain the arithmetic operators available in Java.
22. Discuss the logical operators in Java and their use in boolean expressions.
23. What are bitwise operators, and when might they be used in Java?
24. Provide examples of complex expressions involving multiple operators.

Comments:

25. Describe the purpose of comments in Java code and different types of comments.
26. Explain when and why comments are essential in Java programming.

Doing Basic Program Output:

27. How can you display output in a Java program? Provide code examples.
28. Discuss the formatting options available for output in Java.

Decision-Making Constructs (Conditional Statements and Loops) and Nesting:

29. Describe the conditional statements available in Java (if, switch) and their usage.
30. Explain the different types of loops in Java (for, while, do-while).
31. Provide examples of nested conditional statements and loops.

Java Methods:

32. Define a method in Java and explain its components.
33. Discuss the scope of variables in Java methods.
34. How do you pass arguments to a method in Java?
35. Explain the concept of method overloading and provide examples.

Type Conversion and Type Checking:

36. What is type casting in Java, and when is it necessary?
37. Describe the difference between implicit and explicit type casting.
38. How does Java handle type checking and type safety?

Built-in Java Class Methods:

39. Discuss some commonly used built-in methods of the **String** class in Java.
40. Explain the significance of the **Math** class and its methods.

Input Through Keyboard Using Command Line Argument:

41. How can you accept input from the keyboard in a Java program using command line arguments?
42. Provide an example of a Java program that accepts command line arguments.

The Scanner Class and BufferedReader Class:

43. Describe the purpose of the **Scanner** class in Java and its basic usage.
44. Explain how the **BufferedReader** class is used for input in Java programs.
45. Compare and contrast the **Scanner** class and **BufferedReader** class for input operations.

Object-Oriented Programming Overview in Java:

46. What are the fundamental principles of Object-Oriented Programming (OOP)?
47. Explain the concept of classes and objects in OOP.
48. How are class variables and methods different from instance variables and methods?
49. Discuss the concept of object references in Java.
50. How can objects be passed as parameters to methods in Java?
51. What is the purpose of declaring a class as **final** in Java?
52. Explain the role of garbage collection in Java and its benefits.

Constructors: Types of Constructors, this Keyword, super Keyword:

53. What is a constructor, and why is it used in Java?
54. Differentiate between default constructors and parameterized constructors.
55. How does the **this** keyword work in Java, and why is it useful?
56. Explain the significance of the **super** keyword in constructors.

Method Overloading and Constructor Overloading:

57. Define method overloading and provide examples.
58. How is constructor overloading different from method overloading?
59. Provide examples of constructor overloading in Java.

Aggregation vs. Inheritance:

60. Explain the concepts of aggregation and inheritance in OOP.
61. Compare and contrast aggregation and inheritance.
62. When should you use aggregation, and when should you use inheritance in Java?

Inheritance: extends vs. implements, Types of Inheritance:

63. Describe the use of the **extends** keyword in Java and its relation to inheritance.
64. Explain how the **implements** keyword is used to implement interfaces.
65. Discuss the various types of inheritance in Java, including single, multiple, and multilevel inheritance.

Interface:

66. What is an interface in Java, and how does it differ from a class?
67. Explain how multiple inheritance is achieved using interfaces.
68. Provide examples of defining and implementing interfaces in Java.

Up-Casting, Down-Casting:

69. Define up-casting and down-casting in Java and explain their significance.
70. Discuss the potential risks and benefits of down-casting.

Auto-Boxing:

71. What is auto-boxing in Java, and when does it occur?
72. Explain how auto-boxing simplifies the use of primitive data types.

Enumerations:

73. Describe the purpose of enumerations in Java.
74. Provide examples of when and how to use enumerations.

Polymorphism: Method Overriding and Restrictions:

75. Explain the concept of polymorphism in Java.
76. Discuss method overriding and its role in achieving polymorphism.
77. What are the restrictions imposed on method overriding in Java?

Package: Pre-Defined Packages and Custom Packages:

78. Define a package in Java and explain its purpose.
79. Discuss the commonly used pre-defined packages in Java.
80. How can you create and use custom packages in your Java projects?

Arrays in Java: Creating & Using Arrays (1D, 2D, 3D, and Jagged Array):

81. Describe how to create and use one-dimensional arrays in Java.
82. Explain the concept of a two-dimensional array and provide examples.
83. Discuss the usage of three-dimensional arrays in Java.
84. What is a jagged array, and how is it different from a regular array?

Array of Object, Referencing Arrays Dynamically:

85. How can you create an array of objects in Java?
86. Explain dynamic referencing of arrays in Java and its benefits.

Strings and I/O: Java Strings:

87. What is the Java **String** class, and how is it used?
88. Describe the immutability of Java strings and its implications.
89. How can you manipulate strings in Java, such as concatenation and substring extraction?
90. Discuss the passing of strings to and from methods in Java.

StringBuffer Classes and StringBuilder Classes:

91. Explain the purpose of the **StringBuffer** and **StringBuilder** classes in Java.
92. Compare and contrast **StringBuffer** and **StringBuilder** with the **String** class.

IO Package: Understanding StreamsFile Class and Its Methods:

93. Describe the concept of streams in Java I/O.
94. What is the role of the **File** class in handling files in Java?
95. Provide examples of how to create, read, and write files using Java classes.

Byte and Character Streams, FileOutputStream, FileInputStream, FileWriter, FileReader, InputStreamReader, PrintStream, PrintWriter:

96. Differentiate between byte streams and character streams in Java I/O.
97. Explain how to use **FileOutputStream** and **FileInputStream** for binary file I/O.
98. Discuss the usage of **FileWriter** and **FileReader** for character-based file I/O.
99. Describe the roles of **InputStreamReader**, **PrintStream**, and **PrintWriter** in Java I/O.

Compressing and Uncompressing File:

100. How can you compress and uncompress files in Java, and why is it useful?

These questions should cover a wide range of Java topics, from basics to more advanced concepts. Use them to test your knowledge or as a study guide for Java programming.

Java Exception Handling:

1. Explain the concept of exception handling in Java. Why is it important in software development?
2. What is the difference between checked and unchecked exceptions in Java? Provide examples of each.
3. Describe the hierarchy of exception classes in Java. How does this hierarchy help in handling exceptions effectively?
4. When should you use the **try**, **catch**, and **finally** blocks in Java exception handling? Provide examples to illustrate their usage.
5. What is the purpose of the **throw** keyword in Java? How can it be used to create custom exceptions?
6. Explain the difference between the **throws** clause and the **throw** keyword in Java exception handling.
7. Can you give an example of a scenario where it is appropriate to create a custom exception class in Java?
8. Discuss the concept of multi-catch in Java and provide an example of its usage.
9. What are the advantages of using the **try-with-resources** statement for resource management in Java? Provide an example.
10. How do you handle exceptions that occur in a multi-threaded Java application?

Java Threading:

11. Describe the difference between a thread and a process in Java. What are the benefits of using threads?
12. Explain the role of the **Thread** class and the **Runnable** interface in Java threading. When and why would you use one over the other?
13. How can you create and start a new thread in Java? Provide code examples for both extending the **Thread** class and implementing the **Runnable** interface.
14. Discuss thread prioritization in Java. How does setting thread priorities affect their execution?
15. Explain the concept of thread synchronization in Java. Provide examples of situations where synchronization is necessary.
16. What is the purpose of the **synchronized** keyword in Java? How does it help in preventing race conditions?
17. How can threads communicate with each other in Java? Provide examples of inter-thread communication techniques.
18. Describe the risks and benefits of suspending and resuming threads in Java. When should you use these operations?
19. What are Java thread pools, and why are they useful in managing threads in a multi-threaded application?
20. Discuss the challenges and best practices of handling exceptions in multi-threaded Java applications.

Java Networking and Database Connectivity:

21. Provide an overview of the java.net package in Java. What are its key classes and their functionalities?
22. Explain the difference between TCP/IP and Datagram programming in Java networking. When would you choose one over the other?
23. How can you establish a client-server communication using sockets in Java? Provide a step-by-step explanation.
24. What are the key steps involved in creating a UDP-based server-client application in Java?
25. Describe the purpose of the Java Database Connectivity (JDBC) API. How does it facilitate database connectivity in Java applications?
26. Explain the steps required to connect to a database using JDBC. Provide code examples for database connection and query execution.
27. What is connection pooling in the context of JDBC? How does it improve database performance and resource management?
28. Discuss the various types of JDBC drivers available and their advantages and disadvantages.
29. How can you handle transactions in JDBC? Describe the use of **commit** and **rollback** operations.
30. Explain the concept of prepared statements in JDBC. What are the benefits of using prepared statements for database queries?