General:

Explain the difference between a class and an object in object-oriented programming (OOP).

Example Answer: "In OOP, a class is a blueprint or template for creating objects. It defines the properties (attributes) and behaviors (methods) that all objects of that class will have. An object, on the other hand, is an instance of a class. It represents a specific instantiation of the class, with its own set of values for the attributes and the ability to perform actions defined by the class methods."

What is the difference between a stack and a queue data structure? Provide examples of real-world applications for each.

Example Answer: "A stack is a Last-In-First-Out (LIFO) data structure, meaning that the last element added to the stack is the first one to be removed. It is commonly used for function calls and managing program execution flow. For example, the undo feature in text editors uses a stack to keep track of previous states.

On the other hand, a queue is a First-In-First-Out (FIFO) data structure, where the first element added to the queue is the first one to be removed. A real-world example of a queue is a line at a ticket counter, where people are served in the order they arrive."

Explain the concept of polymorphism in object-oriented programming and provide an example.

Example Answer: "Polymorphism is the ability of objects of different classes to be treated as objects of a common superclass. It allows methods to be called on objects of different classes without knowing their specific type at compile time. An example of polymorphism is a Shape superclass with subclasses such as Circle and Rectangle. Each subclass can override the draw() method to provide its own implementation, and objects of any subclass can be treated as Shape objects and called with the draw() method."

Explain the difference between a compile-time error and a runtime error in programming.

Example Answer: "A compile-time error occurs during the compilation phase of the program, where the compiler detects an error in the source code and prevents the program from being successfully compiled into executable code. Common examples include syntax errors, type errors, and missing or incorrect import statements.

However, a runtime error occurs during the program's execution, where an error is detected while it is running. This could be due to invalid input, division by zero, or accessing an array element out of bounds. Runtime errors typically cause the program to terminate abruptly or produce unexpected behavior."

Explain the time complexity of the binary search algorithm. How does it compare to linear search?

Example Answer: "The binary search algorithm has a time complexity of O(log n), where n is the number of elements in the sorted array. It achieves this by repeatedly dividing the search interval in half until the target element is found or the interval is empty. In contrast, the linear search algorithm has a time complexity of O(n), as it iterates through each element in the array until the target element is found. Binary search is significantly faster than linear search for large arrays, especially when the data is sorted."

What is a hash table and how does it work? Explain how collisions are handled in hash tables.

Example Answer: "A hash table is a data structure that stores key-value pairs, allowing for fast insertion, deletion, and lookup operations. It works by using a hash function to map keys to array indices, where the corresponding values are stored. Collisions occur when two different keys hash to the same array index. To handle collisions, various collision resolution techniques can be used, such as chaining or open addressing. Chaining involves storing multiple key-value pairs at each array index, typically using a linked list or another data structure. Open addressing involves finding an alternative location within the hash table to store the collided key-value pair."

Java specific:

What is the difference between an interface and an abstract class in Java? Provide examples of when you would use each.

Example Answer: "In Java, an interface defines a contract for classes to implement, specifying method signatures without method implementations. An abstract class, on the other hand, is a class that may contain both abstract and concrete methods, allowing for partial implementation. Interfaces are used to achieve multiple inheritance and define common behavior across unrelated classes, while abstract classes are used to provide a common base for related classes with shared functionality."

What is polymorphism in Java? How is it achieved, and why is it important?

Example Answer: Polymorphism in Java allows objects of different classes to be treated as objects of a common superclass. It is achieved through method overriding and method overloading. Method overriding enables a subclass to provide a specific implementation of a method defined in its superclass, while method overloading allows multiple methods with the same name but different parameter lists. Polymorphism is important because it promotes code reusability, flexibility, and extensibility.

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

Example Answer: Exception handling in Java involves handling errors and abnormal conditions that may occur during program execution. Checked exceptions are checked at compile time and must be either caught using a try-catch block or declared in the method signature using the "throws" keyword. Unchecked exceptions, also known as runtime exceptions, do not need to be declared or caught explicitly and typically indicate programming errors or unexpected conditions.

Explain the concept of method overloading and method overriding in Java. Provide examples of each.

Example Answer: "Method overloading refers to the ability to define multiple methods in a class with the same name but different parameters. The compiler determines which method to invoke based on the number and types of arguments provided. Method overriding, on the other hand, occurs when a subclass provides a specific implementation for a method that is already defined in its superclass. This allows the subclass to provide its own behavior while adhering to the superclass contract."

What is the difference between ArrayList and LinkedList in Java? When would you choose one over the other?

Example Answer: "ArrayList and LinkedList are both implementations of the List interface in Java, but they differ in their underlying data structures and performance characteristics. ArrayList uses an array to store elements, allowing for fast random access but slower insertion and deletion operations. LinkedList, on the other hand, uses a doubly linked list, providing fast insertion and deletion but slower random access. I would choose ArrayList when I need fast access to elements by index and LinkedList when I need efficient insertion and deletion in the middle of the list."

What is the difference between "==" and ".equals()" in Java? When would you use each of them for comparing objects?

Example Answer: In Java, the "==" operator checks for reference equality, meaning it compares whether two object references point to the same memory location. The ".equals()" method, on the other hand, checks for object equality, meaning it compares the contents or state of two objects. Use "==" to compare object references, and use ".equals()" to compare the contents of objects, especially when dealing with objects of user-defined classes.

Coding questions:

Reverse a String:

This question assesses the candidate's understanding of string manipulation and iteration.

Check if a String is a Palindrome:

Palindrome is a sequence that reads the same backward as forward (e.g., "racecar"). This question evaluates the candidate's ability to check for string equality and perform basic string manipulation.

Find the Fibonacci Series:

Fibonacci series is a sequence of numbers where each number is the sum of the two preceding ones (e.g., 0, 1, 1, 2, 3, 5, 8, ...). This question tests the candidate's understanding of loops and basic arithmetic operations.

Check if a Number is Prime:

This question evaluates the candidate's understanding of prime numbers and basic loop constructs.

Find the Factorial of a Number:

Factorial of a non-negative integer is the product of all positive integers less than or equal to that number (e.g., factorial of 5 = 5 \* 4 \* 3 \* 2 \* 1 = 120). This question assesses the candidate's understanding of loops and basic arithmetic operations.

Reverse an Array:

This question evaluates the candidate's ability to manipulate arrays and perform in-place modifications.

Find the Maximum Element in an Array:

This question assesses the candidate's understanding of array traversal and basic comparison operations.

Check if Two Arrays are Equal:

This question evaluates the candidate's understanding of array comparison and Java's built-in methods for array equality.

Count the Occurrences of an Element in an Array:

This question tests the candidate's ability to traverse an array and count occurrences of a specific element.

Remove Duplicates from an Array:

This question assesses the candidate's understanding of sets and their application for removing duplicates from an array while preserving order.