Quiz Performance Report

Name: Chaitu

Topic: OOP

Date: 2025-09-16 11:06:49

Final Score: 5 / 10

AI Performance Summary:

Could not generate a detailed analysis. Your final score was 5/10. Please try again later.

AI Recommendations & Resources:

No recommendations available.

Question Breakdown:

Question 1: Which of the following best describes the concept of polymorphism in Object-Oriented Programming?
Your Answer: The ability of an object to take on many forms, such as changing its data type during runtime. (Incorrect ■)
Correct Answer: The ability of a single method name to perform different actions depending on the object it is called upon.
Difficulty: medium

Question 2: Which of the following best describes polymorphism in the context of Object-Oriented Programming?
Your Answer: The process of hiding internal details of an object and exposing only essential information to the outside world. (Incorrect ■)
Correct Answer: The ability of an object to take on many forms, allowing objects of different classes to be treated as objects of a common type.
Difficulty: medium

Question 3: Which of the following is a fundamental concept of Object-Oriented Programming (OOP)?
Your Answer: Encapsulation, which bundles data and methods that operate on that data within a class. (Correct ■)
Correct Answer: Encapsulation, which bundles data and methods that operate on that data within a class.
Difficulty: easy

Question 4: Which of the following best describes the concept of polymorphism in Object-Oriented Programming?
Your Answer: The process of hiding internal data and methods within a class, exposing only necessary interfaces. (Incorrect ■)
Correct Answer: The ability of objects of different classes to respond to the same method call in their own specific way.
Difficulty: medium

Question 5: Which of the following best describes the concept of polymorphism in object-oriented programming?
Your Answer: The ability of objects of different classes to respond to the same method call in their own specific way. (Correct ■)
Correct Answer: The ability of objects of different classes to respond to the same method call in their own specific way.
Difficulty: medium

Question 6: Consider a scenario where you need to implement the Strategy pattern to handle different payment methods (Credit Card, PayPal, Bank Transfer). Each payment method has a specific validation process before processing the payment. However, some payment methods share common validation steps (e.g., checking for sufficient funds). Which design pattern, in conjunction with the Strategy pattern, would be MOST effective for reusing and managing these common validation steps, while maintaining flexibility and avoiding code duplication?
Your Answer: Decorator: To dynamically add validation responsibilities to individual payment method strategies without altering their structure. (Incorrect ■)
Correct Answer: Template Method: To define a skeleton algorithm in a base class, allowing subclasses (each payment method) to override specific steps while maintaining a common structure.
Difficulty: hard

Question 7: Consider a scenario where you need to implement the Strategy pattern to dynamically change the sorting algorithm used on a collection of objects. You have three concrete strategy classes: QuickSort, MergeSort, and HeapSort. Which design choice would BEST handle the potential for adding new sorting algorithms in the future, while minimizing code changes to existing components and ensuring flexibility and open/closed principle adherence?
Your Answer: Employ a single, overarching interface for all sorting algorithms. New algorithms implement this interface without requiring changes to the client code that uses the strategy pattern, ensuring extensibility without modification of existing classes. (Correct ■)
Correct Answer: Employ a single, overarching interface for all sorting algorithms. New algorithms implement this interface without requiring changes to the client code that uses the strategy pattern, ensuring extensibility without modification of existing classes.
Difficulty: hard

Question 8: Consider a scenario where you're designing a system for managing different types of vehicles (cars, trucks, motorcycles). Which OOP principle is BEST suited to ensure that each vehicle type can respond to a 'startEngine()' method differently, without requiring modification to existing code?
Your Answer: Inheritance: Creating new vehicle types by extending existing ones, inheriting common attributes and methods. (Incorrect ■)
Correct Answer: Polymorphism: Allowing objects of different classes to respond to the same method call in their own specific way.
Difficulty: medium

Question 9: Consider a class hierarchy designed for a game: `Creature` is a base class with subclasses `Human`, `Elf`, and `Orc`. Each subclass inherits attributes like `health` and `strength` from `Creature`, but also has unique attributes. `Human` has `intelligence`, `Elf` has `agility`, and `Orc` has `ferocity`. Which of the following best describes the OOP principle being demonstrated?
Your Answer: Inheritance: Subclasses inherit common attributes and methods from the `Creature` class, avoiding redundant code. (Correct ■)
Correct Answer: Inheritance: Subclasses inherit common attributes and methods from the `Creature` class, avoiding redundant code.
Difficulty: medium

Question 10: Which of the following best describes polymorphism in Object-Oriented Programming?
Your Answer: The ability of an object to take on many forms, allowing objects of different classes to be treated as objects of a common type. (Correct ■)
Correct Answer: The ability of an object to take on many forms, allowing objects of different classes to be treated as objects of a common type.
Difficulty: medium