OBJECT-ORIENTED PROGRAMMING THROUGH JAVA(24CS204)

Academic Year :2025 – 26, II YR – I SEM

Module -1 Question Bank

1. Develop a cricket player statistics tracker analysis.

a) Describe a base class called "CricketPlayer" with the attributes name, age, and country, as well as the methods getters and setters. Create "Batsman" and "Bowler" classes with additional attributes and methods tailored to their respective player types. Batsmen should have attributes such as runs scored and centuries, as well as a method for calculating strike rate. Bowlers should have attributes such as wickets taken and bowling average, as well as a method for calculating economy rate.

b) Analyze the performance of the player by considering the following statistical metrics like: Batting Average, Strike Rate,Runs Scored, Centuries and Half-Centuries, Bowling Average, Economy Rate, Wickets Taken, Fielding Status.

c) Develop a user management system for an online platform. The system should allow users to register, log in, and manage their profiles. Implement inheritance to represent different types of users, such as Customers and Administrators. Additionally, the system should include features like password encryption, profile updates, and role-based access control.

2. Design and implement a Java program that fulfills the following requirements:

a) Discuss a base class called User with attributes like username, email, and password. Implement methods for user registration, login, and profile management, including updating the email and password.

b) Apply a subclass called Customer that inherits from the User class. Add additional attributes like name, address, and contact information specific to customers. Customers should be able to view and update their profile information, such as the name and address. Implement appropriate methods to handle these operations. Implement another subclass called Admin that also inherits from the User class. Admins have additional privileges compared to customers, such as the ability to manage user accounts. Admins should be able to view and update customer profiles, as well as create or delete customer accounts. Implement methods to support these admin-specific functionalities.

c) Create a secure password encryption mechanism, such as using hashing algorithms (e.g., bcrypt or SHA-256), to store and compare user passwords securely. Ensure that the passwords are not stored in plain text format.Implement role-based access control to restrict certain functionalities based on user roles. For example, customers should not have access to admin-specific operations, such as user management. Implement appropriate methods to enforce access control rules.

3. Develop Java program that analyzes text and provides various functionalities for text processing and analysis using strings.

   a) Describe a Java program that accepts a line of text as input from the user. Implement a method to count the total number of characters, including spaces. Create another method to count the number of words in the text using string splitting.Also, implement a method to display the frequency of each unique word in the text.

   b) Analyze the program to find and display the longest word present in the input text. Add functionality to check whether the given text is a palindrome (ignoring spaces and case). Implement logic to determine and print the most frequently occurring character.These tasks require analytical thinking using string and collection manipulation.

   c) Create a method that converts the given text into title case format.Write a method to remove all punctuation and special characters from the input.Implement a simple substitution cipher that encrypts text by shifting characters forward.This question emphasizes creativity and problem-solving using character operations.

4. Design a ticket reservation system for a theater that includes exception handling mechanisms. The system should allow users to select and reserve seats for a particular show. Implement the following functionalities with appropriate exception handling:

   a) Discuss a method to check whether the tickets are available or not. Handle the exception when the show is not found or when an error occurs while accessing the database and method to reserve seats for a customer. Handle exceptions related to invalid seat selections (e.g., already reserved or out of range). Display appropriate error messages to the user.

   b) Apply a payment processing mechanism to complete the reservation. Handle exceptions related to payment failures or errors during the transaction. Display an error message if the payment cannot be processed. Develop a method to cancel a reservation. Handle exceptions when the reservation does not exist or when an error occurs during the cancellation process.

   c) Evaluate a mechanism to log any exceptions that occur during the ticket reservation process. Ensure that appropriate error messages are logged, including relevant details such as timestamps and the nature of the exception Validate user inputs throughout the system to prevent invalid inputs or potential security vulnerabilities. Implement exception handling to handle validation errors and display helpful error messages to the user.

5.

a) Describe a base class Shape with a method calculateArea() to compute area. Extend it into two subclasses: Rectangle and Circle with specific area formulas. Override calculateArea() in each subclass to implement appropriate logic. This demonstrates method overriding and inheritance in geometry applications.

b) Apply objects of Rectangle and Circle and assign them to Shape references. Use polymorphism to call calculateArea() at runtime for each shape. Print the area for both objects using overridden methods in derived classes. This illustrates dynamic binding and runtime polymorphism in action.

c) Create a Calculator class with a divide() method accepting two numbers. Use a try-catch block to handle division by zero using ArithmeticException. Display a meaningful message when an invalid division occurs. This ensures the program remains stable and user-friendly during errors.

6.

a) Describe a Java method to find indices of two elements in an array that sum to a target value. Use a single loop approach to return the index pair without reusing elements. Ensure the method returns the correct result for inputs like [2, 7, 11, 15] and target = 9. This task involves array traversal, conditional logic, and efficient data lookup.

b) Apply a method to find the element that appears only once in an array of duplicates. Use bitwise XOR logic to achieve linear time and constant space complexity. Next, write a method to add two integers without using + or - operators. This encourages creative use of bitwise operations like AND, OR, and XOR.

c) Develop a class NumberOperations with overloaded methods to process integer and double arrays. Implement methods to calculate maximum, minimum, and average values usingmethodoverloading.Use input arrays like int[] numbers1 = {10, 5, 20, 15, 30} and double[] numbers2={...}.This task assesses understanding of object-oriented concepts and method reuse.

7.

a) Discuss a method to read a text file containing employee records in a line-by-line format.Use BufferedReader to read the file and understand the structure of the data. Demonstrate how to use StringTokenizer to split each line based on commas. Extract individual values like EmployeeID, Name, Department, and Salary.

b) Analyzing StringTokenizer, tokenize each record and separate employee details. Store the extracted values in a suitable data structure like an Employee object. Add all employee objects into a list for easy traversal and future processing. Ensure that the tokenizer handles variable-length names and whitespace correctly.

c) Develop the list of employees and format the output for clarity. Print EmployeeID, Name, Department, and Salary in a readable structure. Implement functionality to allow further processing like filtering or sorting. Design the code to be modular and reusable for different input files.

8.

a) Discuss a Java class that calculates the factorial of a given number using a recursive approach. Then, manually convert the code into bytecode representation using a tool like javap or an online bytecode viewer. Analyze the bytecode and explain the bytecode instructions and their corresponding operations.

b) Apply the factorial calculation class to use an iterative approach instead of recursion. Recompile the class and analyze the updated bytecode to observe any differences compared to the previous version. Explain the bytecode instructions and any changes in their opcodes or operands.

c) Create a subclass that extends the factorial calculation class and overrides one of its methods. Recompile the subclass and examine the bytecode to understand how the Java compiler handles inheritance and method overriding in bytecode representation.

9.

a) Describe an abstract class Animal with abstract methods: eat(), makeSound(), and move().Create subclasses Mammal, Bird, and Reptile to implement the specific behaviors. Each subclass should override the methods to reflect its unique characteristics. This structure demonstrates abstraction and polymorphism in animal behavior modeling.

b) Analyze one object each for Mammal, Bird, and Reptile classes. Invoke overridden methods to display unique sound, eating, and movement behavior.

Use polymorphism to treat all objects as type Animal while showing subclass output. This proves dynamic method dispatch and real-time method resolution.

c) Develop packages: customers, products, and orders for e-commerce organization. Define one class per package with relevant fields and access modifiers. Use methods to interact between packages for adding and processing orders. This approach improves code maintainability, clarity, and modular development.

10. Class hierarchy for media types (books, CDs, DVDs) with appropriate attributes and methods.Utilize method overloading to implement the necessary operations for each media type.Organize your classes into packages to maintain a structured codebase.

a) Describe a Java application for a library management system. The system needs to handle different types of media, including books, CDs, and DVDs. Each type of media has specific attributes such as title, author, duration, and so on.

b) Apply a set of classes and packages to represent the media types and their attributes. Utilize method overloading to handle different operations on each type of media, such as adding, updating, and deleting records.

c) Evaluate your classes into appropriate packages to maintain a well-structured codebase. Consider using packages to separate media types (books, CDs, DVDs) and related operations.