VIT-AP UNIVERSITY	Final Assessment Test – Winter Sem(2024-25) Fresher -May 2025	
	Maximum Marks: 100	Duration: 180 Mins
Course Code: CSE2005	Course Title: Object Oriented Programming using JAVA	
Set No: 6	Exam Type : Closed Book	
Date: 23/05/2025	Slot: C1	School: SCOPE
		Session: FN

Keeping mobile phone/smart watch, even in 'off' position is treated as exam malpractice

1. **"fx series" - non-Programmable calculator are permitted: NO**
2. **Reference tables permitted: YES / NO if Yes, please specify: NO**
3. **Assume data when ever necessary and clearly sate the assumption.**

PART – A: Answer any 10 Questions, Each Question Carries 10 Marks (10×10=100 Marks)

1. Write a Java program that defines a class named Student with the following attributes: studentId (a String), name (a String), age (an int), and marks (an integer array to store the marks of five subjects). The class should include a parameterized constructor that uses this keyword to initialize all the attributes. Implement a method displayDetails() to print the student's ID, name, age, and marks, and another method calculateAverage() to compute and return the average of the marks. Use arrays to manage the marks efficiently and apply appropriate access specifiers to ensure encapsulation of data. In the main() method, create an object of the Student class, accept input from the user, call the methods to display student details and calculate the average, and display whether the student has passed or failed based on the condition that the average mark should be greater than or equal to 50 to pass. **(10M)**

2. Write a Java program to create a class named BankAccount that demonstrates the concept of encapsulation by declaring its attributes accountNumber (a String), accountHolderName (a String), and balance (a double) as private, and providing public methods to access and modify these data members. Include a static variable bankName that is shared among all account objects. The class should have a constructor that uses the this keyword to initialize the account details. Implement public methods such as deposit(double amount) to increase the account balance and withdraw(double amount) to deduct money from the account, while ensuring that withdrawals are not allowed if the balance is insufficient by using appropriate control statements. Also, implement a method displayBalance() to print the account holder's details and the current balance. As a condition, allow withdrawal only if the remaining balance after withdrawal does not go below a minimum required balance of ₹1000. In the main() method, create at least two objects of the BankAccount class, perform deposit and withdrawal operations on each, and display the output. **(10M)**

3. In the Vehicle Management System, create a Vehicle class with attributes like vehicleId, vehicleName, fuelCapacity, and maxSpeed, and a startEngine() method. Then, create derived classes Car, Truck, and Motorcycle with an attribute time, distance and override the startEngine() method to display specific messages "started" and calculateTime(int distance) The stopEngine() method should print the time taken to travel the distance. Use the super keyword to call the base class constructor. Write a Java program to create an object for car, truck and motorcycle, read the input and display the time taken for each vehicle to travel a particular distance. (Time = Distance / Speed) **(10M)**

4. In a Library Management System, write a Java program that defines an abstract class named Book with attributes such as bookTitle, author, isbn, and price, along with an abstract method

displayDetails(). Create two subclasses, Ebook and PrintedBook, that extend the Book and provide concrete implementations of the displayDetails() method to display their speci details. Use constructors to initialize the attributes using the super keyword where appropriate. As a condition, ensure that the price of an Ebook must be less than ₹500, and the price of a PrintedBook must be greater than or equal to ₹500; otherwise, display a message indicating that the price is invalid. In the main program, create objects of both Ebook and PrintedBook, invoke their displayDetails() methods, and demonstrate abstraction and method overriding in Java. **(10M)**

5. Write a Java program to manage various types of media files such as audio, video, and image files. To improve organization and maintainability, group the related classes into a package named media. This package should contain three classes: AudioFile, VideoFile, and ImageFile. Each class must include attributes for the media type and file size, along with methods to display the type of media and its corresponding size. Implement a Main class outside the package to demonstrate the usage of the media package by creating instances of AudioFile, VideoFile, and ImageFile and invoking their methods to display the media details. As a condition, ensure that the file size for any media file must be greater than 0 MB; otherwise, display an error message indicating that the file size is invalid. This program should demonstrate the use of Java packages, class design, and conditional checks for better modularity and code management. **(10M)**

6. Write a Java program to build a simple payment gateway system that supports various payment methods such as CreditCard, PayPal, and BankTransfer. Define a PaymentMethod interface with a method named processPayment(double amount). Implement this interface in three classes: CreditCard, PayPal, and BankTransfer, where each class provides its own version of the processPayment() method to simulate how that specific payment method would process a transaction. In your implementation, ensure that the method prints a confirmation message along with the amount paid. As a condition, validate that the payment amount must be greater than ₹100; otherwise, display an error message stating that the payment amount is too low and cannot be processed. In the main class, create instances of each payment method and demonstrate? **(10M)**

7. Write a Java program to build a simple ATM application where users can withdraw money from their account. The application should begin with a predefined account balance and prompt the user to enter the withdrawal amount. The system must handle two specific exceptions: ArithmeticException, which should be thrown when the withdrawal amount exceeds the available balance, and InputMismatchException, which should occur when the user enters a non-numeric value instead of a valid amount. Use try-catch blocks to catch and handle these exceptions and display appropriate error messages such as "Insufficient Balance" for ArithmeticException and "Invalid Input! Please enter a numeric value" for InputMismatchException. As a condition, withdrawals should only be allowed if the amount is a multiple of 100; otherwise, the system should display an error message stating "Withdrawal amount must be in multiples of 100." After a successful transaction, print the withdrawn amount and the remaining account balance. **(10M)**

8. Write a Java program to develop a simple login system that validates a user's username based on specific rules. The program should prompt the user to enter a username and check if it meets the following conditions: the username must be at least 6 characters long and must not contain any spaces. If the username is shorter than 6 characters, the program should throw a user-defined exception named InvalidUsernameException with a message indicating that the username is too short. If the username contains one or more spaces, the program should throw a predefined IllegalArgumentException with an appropriate error message. Use try-catch

blocks to handle both exceptions and display clear, user-friendly error messages in each case. If the username passes all validations, print a success message such as "Login successful." **(10M)**

9. Write a Java program to develop a simple student management system where you store a list of student names and their corresponding roll numbers. Use non-generic collections, specifically the ArrayList class to store the names of the students and the HashMap class to associate roll numbers with student names. Populate the ArrayList with a few student names and map each roll number to a name using the HashMap. To display the student names, use an Iterator to traverse the ArrayList. For the HashMap, use a for-each loop to iterate over the entry set and display each roll number along with the associated student name. As a condition, ensure that duplicate roll numbers are not allowed in the HashMap; if a duplicate roll number is detected while inserting, display a message saying "Duplicate roll number not allowed." This program demonstrates the use of non-generic collections in Java and how to efficiently iterate over them using appropriate techniques. **(10M)**

10. Write a Java program to design a generic class named AverageCalculator<T> that calculates the average of an array of values. The class should accept an array of type T[] through its constructor and include a method calculateAverage() that returns the average as a double. Inside the method, convert the elements to double where necessary to perform arithmetic operations. In the main() method, demonstrate the usage of the AverageCalculator class with arrays of different numeric types such as Integer and Double. As a condition, ensure the array is not empty before performing the calculation; if it is, display a message like "Array is empty. Cannot calculate average." This program demonstrates the use of generic classes and shows how generics improve code reusability and flexibility across different data types. **(10M)**

11. Write a Java program to develop a multi-threaded banking system where multiple users can perform operations like checking their balance or withdrawing money concurrently. Each user should be represented by a separate thread, which can be created either by extending the Thread class or implementing the Runnable interface. The program should demonstrate synchronization techniques to ensure thread safety when users are accessing shared resources like their account balance. Also maintain a minimum balance of 1000. Proper exception handling should also be in place to handle situations like insufficient funds during withdrawal attempts. **(10M)**

12. Write a Java program to simulate a multi-threaded ticket booking system for a cinema, where multiple users (represented by separate threads) attempt to book tickets concurrently. The cinema has a fixed number of seats available for each show, and each user can either book one or more tickets at a time. The program should ensure that only one thread can access and modify the number of available seats at a time, preventing race conditions and ensuring that no more tickets are booked than are available. Use synchronization mechanisms (such as synchronized methods or blocks) to manage access to the shared resource—the number of available seats. Additionally, the system should display an error message if a user attempts to book more tickets than are available, with a message such as "Not enough tickets available." Also calculate the total ticket price (each ticket cost is 200). After all threads have completed their operations, the program should display the final number of available tickets, demonstrating how synchronization prevents issues like race conditions in a concurrent environment. **(10M)**