Course Code	Course Title	L	T	P	C
BACSE102	Problem Solving using Java	0	0	4	2
Pre-requisite	Nil	**Syllabus Version**			
		1			

Course Objectives

To introduce the fundamental concepts of object-oriented programming using the Java language, including data types, control structures, and basic programming constructs
To enable students to design and implement Java programs utilizing classes, objects, inheritance, polymorphism, and interfaces to build modular and reusable code
To equip students with a comprehensive understanding of Java's exception handling and multithreading capabilities. Java's File Object, input/output (I/O) streams and to equip students with the skills to leverage generic programming and the Java Collection Framework with Databases

Course Outcomes

Develop basic Java programs utilizing fundamental programming constructs, object-oriented principles such as classes, objects, inheritance, and polymorphism
Implement robust and efficient Java applications by applying exception handling techniques, managing concurrent execution using threads, and performing file input/output operations.
Utilize the Java Collection Framework, including Lists and Maps, along with generic programming concepts to effectively manage and manipulate data within Java applications.

Indicative Experiments

1. a. Calculating Interest for Financing Options, Bobby is a financial analyst working for a construction company. He is tasked with evaluating two financing options for funding a new construction project. Each option involves different principal amounts and annual interest rates. Write a program for him that takes the principal amounts and interest rates for both financing options as input and calculates the respective interests for the financial options. Note: Multiply the principal amount by the annual interest rate and then divide by 100 to find the total interest paid b. Three-Digit Number Analysis: Sum of Digits and Parity Check. Alex is teaching a class on number properties and wants to create an exercise for his students. He needs a program that takes a three-digit number as	2 hours

input, calculates the sum of its digits, and determines whether the sum is even or odd. Write a program to assist Alex in this task. c. Generating Prime Numbers Within a Specified Range. Arun is tasked with creating a program that prints prime numbers within a given range. The program should take two integers, start and end, as input, and output the prime numbers between these two values (inclusive). Help Arun to complete the task using a 'for' loop.	
2. a.Calculating Total Distance from Merged Daily Step Counts You are given the number of steps a person takes on two consecutive days. Your task is to merge the step counts from both days and calculate the total distance the person covers, assuming that each step covers a fixed distance of 1 unit. Write a program that takes the step counts for the two days as input, merges them, calculates the total distance covered and outputs the result. b. Identifying and Locating Prime Numbers in a 2D Grid Yashna is creating a program to analyse a 2D grid, identifying prime numbers and printing their coordinates. The input consists of the grid's dimensions and elements. The program's output displays the coordinates of each prime number found in the grid, aiding in the exploration of numerical patterns. Can you assist Yashna in creating the program?	2 hours
3. a. Designing a Rectangle Class to Calculate Area and Perimeter. Arun wants to design a program using a class that calculates and prints both the area and perimeter of a rectangle. Define a class named Rectangle to encapsulate the properties and behaviour related to a rectangle. Include two integer variables length and breadth to represent the dimensions of the rectangle. Calculate and print the area and perimeter of the rectangle. Help Arun to design the program b.Developing an ArrayConcatenator Class for Merging Arrays.Help Roshni to complete the program. Input Format The first line of input consists of an integer N, representing the number of elements of the first Roshni is tasked with developing a program for concatenating two arrays provided by the user. To accomplish this, she wants to create a class named ArrayConcatenator. The class includes a constructor to concatenate the elements of the input arrays. She wants to print the elements of the resulting array. The second line consists of N space-separated integers representing the first array elements. The third line consists of an integer M, representing the number of elements of the second array. The fourth line consists of M space-separated integers, representing the second array elements. Output Format:	2 hours

The output prints the concatenated array of elements separated by space. Refer to the sample output for the formatting specifications. Constraints: In this scenario, the test cases fall under the following constraints 1 ≤ N, M ≤ 10 1 ≤ array elements ≤ 100	
4. a. Calculating Employee Bonuses Using Hierarchical Inheritance ☐ Shasha, an HR manager in a multinational corporation, requires a program to compute bonuses for developers and designers using hierarchical inheritance. Three classes: Employee with a base salary attribute; Developer and Designer extending Employee, each with a bonus percentage, work hours, and methods to calculate and display the final income. Shasha inputs the base salary and work hours for a developer and a designer, and the program outputs their respective final incomes. ☐ Note: For developers, the bonus percentage is 10%, for designers, it is 5% ☐ Formula: Final income = Base salary + (base salary * bonus percentage * work hours / 100) ☐ b.Designing a Fantasy Game Character System.Jessica is tasked with designing a fantasy game character system. The system includes an abstract class named GameCharacter with two abstract methods: attack() and defend().Two subclasses, Warrior and Wizard, extend the GameCharacter class.The program prompts the player to choose a character class (1. Warrior, 2. Wizard) and input their character's strength or magic power. The dynamic calculations involve tripling the strength (strength * 3) for the Warrior's attack and doubling the magic power(power * 2) for the Wizard's attack. Jessica needs your help in completing the program. Help her finish it.	2 hours
5. a. Designing a Washing Machine Control System. Alex and Bob are designing a control system for household appliances, and one of the appliances is a washing machine. You want to create a program to help them that models the washing machine as a motor and calculates its electricity consumption based on its capacity. Define an interface named Motor with the following methods: void run() double consume(double capacity) Create a class called WashingMachine that implements the Motor interface. In the WashingMachine class: Implement the run() method to print "Washing machine is running."	2 hours

Implement a consume() method to print "Washing machine is consuming electricity." Implement the consume(double capacity) method to calculate the electricity consumption (in kWh) of the washing machine based on its capacity. The formula for electricity consumption is (capacity * 0.05). b. Unique Character Extraction.Ashok needs to implement a program using the java.lang package to process a given string. The goal is to create a program that extracts and displays a new string containing only the unique characters from the provided input string. Ashok should efficiently use the java.lang.StringBuilder class and display the output. Help him to complete the program.	
6. a. Validating Student Grade Input with Multi-Catch Exception Handling. Sampad wants to implement a program that takes input for a student's name and grade, validates the input, and then displays the grade for the given student. The grade should be an integer value. The program should validate the grade using the validateGrade() method. The method should throw an IllegalArgumentException if the grade is less than 0 or greater than 100. If the input is invalid due to a non-integer grade, catch the NumberFormatException and print the built-in exception message. If the input is invalid due to an out-of-range grade, catch the IllegalArgumentException and print the built-in exception message. Catch the exceptions using the multi-catch block. Assist Sampad to implement this program. b. Implementing Custom Exceptions for Time Input Validation.Alice is tasked with creating a program that validates user input for time values (hours, minutes, and seconds). The program should check whether the input values are within the valid range. If any value falls outside the valid range, a custom exception, InvalidHourException, InvalidMinuteException, or InvalidSecondException, should be thrown, indicating which part of the time (hours, minutes, or seconds) is invalid. If no exceptions occur, print "Correct Time - " followed by the time in the format "hours:minutes:seconds."	2 hours
7. Synchronizing Producer and Consumer Threads with a Shared Message Buffer.A shared resource, a message buffer, needs to be managed by two threads: a Producer and a Consumer. The Producer thread will generate messages and place them into the buffer. The Consumer thread will retrieve and process messages from the buffer. To ensure proper synchronization and avoid issues like the Consumer trying to read from	2 hours

an empty buffer or the Producer overwriting a message before it's consumed, you need to implement inter-thread communication using wait(), notify(), and synchronized blocks. Create a MessageBuffer class that holds a single message. The Producer thread will put a message into this buffer, and the Consumer thread will take a message from it. Implement the necessary synchronization mechanisms so that: The Consumer waits if the buffer is empty.The Producer waits if the buffer is full (i.e., already contains a message that hasn't been consumed).The Producer notifies the Consumer when a new message is available.The Consumer notifies the Producer after consuming a message, indicating the buffer is now empty.Write a Producer class that generates a sequence of messages (e.g., "Message 1", "Message 2", etc.) and puts them into the MessageBuffer.Write a Consumer class that retrieves and prints the messages from the MessageBuffer.Create a Main class to instantiate the MessageBuffer, a Producer thread, and a Consumer thread, and then start both threads.	
8. Calculating Final Prices with Tax: File Input and Output.Ella is managing her expenses and wants to calculate the final prices after applying a 10% tax to her purchases. She has a list of item prices that she wants to process. Create a file named prices.txt and write the entered item prices to this file. Then, the program should read from prices.txt, apply a 10% tax to each item, write the taxed amounts to a file named tax.txt, and display the taxed amounts. Input Format: The first line of input consists of an integer N, representing the number of items Ella wants to process.The second line consists of N space-separated double values, representing the price of an item.Output Format:The output displays a double value, representing the taxed amounts, each with two decimal places, separated by a space.Refer to the sample output for formatting specifications.Constraints: this scenario, the test cases fall under the following constraints:1 ≤ N ≤ 10.	2 hours
9. Basic Sentiment Analysis: Classifying Sentences from File Input. ☐Mr. Styles is working on a sentiment analysis program to understand the sentiments conveyed in various sentences. he needs your assistance in developing a program that analyzes the sentiment of a given sentence and classifies it as positive, negative, or neutral. ☐positive keywords = happy, good, excellent, positive. ☐ negative keywords = sad, bad, terrible, negative. ☐Anything else is Neutral. ☐Create a file named input.txt, the input is written into the file input.txt. The program then reads the sentence from input.txt and sentiment analysis is performed based on predefined positive and negative keywords. The classified sentiment (positive, negative, or neutral) is written to a new file named output.txt and displayed.	2 hours

10. Converting Speed Units (km/h to m/s) with File Input and Output: Nick is developing a program to convert speed units from kilometers per hour (km/h) to meters per second (m/s). However, he needs assistance in creating a structured and user-friendly program. Create a file named data.txt to enter the speed in km/h as given in the input. The program reads the speed from data.txt and then converts the speed from km/h to m/s using the formula. The converted speed is written to a new file named converted.txt and then displayed.	2 hours
11. Implementing Persistent Bank Account Transactions using Serialization. Sam is managing his bank account and wants to perform various transactions using a program. He wants to deposit and withdraw funds and then save the account details using serialization. Upon restarting the program, he wants to be able to load the serialized data and continue with his banking transactions. Write a program to assist Sam in managing his bank account, create a BankAccount class, and Serialize the BankAccount object to a file named bankAccount.ser after the transactions. Deserialize the BankAccount object from the file when the program restarts. Display the final balance after the transactions in a formatted manner with two decimal places.	2 hours
12. Serializing and Deserializing Savings Data to Determine Savings Category. Alice is managing her expenses and wants to create a simple program to track her savings she wants to categorise her savings as "Poor savings," "Good savings," or "High savings" using a serialization mechanism. Design a program to create an object of the SavingsData class with the provided salary and savings data. Serialize the created object to a file named savings.ser. Deserialize the SavingsData object from the savings.ser file. Display the category determined by the program after deserialization. Note: "Poor savings", if the savings percentage is between 1% (inclusive) and 10%. "Good savings", if the savings percentage is between 10% (inclusive) and 20%. "High savings", if the savings percentage is greater than or equal to 20%. "Invalid input", if none of the above conditions are met.	2 hours
13.	2 hours

Implementing a Generic Record Holder for Academic Data. A university needs a system to manage different types of academic records, such as student grades and course enrollments. To create a flexible and type-safe system, they want to use generic classes.Design a generic class named RecordHolder that can hold any type of academic record. This class should have the following functionalities: A private instance variable to store the record. A constructor that takes an object of the specific record type and initializes the instance variable. A method named getRecord() that returns the stored record. A method named displayRecordInfo() that prints some relevant information about the record. The implementation of this method will depend on the specific type of record being held. Create two concrete classes to represent different types of academic records: GradeRecord: This class should have private instance variables for studentName (String) and grade (Double). It should have a constructor to initialize these variables and a displayRecordInfo() method that prints: "Grade for [studentName]: [grade]". EnrollmentRecord: This class should have private instance variables for courseName (String) and studentId (Integer). It should have a constructor to initialize these variables and a displayRecordInfo() method that prints: "Student [studentId] is enrolled in [courseName]". In your Main class, demonstrate the use of the RecordHolder generic class with both GradeRecord and EnrollmentRecord objects. Create instances of RecordHolder to hold a GradeRecord and an EnrollmentRecord, and then call the getRecord() and displayRecordInfo() methods on these RecordHolder instances.	
14. Student Registration System using ArrayList in Java. Raman is a computer science teacher who is responsible for registering students for his programming class. He wants to streamline the registration process using a simple program. The program should allow him to input the names of students and later retrieve a student's name based on the index entered. Raman has decided to use an ArrayList to store the names.	2 hours

15. Calculating Median Temperature from City Data using HashMap. As a data analyst for a weather forecasting agency, you have a HashMap storing temperature data for various cities. Each city's entry consists of temperature readings. Your task is to find and report the median temperature, providing a more representative measure of the overall temperature patterns over a specific period. Note: To calculate the median, sort the numbers in ascending order. If the count is odd, the median is the middle number. If the count is even, the median is the average of the two middle numbers.	2 hours
16. You are tasked with developing a simple Employee Management System using Java and JDBC. The system should allow users to perform the following operations on employee records stored in a MySQL database Develop a menu-driven program (EmployeeApp.java) that allows users to: 1. Add Employee 2. View Employees 3. Update Employee Salary 4. Delete Employee 5. Exit	2 hours
Total Laboratory Hours:	**60 hours**

Text Book(s)

Marc Loy, Patrick Niemeyer and Daniel Leuck, , "**Learning Java, "**, O Reilly Media, 5th Edition, **2020**

Herbert Schildt and Dr. Danny Coward, " **Java: The Complete Reference**", McGraw Hill, 13th Edition, **2024**

Reference Books

Kathy Sierra, Bert Bates and Trisha Gee,, " **Head First Java**", O Reilly Media, 3rd Edition, **2022**

Dhruti Shah, "**100+ Solutions in Java: A Hands-On Introduction to Programming in Java, "**, BPB Publications, 1st Edition, **2020**

Mode of Evaluation : Lab Continuous Assessment, Lab Final Assessment	
Recommended by Board of Studies :	21-05-2025
Approved by Academic Council : No. 78	12-06-2025