



2024-25



جب کوئی قوم فن اور علم سے
عساری ہو جاتی ہے تو وہ
عسرت کو دعوت دیتی ہے
اور جب عسرت آتی ہے تو
وہ ہزاروں حبرائیم کو جسم دیتی
ہے۔

*"When a nation
becomes devoid of art
and learning, it
invites poverty and
when poverty comes
it brings in its wake
thousands of crimes."*

*-Sir Syed Ahmad
Khan*

Laboratory Course-II

Course Code- CAMS2P01



MCA II Semester Lab Manual

DEPARTMENT OF COMPUTER SCIENCE

ALIGARH MUSLIM UNIVERSITY, ALIGARH

2024-2025

Credits

The following lab manual up-gradation committee updated the lab manual:

- ☐ *Prof.Arman Rasool Faridi (Chairperson)*
- ☐ *Prof. Mohammad UbaidUllah Bokhari*
- ☐ *Prof. Aasim Zafar*
- ☐ *Prof. Suhel Mustajab*
- ☐ *Dr. Faisal Anwar*
- ☐ *Dr. Mohammad Sajid*
- ☐ *Dr. Mohammad Nadeem*
- ☐ *Dr. Faraz Masood*

The following committee member originally design the lab manual:

- ☐ *Prof. Mohammad Ubaidullah Bokhari*
- ☐ *Dr. Arman Rasool Faridi*
- ☐ *Dr. Faisal Anwer*
- ☐ *Prof. AasimZafar (Converner)*

Design & Compilation:

- ☐ *Dr. Faraz Masood*

Revised Edition: Jan, 2025

*Department of Computer Science, A.M.U.,
Aligarh (U.P.) India*

COURSE TITLE: Laboratory Course-II
CREDIT: 04
CONTINUOUS ASSESSMENT: 40

COURSE CODE: CAMS2P01
PERIODS PER WEEK: 06
EXAMS: 60

COURSE DESCRIPTION

This course is designed to help students in learning JAVA using object-oriented paradigm. Approach in this course is to take JAVA as a language that is used as a primary tool in many different areas of programming work.

COURSE CONTENT

This course is designed to provide the students the opportunity to learn the differences between C++ and JAVA programming and to develop, debug, and execute JAVA programs.

OBJECTIVES

This course is designed to help students in:

- ☐ Gaining knowledge about basic Java language syntax and semantics to write Java programs and use concepts such as variables, conditional and iterative execution methods etc.
- ☐ Understanding the fundamentals of object-oriented programming in Java, including defining classes, objects, invoking methods, etc., and exception handling mechanisms.
- ☐ Learning the use of arrays, access protection, wrapper classes etc.
- ☐ Understanding the principles of inheritance, packages, and interfaces.
- ☐ Familiarizing with the important topics and principles of software development.
- ☐ Learning the ability to write a computer program to solve specified problems.

- ❑ Understanding to use the Java SDK environment to create, debug and run simple Java programs.

OUTCOMES

After completing this course, the students would be able to:

- ❑ Understand the fundamental features of an object-oriented language like JAVA: object classes and interfaces, exceptions, and libraries of object collections.
- ❑ Understand how to implement, compile, test, and run JAVA programs comprising more than one class to address a particular software problem.
- ❑ Understand the concept of multithreading, multitasking, and multiprogramming.
- ❑ Understand the use of members of classes found in the JAVA API (such as the Math class, Wrapper classes).
- ❑ Understand the implementation of the keyboard/mouse events.
- ❑ Understand the concept of object-oriented programming, inheritance, constructors, interfaces, and packages.
- ❑ Understand the concept of Exception Handling, Files and Streams.
- ❑ How to take the statement of a business problem and, from this, determine suitable logic for solving the problem; then, be able to code that logic as a program written in JAVA.

RULES AND REGULATIONS

Students are required to strictly adhere to the following rules.

- ❑ The students must complete the weekly activities/assignments well in time (i.e., within the same week) that need to be checked and signed by the

concerned teachers in the lab in the immediate succeeding week. Failing which activities/assignments for that week will be treated as incomplete.

- ❑ The students must maintain the Lab File of their completed activities/assignments in the prescribed format (**Appendix-1**).
- ❑ At least **TEN (10)** such timely completed and duly signed weekly activities/assignments are compulsory, failing which students will not be allowed to appear in the final Lab Examination.
- ❑ The students need to submit the following three deliverables for each exercise duly signed by the Teacher:
 - ❖ Coding
 - ❖ Input /Output
- ❑ Late submissions would not be accepted after the due date.
- ❑ Cooperate, collaborate, and explore for the best individual learning outcomes, but copying is strictly prohibited.
- ❑ The Continuous Lab assessment will be based on two sessionals, each carrying 30 marks.
- ❑ Marks distribution for each sessional:
 - ❖ 20 Marks: For a duly signed lab report by the respective lab teacher.
 - ❖ 5 Marks: For solving a lab question/program on the day of the sessional.
 - ❖ 5 Marks: For the viva conducted during the sessional.
- ❑ This distribution ensures a balanced evaluation of students' practical work, problem-solving skills, and conceptual understanding.

APPENDIX-1

Template for the Index of Lab File

WEEK NO.	PROBLEMS WITH DESCRIPTION		PAGE NO.	SIGNATURE OF THE TEACHER WITH DATE
1	1#			
	2#			
	3#			
2	1#			
	2#			
	3#			
3	1#			
	2#			
	3#			

Note: The students should use Header and Footer mentioning their roll no. & name in footer and page no in header.

WEEK #1

OBJECTIVES

- ☐ To help the students understand the importance of programming in Object Oriented environment using JAVA
- ☐ To help the students in understanding the basic structure of JAVA Program
- ☐ To help students get familiar with the JAVA programming environment and IDE.
- ☐ To learn the students in writing, debugging and executing JAVA programs.
- ☐ How to read input from the keyboard?
- ☐ To learn the constant and variables.

OUTCOMES

After completing this, the students would be able to:

- ☐ Write, debug, and execute simple JAVA programs.
- ☐ Use the constant and variables in JAVA.

PROBLEMS

- 1# Write a java program to print the corresponding address of a student.
- 2# Write a java program for finding the sum, difference, product, quotient, minimum and maximum of any two integers.
- 3# Write a Java program to convert the given temperature in Fahrenheit to Celsius using the following conversion formula $C = (F - 32) / 1.8$ and display the value in a tabular form.
- 4# Write a java program 'MyNumber.java' that performs following operations on a variable 'num' of type double:

- i) Finds the round value of 'num' and stores the result in a variable numRound of type double.
- ii) Finds the ceil value of 'num' and stores the result in a variable numCeil of type double.
- iii) Finds the floor value of 'num' and stores the result in a variable numFloor of type double.

Cast 'num' to type int and stores the result in a variable numInteger of type int. Display output of numRound, numCeil, numFloor and numInteger on screen.

Note: Test your program with following value of num

num=56.764

num=36.432

5# Write Java program to show uses of all Math class methods.

WEEK #2

OBJECTIVE

- ❑ To learn using JAVA programming coding: data types, variable, constants, operators, Control Statement (*if, switch, loops*)
- ❑ To learn break, continue statements, ternary operator, bit-wise operators, user-defined data types in JAVA, order of evaluation of different operators in Java.
- ❑ To learn the controls statements and loops.

OUTCOMES

After completing this,

- ❑ The students would be able to use different control statements and loops available in JAVA.

PROBLEMS

- 1# Write a java program to prints the count of odd and even no's entered.
- 2# Write a java program to print the squares and cubes for the numbers 1 to 5.
- 3# Write a java program that computes the sum of the reciprocals:
$$\frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \dots \dots \dots + \frac{1}{10}$$
- 4# Write Java program to compute the sum of the 2+4+6+-----N Terms.
- 5# Shown below is a Floyd's triangle:

```
1
2 3
4 5 6
7 8 9 10
```

11 12 13 14
15 16 17 18 19

- i) Write a program to print the above triangle.
- ii) Modify the program to produce the following form of Floyd's triangle.

```
1
0 1
1 0 1
0 1 0 1
1 0 1 0 1
0 1 0 1 0 1
```

WEEK #3

OBJECTIVE

- ❑ To learn using JAVA programming coding: data types, variable, constants, operators, Control Statement (*if, switch, loops*)
- ❑ To learn break, continue statements, ternary operator, bit-wise operators, user-defined data types in JAVA, order of evaluation of different operators in Java.
- ❑ To learn the controls statements and loops.

OUTCOMES

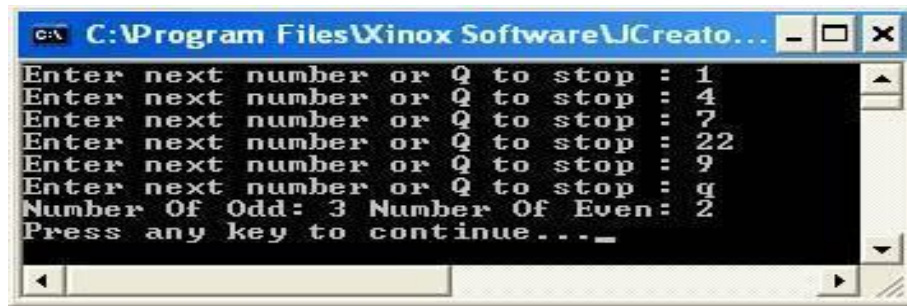
After completing this,

- ❑ The students would be able to use different control statements and loops available in JAVA.

PROBLEMS

- 1# Write a program in the sequence 0, 1, 1, 2, 3, 5, 8, 13, 21 is called *Fibonacci numbers*. Write a program using a *do...while* loop to calculate and print the first m *Fibonacci numbers*.
- 2# Write a program to accept three digits (i.e. 0 - 9) and print all its possible combinations. (*For example if the three digits are 1, 2, 3 than all possible combinations are: 123, 132, 213, 231, 312, 321*)
- 3# Write a Java Program which prompts the user to enter 4 numbers. The program will then computes and display their sum and their product.

- 4# Write a Java program which reads a 4-digit number and prints the digits on separate lines. (Each digit is printed on one line). Output of your program must be in the following format:



```
C:\Program Files\Xinox Software\JCreato...
Enter next number or Q to stop : 1
Enter next number or Q to stop : 4
Enter next number or Q to stop : 7
Enter next number or Q to stop : 22
Enter next number or Q to stop : 9
Enter next number or Q to stop : q
Number Of Odd: 3 Number Of Even: 2
Press any key to continue..._
```

- 5# The intersection method computes the intersection of two rectangles- that is, the rectangle that is formed by two overlapping rectangles: You call this method as follows: **Rectangle r3 =r1.intersection (r2);**

Write a program that constructs two rectangle objects, prints them, and then prints their intersection. What happens when the rectangles do not overlap?

WEEK #4

OBJECTIVES

- ❑ To learn the object oriented features (classification, encapsulation, inheritance, polymorphism, abstraction) and Java features like secure, platform independent, portable, threading.
- ❑ To learn the concept of class like how to declare a class, how to define methods inside the class, how to access the properties of base class into derived class, Code reusability.
- ❑ To learn the use of arrays, access protection, wrapper classes.
- ❑ To learn the use of this keyword, use of toString () method.

OUTCOME

After completing this, the students would be able to:

- ❑ employ various types of selection constructs in a JAVA program
- ❑ demonstrate the hierarchy of JAVA classes to provide a solution to a given set of requirements.

PROBLEMS

1# Write Java program involving two classes: *OddAndEven* & *TestOddAndEven*. *OddAndEven* has the following:

- Instance variables `countOfOdd` and `countOfEven` both of type `int`.
- A method `addNumber` that takes a number as parameter and increment `countOfOdd`, if the number is odd, else increment `countOfEven`.
- A method `toString` that returns a string in the form: "Number of Odd: x, Number of Even: y", where x and y are the values of the instance variables.

The *TestOddAndEven* class first creates *OddAndEven* object, then in a loop, read a number and use it to call the *addNumber* method until the user enters Q. Finally, it prints the count of odd and even numbers entered.

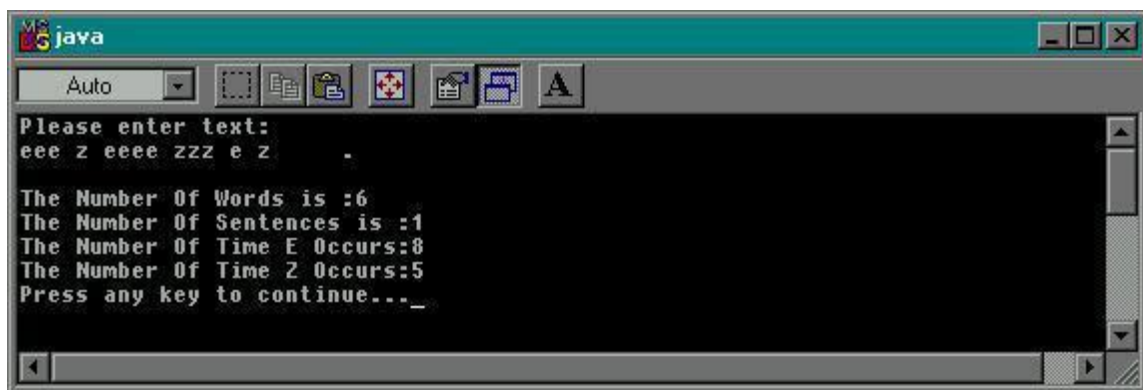
2# Design a class *Circle* and implement the following methods:

- Define a circle method to compute its area
- Define a circle method to compute its perimeter
- Define a method that takes a given point represented by a pair of numbers and checks whether or not the point is inside the circle.

The circle class needs to have instance variables to store the radius of the circle, and the x and y coordinates of the center. Add main program to test the class *Circle* repeatedly, until user enters negative value for the radius of the circle.

3# Write a program in Java that reads in text and prints as output the following:

- The number of words in the text
- The number of sentences in the text
- The number of times the letter "e" occurs in the text
- The number of times the letter "z" occurs in the text



(Note: Use *StringTokenizer* class)

4# A sales person is paid commission based on the sales he makes as shown by the following table:

Sales	Commission
Under ₹500	2% of Sales
Between ₹500 to ₹5000	5% of Sales
₹5000 and Above	8% of Sales

Write a class *Commission* which has an instance variable *sale*, an appropriate constructor, and a method *commission ()* that returns the commission.

Now write a demo class to test the *Commission* class by reading a sale from the user, using it to create a *Commission* object after validating that the value is not negative. Finally, call the *commission ()* method to get and print the commission. If the sales are negative, your demo should print the message "Invalid Input".

WEEK #5

OBJECTIVES

- ☐ To learn the object oriented features (classification, encapsulation, inheritance, polymorphism, abstraction) and Java features like secure, platform independent, portable, threading.
- ☐ To learn the concept of class like how to declare a class, how to define methods inside the class, how to access the properties of base class into derived class, Code reusability.
- ☐ To learn the use of arrays, access protection, wrapper classes.
- ☐ To learn the use of this keyword, use of toString () method.

OUTCOME

After completing this, the students would be able to:

- ☐ employ various types of selection constructs in a JAVA program
- ☐ demonstrate the hierarchy of JAVA classes to provide a solution to a given set of requirements.

PROBLEMS

You are tasked with designing and implementing a Java program that simulates an online library catalog system. The system should have classes representing books, users, and the catalog itself. Users should be able to borrow and return books, and the catalog should maintain information about available books and their status. Ensure that proper encapsulation, inheritance, and polymorphism principles are applied.

Requirements:

1. *Book Class:*

- Design a class named `Book` to represent individual books in the library.
- Include attributes such as book ID, title, author, genre, and availability status.
- Implement appropriate methods, including getters and setters.

2. User Class:

- Design a class named `User` to represent library users.
- Include attributes such as user ID, name, and a list to store borrowed books.
- Implement methods to borrow and return books.

3. Catalog Class:

- Design a class named `Catalog` to represent the library catalog.
- Use collections (e.g., ArrayList) to store a collection of books.
- Implement methods to add books to the catalog, display available books, and update book availability.

4. Encapsulation:

- Ensure that proper encapsulation is applied to protect the internal state of the classes. Use access modifiers appropriately.

5. Inheritance and Polymorphism:

- Utilize inheritance to create a more specific type of book (e.g., FictionBook, NonFictionBook) that extends the base `Book` class.
- Demonstrate polymorphism through a method like `displayDetails()` that can provide different information based on the book type.

6. Main Program:

- In the main program, instantiate the classes, add books to the catalog, create users, and simulate the borrowing and returning of books.
- Display the catalog's status after each transaction.

Note:

- Pay attention to proper error handling and validation, such as checking if a book is available before borrowing.
- Consider using meaningful names for classes, variables, and methods.

WEEK #6

OBJECTIVES

- ☐ To learn the concept Interface, how to implement it,
- ☐ To learn the use of member functions and how we access them, using interfaces for code reusing,
- ☐ To learn converting between class and interface types, using interfaces for callbacks; Polymorphism, Inheritance

OUTCOMES

After completing this, the students would be able to:

- ☐ handle interfaces, converting between class and interface types, callbacks, Polymorphism, Inheritance, Inheriting instance fields and methods.
- ☐ access the member functions of a class.

PROBLEMS

- 1# Write a program that reads in a sentence from the user and prints it out with each word reversed, but with the words and punctuation in the original order:



- 2# Write a program where interface can be used to support multiple inheritances. Develop a standalone Java program for this.
- 3# Write a program that reads in three strings and sorts them lexicographically.
Hint: Enter strings: Charlie Able Banker

Output: Able Banker Charlie

- 4#** Implement the classes for the shapes using an interface for the common methods, rather than inheritance from the super class, while still Shape as a base class.

WEEK #7

OBJECTIVES

- ☐ Assess file handling, exception handling, and design practices in Java.
- ☐ Evaluate the use of try-catch blocks, FileInputStream, BufferedReader, and FileOutputStream.
- ☐ Ensure the candidate can implement a complete and effective solution.

After completing this, the students would be able to:

OUTCOME

- ☐ Candidates will demonstrate proficiency in file handling, exception handling, and design practices in Java.
- ☐ Candidates will effectively utilize try-catch blocks, FileInputStream, BufferedReader, and FileOutputStream.
- ☐ Candidates will implement complete and functional solutions following proper coding standards.

PROBLEMS

1# Instructions:

You are tasked with implementing a Java program that reads numeric values from a text file, calculates the average of these values, and outputs the result to another text file. The program should handle file-related issues and invalid data using exception handling.

Requirements:

1. File Input:

- Design a Java class named `DataProcessor` with a method `readValuesFromFile(String filePath)` that takes a file path as a parameter and reads numeric values from the specified text file.

- Implement exception handling to catch and handle `FileNotFoundException` and `IOException` appropriately.

2. Data Validation:

- Within the `DataProcessor` class, implement a method `validateData(List<Double> values)` to validate the read numeric values. Ensure that only valid numeric data is considered for calculating the average.

- Handle invalid data by throwing a custom exception (`InvalidDataException`) for non-numeric values.

3. Average Calculation:

- Create a method `calculateAverage(List<Double> validValues)` within the `DataProcessor` class to calculate the average of the valid numeric values.

4. File Output:

- Implement a method `writeResultToFile(double average, String outputPath)` within the `DataProcessor` class to write the calculated average to another text file specified by the `outputPath`.

- Handle `IOException` during the file writing process.

5. Main Program:

- In the main program, instantiate the `DataProcessor` class, call the methods in sequence to read values, validate data, calculate the average, and write the result to another file.

6. Exception Handling:

- Define a custom exception class `InvalidDataException` that extends `Exception`. Use this exception for handling invalid data.

.

WEEK #8

OBJECTIVES

- ☐ Assess understanding of exception handling in Java.
- ☐ Evaluate the ability to develop a program that counts exceptions in a Java program.
- ☐ Test the ability to report the total number of exceptions identified.

After completing this, the students would be able to:

OUTCOMES

- ☐ Candidates will demonstrate a clear understanding of exception handling concepts in Java.
- ☐ Candidates will develop a functional program capable of counting exceptions in a Java program.
- ☐ Candidates will accurately report the total number of exceptions identified in the program..

PROBLEMS

1# Create a Java program named “ExceptionCounter.java”.

- Implement a method named “countExceptions” that takes the path to a Java program file as input and returns the total number of exceptions present in that program.
- The program should read the specified Java file, analyze its content, and count occurrences of exception-related keywords or patterns.
- Use appropriate exception handling mechanisms to handle scenarios such as file not found or syntax errors in the Java program.
- Print the total number of exceptions found on the screen.

2# Create a Java program named “NullPointerExceptionHandler.java”.

- Implement three different versions of the program using the provided blocks of code (a, b, and c) to identify, handle, or prevent Null Pointer Exceptions.
- Each version should include appropriate comments explaining the chosen approach and the logic behind it.
- Demonstrate the functionality of each version by specifying scenarios in which Null Pointer Exceptions may occur.
- Print the details of the Null Pointer Exception on the screen in each version.

.

WEEK #9

OBJECTIVES

- ❑ Assess understanding of logging and reporting in Java, specifically for unchecked exceptions using logging frameworks.
- ❑ Evaluate the ability to handle severe runtime exceptions in Java.
- ❑ Test the ability to implement graceful termination to prevent further damage or data corruption.

OUTCOMES

After completing this, the students would be able to:

- ❑ Candidates will demonstrate an understanding of logging and reporting unchecked exceptions using logging frameworks in Java.
- ❑ Candidates will effectively handle severe runtime exceptions in Java.
- ❑ Candidates will implement graceful termination techniques to prevent further damage or data corruption.

PROBLEMS

- 1# Create a Java program that simulates a scenario where an unchecked exception might occur. This could involve operations such as reading from a file, parsing data, or any other operation that could potentially throw an unchecked exception.
- 2# Implement exception handling using a try-catch block for the specific type of exception that might be thrown.

- 3#** Integrate a logging framework, either Log4j or SLF4J, into your program to log the details of the exception.
- 4#** Ensure that your program handles the exception gracefully by logging the exception details and providing a meaningful message.
- 5#** Create a Java program that simulates a scenario where a severe runtime exception might occur. This could involve critical operations or conditions that, if not handled properly, could lead to potential issues.
- 6#** Implement exception handling using a try-catch block for the specific type of exception that might be considered severe.
- 7#** Integrate a logging framework (e.g., Log4j or SLF4J) into your program to log the details of the severe runtime exception. Use “System.exit()” to gracefully terminate the application with an error code after logging the exception.

.

WEEK #10

OBJECTIVES

- ☐ Assess mastery of handling checked exceptions in Java.
- ☐ Evaluate the ability to manage database connections effectively in Java.
- ☐ Test proficiency in utilizing advanced techniques for exception handling and resource management.

OUTCOMES

After completing this, the students would be able to:

- ☐ Candidates will demonstrate mastery in handling checked exceptions in Java.
- ☐ Candidates will effectively manage database connections in Java.
- ☐ Candidates will apply advanced techniques for exception handling and resource management.

PROBLEMS

1# Extend the provided Java program to include advanced techniques for handling checked exceptions related to opening a database connection.

2# Implement the following enhancements:

- Use the “try-with-resources” statement to manage the “Connection” object. Ensure proper closure of the connection, even if an exception occurs.
- Add a “finally” block to log a message indicating the successful closure of the database connection, regardless of whether an exception occurred or not.

- Introduce a custom exception class for representing database connection-related issues. Throw this custom exception instead of the generic “SQLException” when encountering problems.

3# Demonstrate the enhanced program by intentionally providing incorrect database connection details. Include a scenario where the custom exception is thrown and caught, and ensure that the connection is properly closed.

WEEK #11

OBJECTIVES

- ❑ Assess advanced understanding of handling unchecked exceptions in Java.
- ❑ Evaluate the ability to handle exceptions during date parsing in Java.
- ❑ Test proficiency in employing advanced techniques for exception handling and providing meaningful feedback.

OUTCOMES

After completing this, the students would be able to:

- ❑ Candidates will demonstrate an advanced understanding of handling unchecked exceptions in Java.
- ❑ Candidates will effectively handle exceptions that occur during date parsing in Java.
- ❑ Candidates will apply advanced techniques for exception handling and provide meaningful feedback.

PROBLEMS

1# Enhance the provided Java program to include advanced techniques for handling unchecked exceptions related to parsing dates.

2# Implement the following enhancements:

- Use the “try-with-resources” statement to manage the “DateFormat” object.

Ensure proper closure of the formatter, even if an exception occurs.

- Introduce a custom exception class for representing date parsing-related issues. Throw this custom exception instead of the generic “ParseException” when encountering problems.

- Add a “finally” block to log a message indicating the successful closure of the “DateFormat” object, regardless of whether an exception occurred or not.

3# Demonstrate the enhanced program by intentionally providing an incorrectly formatted date string. Include a scenario where the custom exception is thrown and caught, and ensure that the “DateFormat” object is properly closed.

WEEK #12

OBJECTIVES

- ☐ Evaluate advanced understanding of handling checked exceptions in Java.
- ☐ Assess the ability to handle network connections and exceptions effectively in Java.
- ☐ Test proficiency in utilizing advanced exception handling techniques, providing meaningful feedback, and managing resources.

OUTCOMES

After completing this, the students would be able to:

- ☐ Candidates will demonstrate an advanced understanding of handling checked exceptions in Java.
- ☐ Candidates will effectively manage network connections and handle related exceptions in Java.
- ☐ Candidates will apply advanced exception handling techniques, provide meaningful feedback, and manage resources effectively..

PROBLEMS

1#. Extend the provided Java program to include advanced techniques for handling checked exceptions related to establishing a network connection.

2# Implement the following enhancements:

- Use the “try-with-resources” statement to manage the “URLConnection” object. Ensure proper closure of the connection, even if an exception occurs.

- Introduce a custom exception class for representing network connection-related issues. Throw this custom exception instead of the generic “IOException” when encountering problems.

- Add a “finally” block to log a message indicating the successful closure of the “URLConnection” object, regardless of whether an exception occurred or not.

3# Demonstrate the enhanced program by intentionally providing an incorrect URL. Include a scenario where the custom exception is thrown and caught, and ensure that the “URLConnection” object is properly closed.

WEEK #13

OBJECTIVES

- ❑ Assess advanced understanding of handling checked exceptions when dealing with file input in Java.
- ❑ Evaluate the ability to apply advanced techniques for exception handling and provide meaningful feedback during file input operations.
- ❑ Test proficiency in managing resources effectively during file input in Java.

OUTCOMES

After completing this, the students would be able to:

- ❑ Assess advanced understanding of handling checked exceptions when dealing with file input in Java.
- ❑ Evaluate the ability to apply advanced techniques for exception handling and provide meaningful feedback during file input operations.
- ❑ Test proficiency in managing resources effectively during file input in Java.

PROBLEMS

1# Extend the provided Java program to include advanced techniques for handling checked exceptions related to file input.

2# Implement the following enhancements:

- Use the “try-with-resources” statement to manage the “FileReader” object. Ensure proper closure of the file reader, even if an exception occurs.

- Introduce a custom exception class for representing file input-related issues. Throw this custom exception instead of the generic “FileNotFoundException” when encountering problems.

- Add a “finally” block to log a message indicating the successful closure of the “FileReader” object, regardless of whether an exception occurred or not.

3# Demonstrate the enhanced program by intentionally providing the name of a nonexistent file. Include a scenario where the custom exception is thrown and caught, and ensure that the “FileReader” object is properly closed.

WEEK #14

OBJECTIVES

- ☐ Assess understanding of the Particle Swarm Optimization (PSO) algorithm.
- ☐ Evaluate the ability to implement the PSO algorithm in Java.
- ☐ Test the ability to apply PSO for solving a simple example problem in Java.

OUTCOMES

After completing this, the students would be able to:

- ☐ Candidates will demonstrate a clear understanding of the Particle Swarm Optimization (PSO) algorithm.
- ☐ Candidates will effectively implement the PSO algorithm in Java.
- ☐ Candidates will apply the PSO algorithm to solve a simple example problem in Java.

PROBLEMS

Instructions:

1. Create a Java program named “PSOImplementation.java” that implements the Particle Swarm Optimization algorithm.
2. Choose a simple example problem for optimization (e.g., finding the minimum of a mathematical function).
3. Implement the PSO algorithm components, including particle initialization, fitness evaluation, velocity and position updates, and termination conditions.

4. Allow customization of key parameters such as the number of particles, iterations, and any algorithm-specific parameters.
5. Print the final optimized solution and the corresponding fitness value.

Tasks:

1. Choose a simple example problem (e.g., a mathematical function) for optimization.
2. Implement the PSO algorithm components within the “psoAlgorithm” method.
3. Define the fitness function specific to the chosen example problem within the “fitnessFunction” method.
4. Allow customization of key parameters in the main method.
5. Demonstrate the functionality by running the “PSOImplementation” program and printing the final optimized solution and its fitness value.