Semester:	**IV**	**Course Code:**	**21UCSL404**
Course Title:	**Object Oriented Programming Laboratory**	**Division:**	**A & B**
Academic Year:	**2022-23**	**Course Instructors:**	**GN and GMS**

List of Lab-works

LW-1) Write a program in Java to simulate a calculator of set-operations like Union, Intersection, Power Set, etc. with choices of operation specified from the menu. The user will choose the operation and enter the two sets as input and the output will be produced as per the operation chosen.

LW-2) "`data.csv`" is a file with academic data of set of students of a division, having, `name`, `USN, IA-1, IA-2, IA-3, sum_of_best_two_IAs, CTA`, and `CIE`, separated by commas. Write a program in Java to read the file and find the toppers based on one of the parameters of choice. Give menu-based options to the user.

LW-3) You are given information about a student like `Name`, and `USN`, along with his/her `IA-1, IA-2, IA-3, CTA`, `SEE, Credits` for 8 subjects of for each semester for the total of 8 semesters. Write a program in Java to read such information and do the computations necessary to find the grades of the student along with `SGPA` and `CGPA`. Display the output like a typical SDMCET grade card.

LW-4) You are given the coordinates (clockwise order) of the line segments that form two different polygons. Write a Java program that determines the type of shape formed by each polygons and determine their intersection relationship (intersect with each other or away from each other or one lies inside the other).

LW-5) Write a Java program to perform the following operations:

i) Create a file named "`alphabets.txt`" and insert appropriate data into it

ii) Read the file and copy all the consonants into a file named "`consonants.txt`" and all the vowels into a file named "`vowels.txt`"

iii) If numbers are encountered while parsing, then throw an exception (custom/ built-in) and continue parsing until the end of file is reached

LW-6) Write a Java program to implement the following scenario:

i) Create a file named "`integers.txt`" and insert large `n` random integers into it.

ii) Create three threads T1, T2 and T3 where each read `n/3` integers in sequence of occurrence of numbers from the file and sort into individual array.

iii) Main Thread waits for all the threads T1, T2 and T3 to complete sorting, then merges the output produced by the threads T1, T2 and T3, and writes the output to another file named "`sorted_integers.txt`"

LW-7) Write a Java program to implement a simple calculator that performs any four arithmetic computations of the choice specified the user. The program must provide appropriate GUI controls to enhance the interactivity of the application.

LW-8) Write a Java program that simulates client-server interaction using threads.

Methodology: Two threads should be created; one thread should act as server, and the other one should act as client. The server thread should accept request from client thread. Upon processing the request, the server thread should send back the acknowledgement message "Message Received" to client thread.

LW-9) Write a GUI based Java program that provides the feature to sort 100000 homogeneous random data. The program must provide the feature for the user to select the data type using appropriate GUI control. After sorting, the program must display the time taken to sort the data.

LW-10) Write a Java program to create a file named "`numbers.txt`" and generate 100000 integers so that each line contains only one integer. You are required to read each line from the file, check whether the integer is prime or not. If the number is prime, then print true in front of the number otherwise print false in the file next to the number. After processing the entire file, count the number of prime numbers in the given input file and display the count as the output.

Sample content of the file "numbers.txt" after checking for prime is given below:

```
2 false
5 true
9 false
13 true
```

…

Common Guidelines:

- The programs must make use of standard Java naming and coding conventions.
- The programs must be well documented. Use the comments to enhance readability of the code.
- Though the correctness of the programs is important, the programs must use object-oriented concepts (like encapsulation, polymorphism, inheritance, dynamic method dispatch, interfaces, packages, etc wherever applicable).
- Incorporate GUI features wherever applicable.
- Handle the exceptions wherever applicable.
- The GUI must be neat, clean, well designed and should be easy to use.

****