North South University

Department of Electrical and Computer Engineering

CSE 215L (Programming Language II Lab)

Lab 15: Introduction to File I/O

Objective:

- Understand the basics of file handling in Java.
- Read and write text files using FileWriter, FileReader, and Scanner.
- Apply File I/O concepts to solve problems involving user input and file operations.

Introduction to File I/O:

File I/O (Input/Output) in Java refers to the process of reading from and writing to files. This is useful when we need to store data permanently instead of using temporary storage like variables.

Commonly Used Java Classes for File Handling

Class	Description
FileWriter	Writes characters to a file.
FileReader	Reads characters from a file
Scanner	Reads input from a file (or user input).
IOException	Handles errors that occur during file operations.

○ Writing to a File (FileWriter)

The FileWriter class allows us to write text into a file.

```
FileWriter myFile = new FileWriter("example.txt");

myFile.write("Hello, Java File I/O!");

myFile.close(); // Always close the file after writing.
```

○ Reading from a File (FileReader and Scanner)

The FileReader class reads text from a file, and Scanner can be used for easy line-by-line reading.

```
FileReader reader = new FileReader("example.txt");

Scanner fileScanner = new Scanner(reader);

while (fileScanner.hasNextLine()) {

   System.out.println(fileScanner.nextLine());

}
```

- **Handling Exceptions (try-catch)**

  When working with files, errors may occur (e.g., file not found). We handle these using try-catch.

```
try {

   FileReader reader = new FileReader("example.txt");

} catch (IOException e) {

   System.out.println("An error occurred.");

}
```

<u>ClassWork:</u>

1. Write a program that takes integers from user and writes them into a file until user inputs a negative number. The program should then read the file and print sum and average of the numbers

2. Write a Java program that performs the following tasks:

   a. The program should continuously accept lines of text from the user and write them into a file named `lines.txt` until the user types **`"STOP"`**. Each entered line should be written on a new line in the file.
   b. After the user has finished entering data (by typing `"STOP"`), the program should read the content from the `lines.txt` file and print each line to the console.
   c. The program should count and display the number of non-empty lines (lines that are not completely empty, i.e., they contain at least one visible character or space).

<u>HomeWork:</u>

1. Create a Quiz class with id and mark. Now write a program that reads a file containing records of Quiz objects and initialize an array. The program should then print all the objects in the Quiz array and print the id of the student who obtained the highest mark.

   Sample File:
   113098 20

```
115089 15
345678 12
234566 18
```

Program Output:

```
ID:113098 mark:20
ID:115089 mark:15
ID:345678 mark:12
ID:234566 mark:18

Highest mark obtained by ID:113098
```