SWE 4201 - OBJECT ORIENTED CONCEPTS I

LAB 09 HANDOUT AND TASKS

Prepared by:
Md. Jubair Ibna Mostafa
Assistant Professor, IUT CSE

Department of Computer Science and Engineering
Islamic University of Technology

Contents

1 FILE

File write example

```java
public class Person {
    private String name;
    private int age;
    private String mobile;
    private String address;

    public Person(String name, int age, String mobile, String address)
    {
        this.name = name;
        this.age = age;
        this.mobile = mobile;
        this.address = address;
    }

    // getter setter
}
```

Listing 1: An Example of File write

File Write

```java
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<Person> people = new ArrayList<>();
        people.add(new Person("Alice", 25, "1234567890", "123 Main St")
    );
        people.add(new Person("Bob", 30, "9876543210", "456 Elm St"));
```

```
12          people.add(new Person("Charlie", 35, "5678901234", "789 Oak St"
    ));
13          people.add(new Person("David", 40, "2345678901", "012 Pine St")
    );
14          people.add(new Person("Eve", 45, "8765432109", "345 Cedar St"))
    ;
15
16          String filePath = "people.txt";
17          writePeopleToFile(people, filePath);
18      }
19
20      static void writePeopleToFile(List<Person> people, String filePath)
     {
21          try (BufferedWriter writer = new BufferedWriter(new FileWriter(
    filePath))) {
22              for (Person person : people) {
23                  writer.write(person.getName() + "," + person.getAge() +
     "," + person.getMobile() + "," + person.getAddress());
24                  writer.newLine();
25              }
26              System.out.println("Data has been written to file.");
27          } catch (IOException e) {
28              System.err.println("Error writing to file: " + e.getMessage
    ());
29          }
30      }
31  }
```

Listing 2: An Example of File append

File read example

```
1  import java.io.BufferedReader;
2  import java.io.FileReader;
3  import java.io.IOException;
4  import java.util.ArrayList;
5  import java.util.List;
```

```
6
7  public class Main {
8      public static void main(String[] args) {
9          List<Person> people = readPeopleFromFile("path/to/your/file.txt
       ");
10         for (Person person : people) {
11             System.out.println(person);
12         }
13     }
14
15     static List<Person> readPeopleFromFile(String filePath) {
16         List<Person> people = new ArrayList<>();
17         try (BufferedReader br = new BufferedReader(new FileReader(
       filePath))) {
18             String line;
19             while ((line = br.readLine()) != null) {
20                 String[] data = line.split(",");
21                 if (data.length == 4) {
22                     String name = data[0];
23                     int age = Integer.parseInt(data[1]);
24                     String mobile = data[2];
25                     String address = data[3];
26                     Person person = new Person(name, age, mobile,
       address);
27                     people.add(person);
28                 } else {
29                     System.out.println("Invalid data format: " + line);
30                 }
31             }
32         } catch (IOException e) {
33             System.err.println("Error reading file: " + e.getMessage())
       ;
34         }
35         return people;
36     }
```

```
37 }
```

Listing 3: An Example of File read

2 TASKS

2.1 File Task **Marks 20**

you are required to create a Java program that manages student information, course information, and their grades. Specifically, you need to implement functionality to store this information in separate files. Here's an outline of the program:

- Define the Student class with attributes id, name, address, residenceStatus (residence/non-residence), and mobile.
- Define the Course class with attributes code, name, credit, type (theory/lab)
- Define the Semester class with a list of courses.
- Create a file to store all students' information (comma-separated).
- Create a file for semester-wise course information (comma-separated).
- Create a file to store the student ID, course code, and obtained number (comma-separated).

Now, calculate grade for each students by reading those files and print to the console.

Sample File: students.txt

```
ID,Name,Address,Residence Status,Mobile
1,Alice,123 Main St,RESIDENCE,1234567890
2,Bob,456 Elm St,NON_RESIDENCE,9876543210
```

Sample File: courses.txt

```
Code,Name,Credit,Type,Semester
CSE101,Introduction to Programming,3,THEORY,1
CSE102,Data Structures and Algorithms,3,LAB,1
```

Sample File: grades.txt

```
Student ID,Course Code,Obtained Number
1,CSE101,85
1,CSE102,90
2,CSE101,78
2,CSE102,82
```