Lab Exercises for Week 12: Programming using Java I/O Streams

BITP 3123 Distributed Application Development

Emaliana Kasmuri

Fakulti Teknologi Maklumat dan Komunikasi

Universiti Teknikal Malaysia Melaka

Table of Contents

Description of Document

This document described the lab exercises that the students must complete to demonstrate their ability to process input and output using Java I/O streams.

Learning Outcomes

At the end of this lab, the student should be able to:-

1. Execute the samples of program that uses the classes of java.io.
2. Write Java programs to process byte-based data using the suitable classes from package **java.io**.
3. Write Java program and character-based data using the suitable classes from package **java.io**.

Tools

1. Eclipse for Java EE.
2. Samples of code from github.

Prerequisite

1. Create a new project in Eclipse named **iostreamdemo**. This project will contains all demo programs that is going to be downloaded from Github.
2. Create another new project named **lab12**. This project will contain all the programs that the student will created from this lab.

Exercise 1 Observing Byte-Based Data Processing

1. Download and import a Java program from this link into the Eclipse project created in Prerequisite.
2. Run `WriteByteDemo.java`. The program should produce an output indicating the program execution has ended and an output file has been produced.
3. Refresh the project where `WriteByteDemo.java` has been imported. A file with similar name should be available in the project.
4. Download and import a Java program from this link.
5. Run `ReadByteDemo.java`. Observe and record the output in the following space.
6. Compare the similarity of data created in `WriteByteDemo.java` with the ones produced in step 5.
7. Compare the human readability of data produced from Step 5 with the data in the output file produced in step 2.
8. Record your observation for your own reference.

Exercise 2 Analysis of Byte-Based Data Processing

Record the answers for these questions in ulearn.

1. What is the name of the stream class that is used in the `WriteByteDemo.java` to write data into the target storage?
2. What is the name of the object used in `WriteByteDemo.java` to write data into the target storage?
3. What is the output produced on the console from `WriteByteDemo.java`?
4. What is the name of the file created from `WriteByteDemo.java` ?
5. What is the name of the stream class that used in the `ReadByteDemo.java` to read data from the input source?
6. What is the name of the object used in `ReadByteDemo.java` to read data from the input source?
7. What is the output produced on the console from `ReadByteDemo.java`?
8. Which of the following are the source of input data from `ReadByteDemo.java`?
   A) A file named outByteDemo.dat.
   B) An input from user.
   C) An object named fisObject.
   D) A class named FileInputStream.
9. Are there any differences between the data created in `WriteByteDemo.java` and `ReadByteDemo.java?`
10. What are the differences and why?

Exercise 3 Observing Primitive Data Processing in Byte-Based Form

1. Download and import a Java program from this link into the Eclipse project created in Prerequisite.
2. Run `PrimitiveDataGenerator.java`. The program should produce an output indicating the program execution has ended and an output file has been produced.
3. Refresh the project where `PrimitiveDataGenerator.java` has been imported. A file with similar name as produced from step 2 should be available in the project after the refresh.
4. Download and import a Java program from this link.
5. Run `PrimitiveDataReader.java`. Observe the output.
6. Compare the similarity of data created in `PrimitiveDataGenerator.java` with the output produced in step 5.
7. Compare the human readability of data produced from step 5 with the data in the output file produced in step 2.
8. Record your observation for your own reference.

Exercise 4 Analysis of Observation

1. Draw a diagram that illustrate stream interaction in `PrimitiveDataGenerator.java.`
2. Draw a class digram that depicts the relationship of all classes in `PrimitiveDataGenerator.java.`
3. Draw a diagram that illustrate stream interaction in `PrimitiveDataReader.java.`
4. Draw a class digram that depicts the relationship of all classes in `PrimitiveDataReader.java.`

Exercise 5 Processing Daily Rainfall Data A Station

This exercise requires the programmer to design the solution before proceed with the implementation.

Task A: Data Creation

Design a Java solution to create a set of data in a file using `java.io.DataOutputStream`. The data represent a 6 days reading of daily rainfall from station Simpang Ampat in Alor Gajah district. The actual is available at this link. The design shall consist a diagram that describe Java I/O interaction to create the data.

Task B: Implementation of Data Creation

Implement the design that was created in Task A. The solution must display an appropriate message that indicates the creation of the data has ended.

Task C: Data Consumption

Design a Java solution to consume a set of data that was created in Task B using `java.io.DataInputStream`. The solution should display the 6-days rainfall data , compute and display the average of the rainfall from the selected station. The design shall consist of two diagrams. The first describes Java I/O interaction to consume the data. The second diagram shall describe the Java classes that will display rainfall data and compute the average of the rainfall for 6 days.

Task D: Implementation of Data Consumption

Implement the design that was created in Task C. The solution must adhere to all requirements described in Task C

This exercise does not require the programmer to read from the mentioned website using web API. Instead, the programmer should copy the data in the solution. The programmer is encouraged to use their creativity to create the data efficiently and display the output meaningfully. The programmer is also advised to treat the data copied from the website as integer data.

The source codes created from this exercises must be well-written, well-documented, and comply to the Java standard. Use appropriate names for all classes in your solution. Share the solutions in your Github. Provide meaningful description that include screen shot of output, design and steps of execution for your solution.

Exercise 6 Processing Daily Rainfall Data from Selected Station using Byte-Based Stream

Create two Java classes to demonstrate the I/O process for the data described in this link. The first class should create the input using `java.io.DataOutputStream` for two stations in each district in Melaka. The data should consist of the station id, station name, name of district and 6-days reading of daily rainfall.

The second class should read the data created from the first class using `java.io.DataInputStream`. All data must be displayed. The second class should compute average of rainfall for each of the station and displayed it. The output should also indicate the number of stations and districts.

This exercise does not require the programmer to read from the mentioned website using web API. Instead, the programmer should copy the data in the solution. The programmer is encouraged to use their creativity to create the data efficiently and display the output meaningfully. The programmer is also advised to treat the data copied from the website as integer data.

The source codes created from this exercises must be well-written, well-documented, and comply to the Java standard. Use appropriate names for all classes in your solution. Share the solutions in your Github. Provide meaningful description that include screen shot of output, design and steps of execution for your solution.

Exercise 7 Processing Daily Rainfall Data from Selected Station using Character-Based Stream

Re-create the solutions described in Exercise 5 using the suitable subclasses in `java.io.Reader` and `java.io.Writer`. Share the solutions in Github.

End of Lab Exercises