Exercise 01

Write down short notes about following topics.

1. Streams
2. Character Streams
3. Byte Streams
4. Input Streams
5. Output Streams

1. Streams:
    - In computing, a stream is a sequence of data elements made available over time.
    - Streams can be used for various purposes, such as reading from or writing to files, network communication, and processing large datasets.
    - They are a fundamental concept in I/O operations, allowing data to be transferred efficiently between a source and a destination.
2. Character Streams:
    - Character streams are a type of stream used for handling character-based data, such as text.
    - They are often used for reading and writing text files, where each character is represented using a character encoding like UTF-8 or ASCII.
    - Common character stream classes in Java include FileReader, FileWriter, and InputStreamReader.
3. Byte Streams:
    - Byte streams are used for handling binary data, which can represent any type of data, including text, images, audio, or other binary formats.
    - Byte streams are suitable for reading and writing raw bytes from and to files or other sources.
    - Examples of byte stream classes in Java include FileInputStream and FileOutputStream.
4. Input Streams:
    - Input streams are used for reading data from a source, such as a file, keyboard, or network connection.
    - They provide methods for reading data in a sequential manner, allowing you to retrieve data as needed.

- In Java, common input stream classes include FileInputStream and InputStreamReader.

5. Output Streams:

- Output streams are used for writing data to a destination, such as a file, console, or network connection.
- They provide methods for writing data in a sequential manner, allowing you to send data in chunks or one byte at a time.
- In Java, common output stream classes include FileOutputStream and OutputStreamWriter.

Exercise 02

Reading Console Input The object of BufferedReader class can be used to take inputs from the keyboard.

1. Write a small program to read a character from the console using the BufferedReader object in Java.

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;


public class ExerciseTwoPartOne {

public static void main(String[] args) {

// Create a BufferedReader object for reading from the console

BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));


try {

System.out.print("Enter a character: ");

// Read a line of input from the console

String input = reader.readLine();


// Check if the input is not empty

if (!input.isEmpty()) {

// Get the first character from the input

char character = input.charAt(0);
```

```
// Display the character
System.out.println("You entered: " + character);
} else {
System.out.println("No input provided.");
}
} catch (IOException e) {
System.err.println("Error reading input: " + e.getMessage());
} finally {
try {
// Close the BufferedReader when done
reader.close();
} catch (IOException e) {
System.err.println("Error closing BufferedReader: " + e.getMessage());
}
}
}
}
```

2. Modify the above program to read the String input from Keyboard in Java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class ExerciseTwoPartTwo {
public static void main(String[] args) {
// Create a BufferedReader object for reading from the console
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));

try {
```

```
System.out.print("Enter a string: ");

// Read a line of input from the console

String input = reader.readLine();


// Check if the input is not empty

if (!input.isEmpty()) {

// Display the string entered by the user

System.out.println("You entered: " + input);

} else {

System.out.println("No input provided.");

}

} catch (IOException e) {

System.err.println("Error reading input: " + e.getMessage());

} finally {

try {

// Close the BufferedReader when done

reader.close();

} catch (IOException e) {

System.err.println("Error closing BufferedReader: " + e.getMessage());

}

}

}

}
```

Exercise 03

1. Write an example program to demonstrate Byte InputStream and OutputStream in Java.

```
import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;


public class ExerciseThreePartOne {
```

```
public static void main(String[] args) {

// Create a byte array to write into a file

byte[] dataToWrite = "Hello, Byte Streams!".getBytes();


// Define the file name

String fileName = "byte_stream_example.txt";


try {

// Create a FileOutputStream to write data to a file

FileOutputStream outputStream = new FileOutputStream(fileName);


// Write data to the file

outputStream.write(dataToWrite);


// Close the output stream

outputStream.close();


// Create a FileInputStream to read data from the file

FileInputStream inputStream = new FileInputStream(fileName);


// Read data from the file

byte[] dataRead = new byte[dataToWrite.length];

inputStream.read(dataRead);


// Close the input stream

inputStream.close();


// Display the read data as a string

System.out.println("Data read from file: " + new String(dataRead));

} catch (IOException e) {
```

```
System.err.println("Error: " + e.getMessage());
}
}
}
```

2. Write an example program to demonstrate Character Reader and Writer Stream in Java.

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class ExerciseThreePartTwo {
public static void main(String[] args) {
// Define the file name
String fileName = "character_stream_example.txt";

try {
// Create a FileWriter to write character data to a file
FileWriter writer = new FileWriter(fileName);

// Write a string to the file
writer.write("Hello, Character Streams!");

// Close the writer
writer.close();

// Create a FileReader to read character data from the file
FileReader reader = new FileReader(fileName);

// Read data from the file
char[] dataRead = new char[50];
```

```
int bytesRead = reader.read(dataRead);


// Close the reader

reader.close();


// Display the read data as a string

System.out.println("Data read from file: " + new String(dataRead, 0,
bytesRead));

} catch (IOException e) {

System.err.println("Error: " + e.getMessage());

}

}

}
```

Exercise 04

Writing to a File Create a Java program to write a string to a file.

```
import java.io.BufferedWriter;

import java.io.FileWriter;

import java.io.IOException;


public class ExerciseFour {

public static void main(String[] args) {

// Define the file name and the string to write

String fileName = "output.txt";

String content = "Hello, this is a string written to a file!";


try {

// Create a FileWriter with the specified file name

FileWriter fileWriter = new FileWriter(fileName);
```

```
// Wrap the FileWriter in a BufferedWriter for efficient writing

BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);



// Write the string to the file

bufferedWriter.write(content);



// Close the BufferedWriter (this will also close the FileWriter)

bufferedWriter.close();



System.out.println("String has been written to " + fileName);

} catch (IOException e) {

System.err.println("Error: " + e.getMessage());

}

}

}
```

Exercise 05

Reading from a File Create a file including your details such as name, address etc. manually. Write a Java program to read the content of the file.

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class ExerciseFive {
public static void main(String[] args) {
// Define the file name
String fileName = "my_details.txt";

try {
// Create a FileReader to read from the file
FileReader fileReader = new FileReader(fileName);

// Wrap the FileReader in a BufferedReader for efficient reading
BufferedReader bufferedReader = new BufferedReader(fileReader);

String line;
System.out.println("Contents of the file:");

// Read and print each line of the file
```

```
while ((line = bufferedReader.readLine()) != null) {
System.out.println(line);
}

// Close the BufferedReader (this will also close the FileReader)
bufferedReader.close();
} catch (IOException e) {
System.err.println("Error: " + e.getMessage());
}
}
}
```