Experiment No.____________ **Date___/___/2020**

TITLE OF EXPERIMENT: - A program in Java to learn File Handling

DIVISION:______________________________ **BRANCH: ______________________**

BATCH:_________________________________ **ROLL NO.: _____________________**

PERFORMED ON DATE: ____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 12

Aim: Write a java program in which data is read from one file and should be written in another file line by line.

Objective: To learn, how to read data from one file and write the data in another file, using Java program. It is very simple in Java Programming. The output will be displayed in the Run module.

Software:

1. Eclipse
2. **JDK 16**

Theory:

File handling plays a major role in doing so as the first essential step is writing content to a file. For this is one must know how to write content in a file using the **FileWriter class**. The secondary step is reading content from a file and print the same. For this, one must have good hands on **File Reader class** to do so.
Now in order to read content from one file and write it into another file, it is already discussed achieving the same how to write content on a file and also how to read contents from a file. Now, the time to combine both of them. Now we will use FileReader class to read the contents from a class and the **FileWriter class** to write it on another file.

I/O Streams:

Java provides to read and write data where a Stream represents an input source or an output I/O Streams destination which could be a file, i/o devise, other programs, etc.

In general, a Stream will be an input stream or, an output stream.

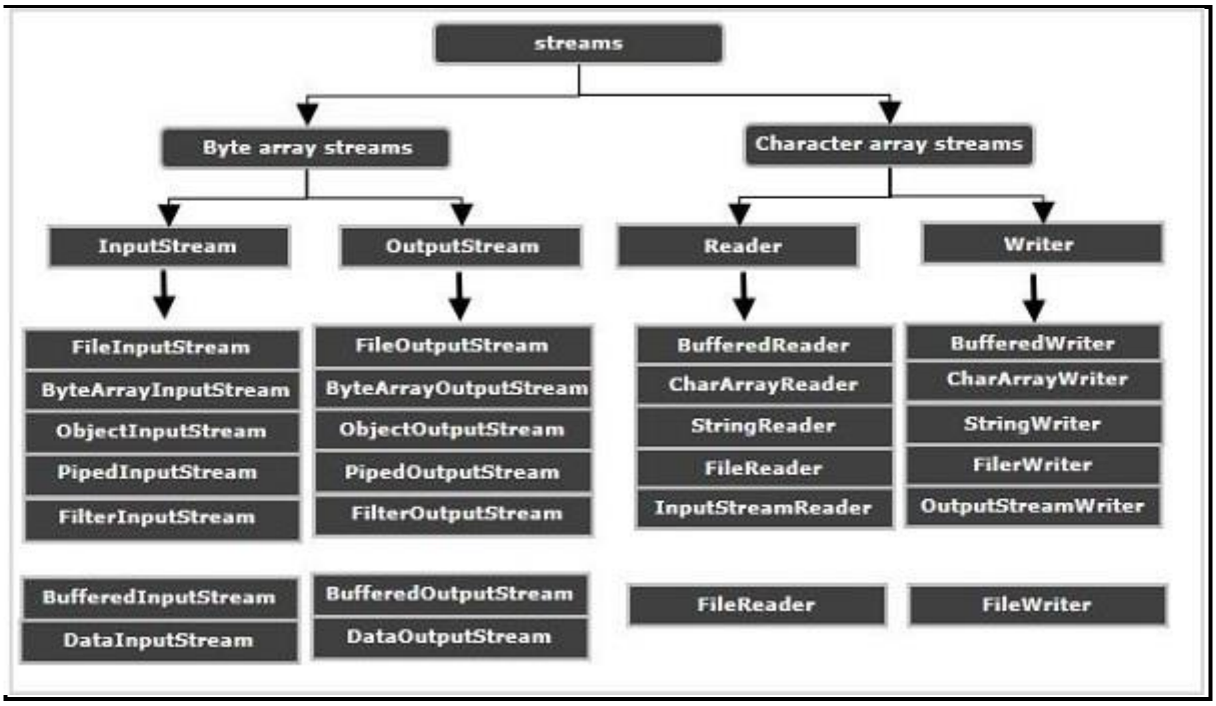
- **InputStream** − This is used to read data from a source.

- **OutputStream** − This is used to write data to a destination.

Based on the data they handle there are two types of streams −

- **Byte Streams** − These handle data in bytes (8 bits) i.e., the byte stream classes read/write data of 8 bits. Using these you can store characters, videos, audios, images, etc.
- **Character Streams** − These handle data in 16 bit Unicode. Using these you can read and write text data only.

The following diagram illustrates all the input and output Streams (classes) in Java.



Among these, you can read the contents of a file using Scanner, BufferedReader and, FileReader classes.

In the same way, you can write data into a file using BufferedWriter, FileOutputStream, FileWriter.

Program:

The object of FileInputStream is created by passing the object of File with the constructor. Here is example from our program:

inStream = new FileInputStream(inputFile);

And to read the data we have use the read() method of FileInputStream class. Here is example code:

inStream.read(buffer))

Then the read data is finally saved to another file using the FileOutputStream class of Java.

FileOutputStream in Java

The FileOutputStream class is part of java.io package in Java and this class is used for writing bytes to the output stream. In this example we will use this class we will use the FileOutputStream class to write bytes to text file. This class is designed to support both binary stream and text stream as it writes the data in the bytes format. In out example we have instantiate the FileOutputStream class with following code:

outStream = new FileOutputStream(outFile);

For writing the bytes following is used in our program:

outStream.write(buffer, 0, length);

So, with the help of FileOutputStream class we are able to write the data to another file.

File class in Java

The file class is used in Java for working with a file or a directory path. This class is used for working with the files and rectories in Java. It provides the methods for working with files and directories. In our example we have used the following function to open the input and output file. Here is the code:

File inputFile = new File("D:\\examples\\input.txt");
File outFile = new File("D:\\examples\\out.txt");

Here is complete code which reads the data from one text file and then write to another text file:

```
import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

/*

 * Example of reading data from one text file

 * and writing the data to another file


 */


public class ReadTextFileWriteToAnother {

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

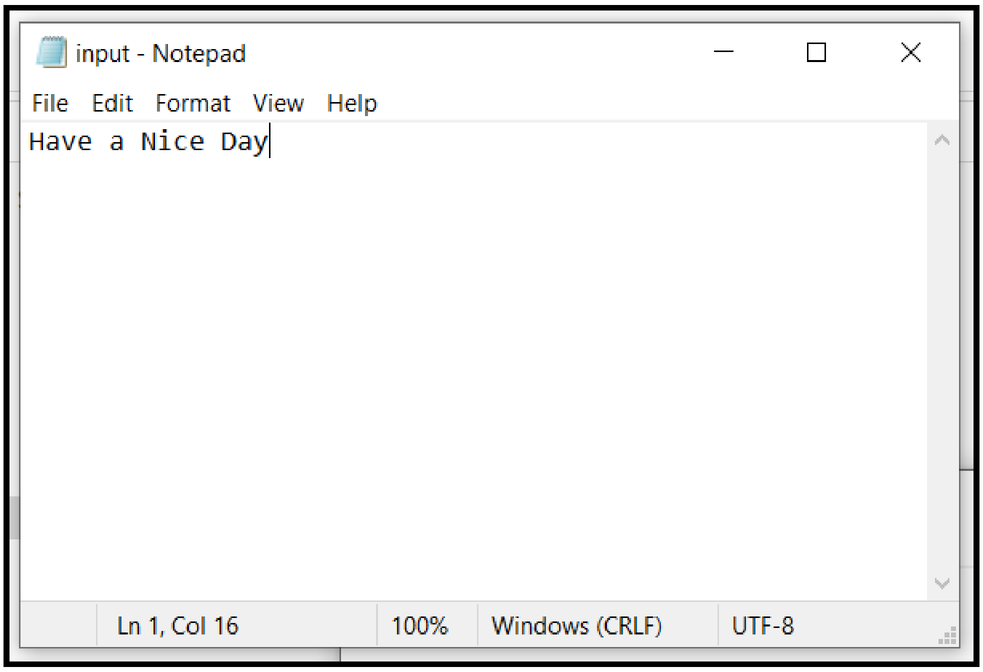
```
//create input and output stream objects
FileInputStream inStream = null;
FileOutputStream outStream = null;

try {
        //Files objects
        File inputFile = new File("D:\\input.txt");
        File outFile = new File("D:\\out.txt");

        //Intialize input and output streams
        inStream = new FileInputStream(inputFile);
        outStream = new FileOutputStream(outFile);

        //The buffer size for reading data
        byte[] buffer = new byte[1024];

        int length;
        //Copy data to another file
        while ((length = inStream.read(buffer)) > 0) {
                outStream.write(buffer, 0, length);
        }

        // Closing the input/output file streams
        inStream.close();
        outStream.close();

        System.out.println("Written Content to another file.");

} catch (IOException e) {
        e.printStackTrace();
}
```

```
}
}
```

If you run the above code it will copy the content of one text file into another text file. But you should change the path of the files when you run your program on your computer. Here is the code where you can change the input and output file paths:

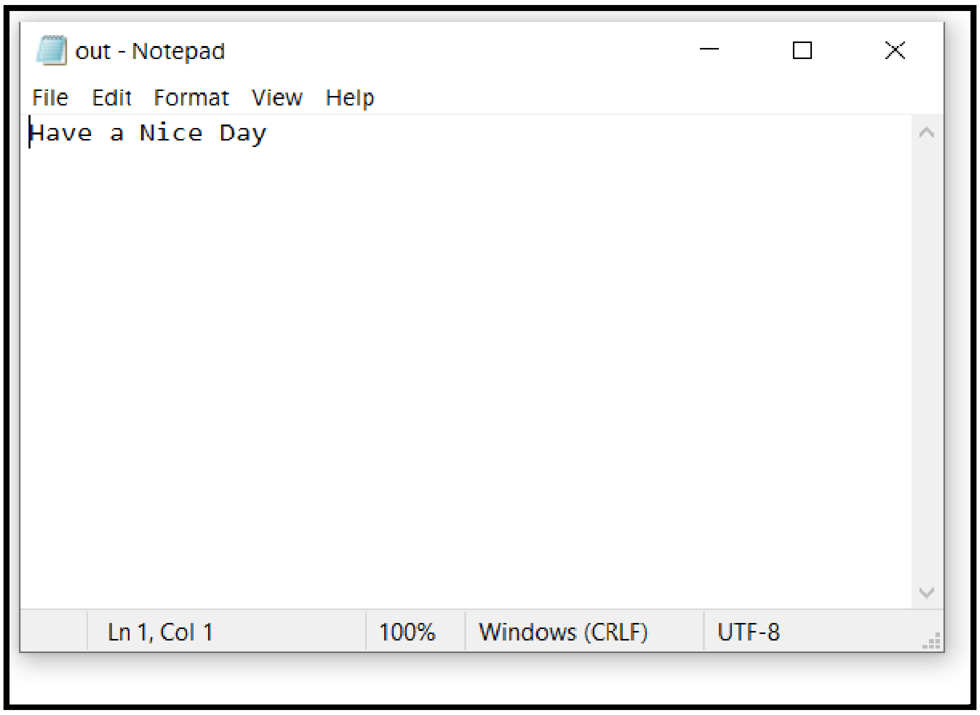
File inputFile = new File("D:\\examples\\input.txt");
File outFile = new File("D:\\examples\\out.txt");

Step 1: Open the Notepad and write the content “Have a Nice Day” and save the file in “input.txt”.



```
Have a Nice Day
```

Step 2: Open another Notepad and save the empty file in “out.txt”.



```
Have a Nice Day
```

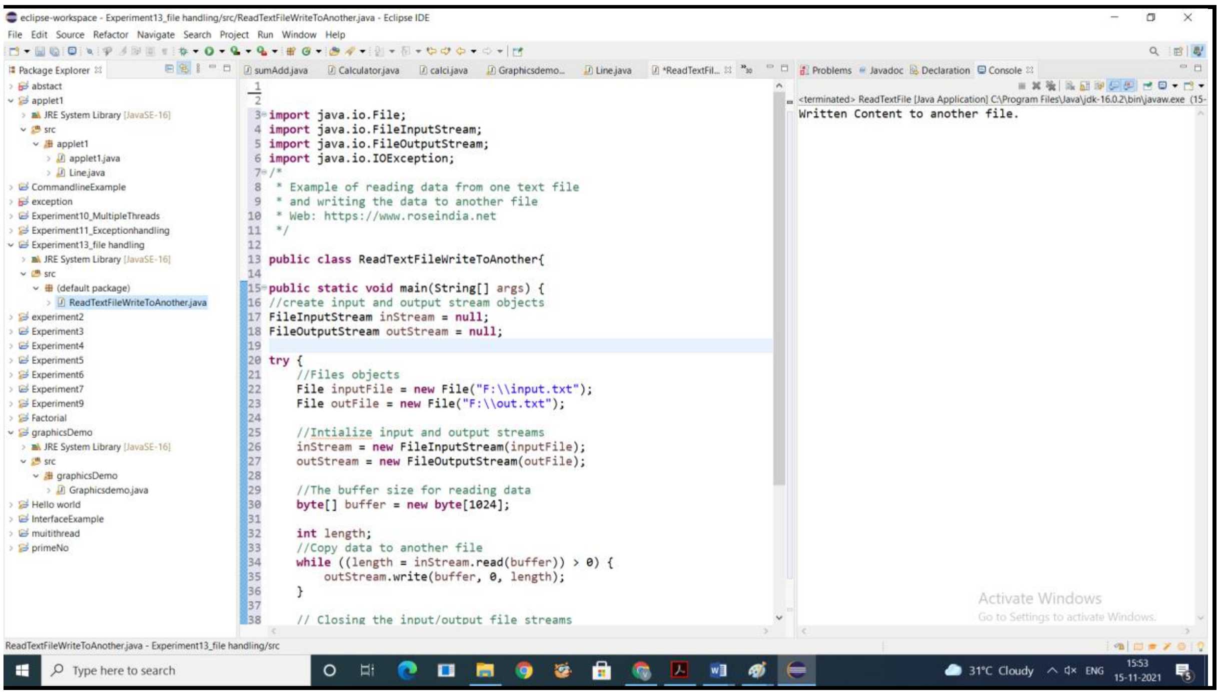
Step 3: Open the "out.txt" file. The content was copied in the file.

Output:

```
Written Content to another file.
```

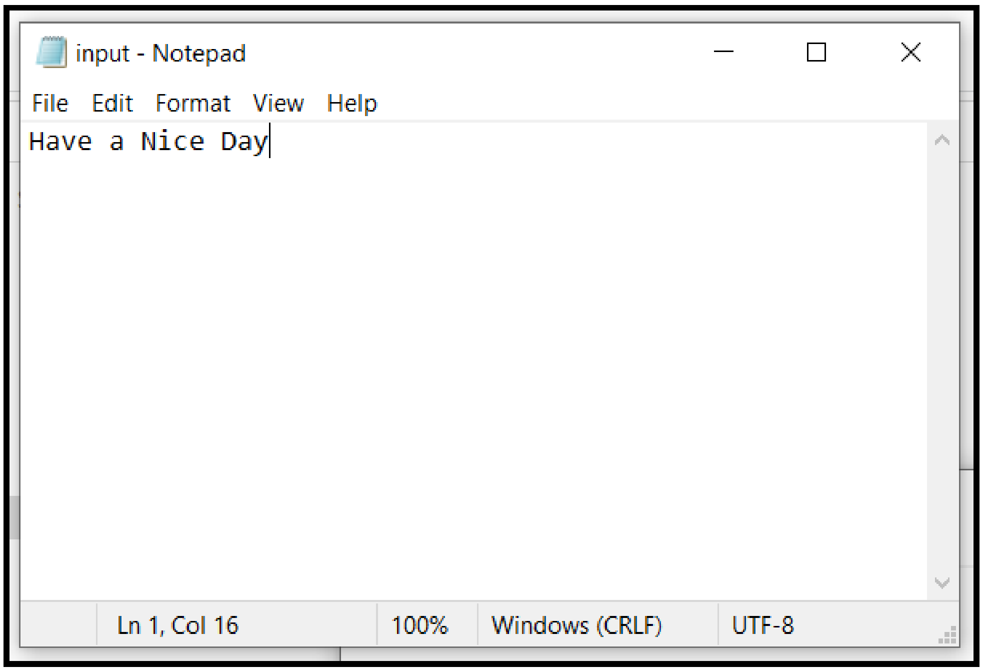
Conclusion:

Screenshot’s of Program and Result:



```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
/*
 * Example of reading data from one text file
 * and writing the data to another file
 * Web: https://www.roseindia.net
 */

public class ReadTextFileWriteToAnother{

public static void main(String[] args) {
//create input and output stream objects
FileInputStream inStream = null;
FileOutputStream outStream = null;

try {
    //Files objects
    File inputFile = new File("F:\\input.txt");
    File outFile = new File("F:\\out.txt");

    //Intialize input and output streams
    inStream = new FileInputStream(inputFile);
    outStream = new FileOutputStream(outFile);

    //The buffer size for reading data
    byte[] buffer = new byte[1024];

    int length;
    //Copy data to another file
    while ((length = inStream.read(buffer)) > 0) {
        outStream.write(buffer, 0, length);
    }

    // Closing the input/output file streams
```

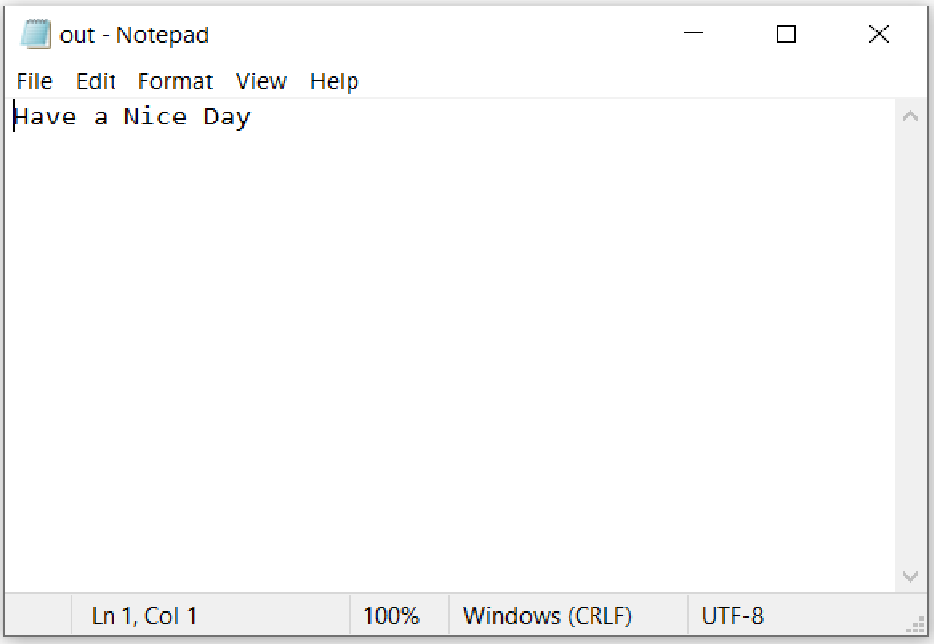
```
<terminated> ReadTextFile [Java Application] C:\Program Files\Java\jdk-16.0.2\bin\javaw.exe (15-
Written Content to another file.
```



input - Notepad

```
Have a Nice Day
```

Ln 1, Col 16   100%   Windows (CRLF)   UTF-8



```
Have a Nice Day
```