

JAVA PROGRAMMING LAB (ETCS – 357)

LAB-10

Faculty Name :

Dr. Sandeep Tayal

Student Name:

Udit Agarwal

12914802719

Sem 5

5C7



Maharaja Agrasen Institute of Technology, PSP Area,

Sector – 22, Rohini, New Delhi – 110085

INDEX

S.no	Experiments	Date of performance	Date checked	Marks					Total Marks	Sign.
				R1	R2	R3	R4	R5		
10.1	Write a program that read from a file and write to file.	2.12.21								
10.2	Convert the content of a given file into the uppcase content of the same file.	2.12.21								
10.3	JDBC (Database connectivity with MS-Access)	2.12.21								

EXPERIMENT 10.1

AIM : Write a program that read from a file and write to file.

Theory :

FileReader : This class inherits from the `InputStreamReader` class. `FileReader` is used for reading streams of characters. This class has several constructors to create required objects. Following is the list of constructors provided by the `FileReader` class.

FileWriter : `FileWriter` is meant for writing streams of characters. For writing streams of raw bytes, consider using a `FileOutputStream`. `FileWriter` creates the output file if it is not present already.

.read() method : The `read()` method of `Reader` Class in Java is used to read a single character from the stream. This method blocks the stream till:

Write() method : The `write()` method of [Reader](#) Class in Java is used to read a single character from the stream. This method blocks the stream till:

Source Code :

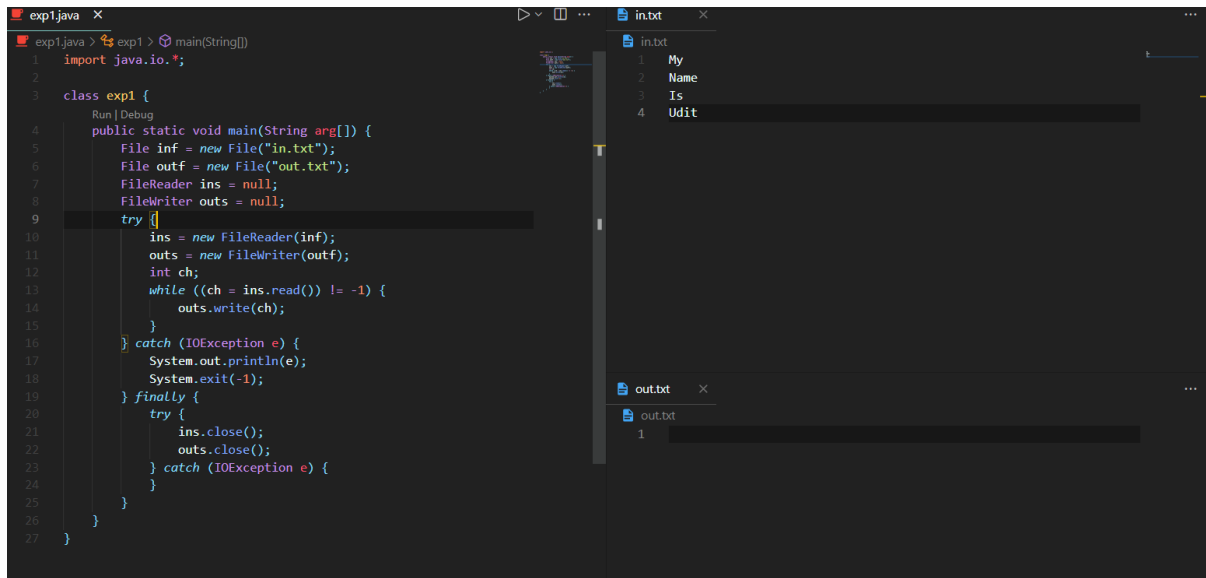
```
import java.io.*;

class exp1 {
    public static void main(String arg[]) {
        File inf = new File("in.txt");
        File outf = new File("out.txt");
        FileReader ins = null;
        FileWriter outs = null;
        try {
            ins = new FileReader(inf);
            outs = new FileWriter(outf);
            int ch;
            while ((ch = ins.read()) != -1) {
                outs.write(ch);
            }
        } catch (IOException e) {
            System.out.println(e);
            System.exit(-1);
        } finally {
            try {
                ins.close();
                outs.close();
            }
        }
    }
}
```

```
        } catch (IOException e) {  
        }  
    }  
}
```

Output :

Before :



```
exp1.java x  
exp1 > main(String[])  
1 import java.io.*;  
2  
3 class exp1 {  
4     public static void main(String arg[]) {  
5         File inf = new File("in.txt");  
6         File outf = new File("out.txt");  
7         FileReader ins = null;  
8         FileWriter outs = null;  
9         try {  
10            ins = new FileReader(inf);  
11            outs = new FileWriter(outf);  
12            int ch;  
13            while ((ch = ins.read()) != -1) {  
14                outs.write(ch);  
15            }  
16        } catch (IOException e) {  
17            System.out.println(e);  
18            System.exit(-1);  
19        } finally {  
20            try {  
21                ins.close();  
22                outs.close();  
23            } catch (IOException e) {  
24            }  
25        }  
26    }  
27 }
```

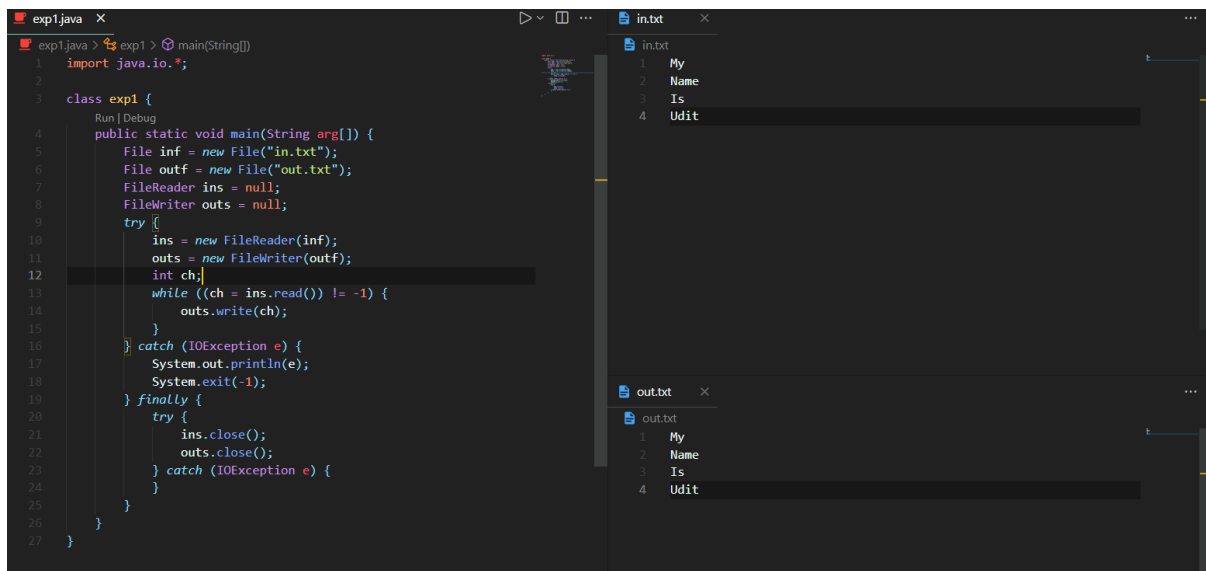
in.txt

```
1 My  
2 Name  
3 Is  
4 Udit
```

out.txt

```
1
```

After :



```
exp1.java x  
exp1 > main(String[])  
1 import java.io.*;  
2  
3 class exp1 {  
4     public static void main(String arg[]) {  
5         File inf = new File("in.txt");  
6         File outf = new File("out.txt");  
7         FileReader ins = null;  
8         FileWriter outs = null;  
9         try {  
10            ins = new FileReader(inf);  
11            outs = new FileWriter(outf);  
12            int ch;  
13            while ((ch = ins.read()) != -1) {  
14                outs.write(ch);  
15            }  
16        } catch (IOException e) {  
17            System.out.println(e);  
18            System.exit(-1);  
19        } finally {  
20            try {  
21                ins.close();  
22                outs.close();  
23            } catch (IOException e) {  
24            }  
25        }  
26    }  
27 }
```

in.txt

```
1 My  
2 Name  
3 Is  
4 Udit
```

out.txt

```
1 My  
2 Name  
3 Is  
4 Udit
```

EXPERIMENT 10.2

AIM : Convert the content of a given file into the uppercase content of the same file.

Theory :

FileReader : This class inherits from the `InputStreamReader` class. `FileReader` is used for reading streams of characters. This class has several constructors to create required objects. Following is the list of constructors provided by the `FileReader` class.

FileWriter : `FileWriter` is meant for writing streams of characters. For writing streams of raw bytes, consider using a `FileOutputStream`. `FileWriter` creates the output file if it is not present already.

Buffered Reader : Java `BufferedReader` class is used to read the text from a character-based input stream. It can be used to read data line by line by `readLine()` method. It makes the performance fast. It inherits `Reader` class.

Buffered Writer : Java `BufferedWriter` class is used to provide buffering for `Writer` instances. It makes the performance fast. It inherits `Writer` class. The buffering characters are used for providing the efficient writing of single arrays, characters, and strings.

Source Code :

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.nio.file.Path;

public class exp2 {
    public static void main(String[] arguments) {
        String sourceName = "in.txt";
        try {
            Path source =
FileSystems.getDefault().getPath(sourceName);
            Path temp = FileSystems.getDefault().getPath("tmp_" +
sourceName);
            FileReader fr = new FileReader(source.toFile());
```

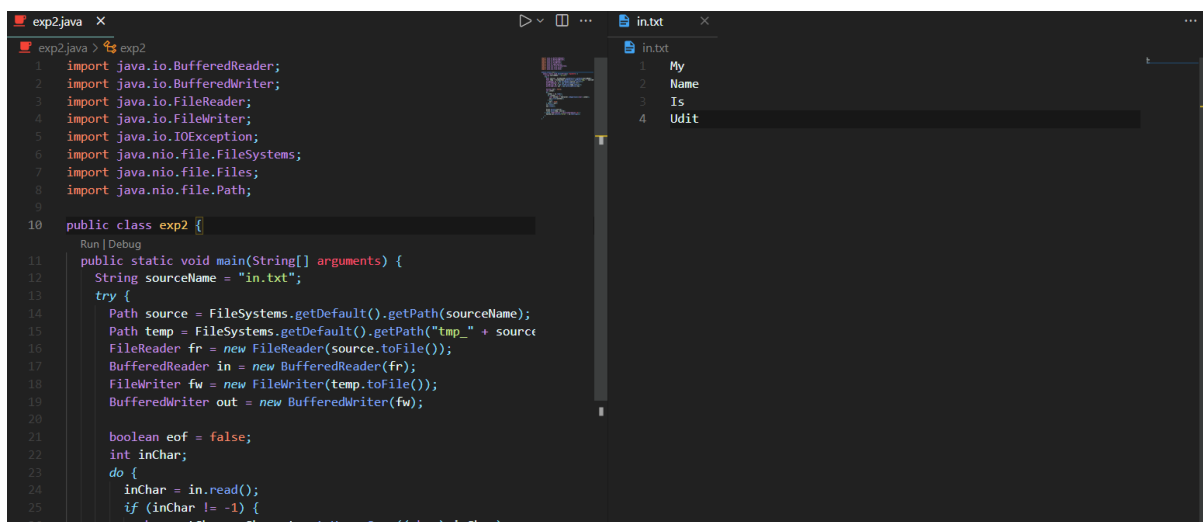
```
        BufferedReader in = new BufferedReader(fr);
        FileWriter fw = new FileWriter(temp.toFile());
        BufferedWriter out = new BufferedWriter(fw);

        boolean eof = false;
        int inChar;
        do {
            inChar = in.read();
            if (inChar != -1) {
                char outChar = Character.toUpperCase((char) inChar);
                out.write(outChar);
            } else
                eof = true;
        } while (!eof);
        in.close();
        out.close();

        Files.delete(source);
        Files.move(temp, source);
    } catch (IOException | SecurityException se) {
        System.out.println("Error" + se.toString());
    }
}
}
```

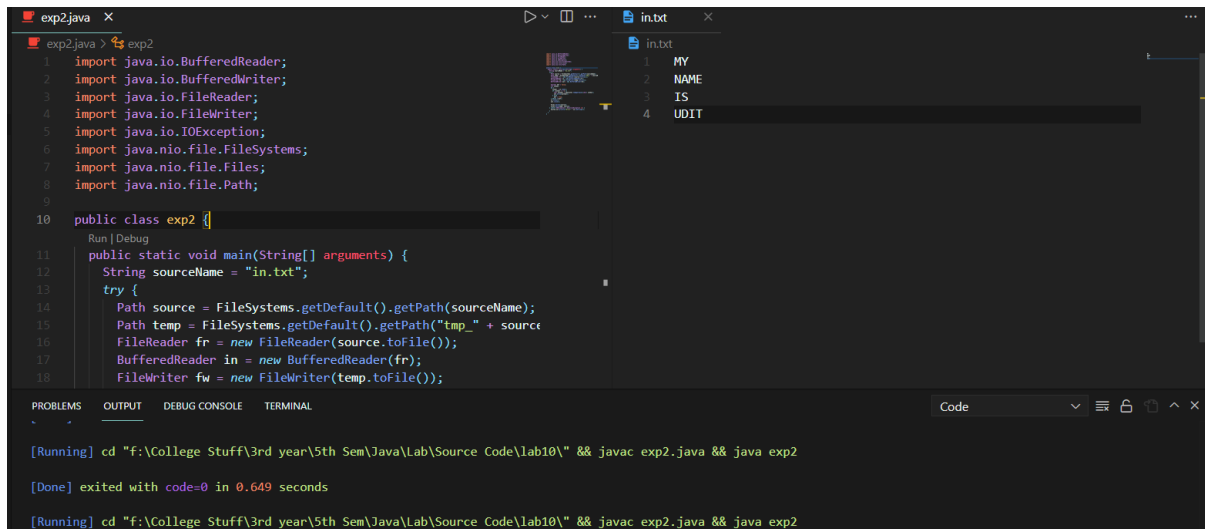
Output :

Before :



The screenshot shows an IDE with a Java file named 'exp2.java' and a text file named 'in.txt'. The Java code in 'exp2.java' is a class 'exp2' with a 'main' method. It imports various Java I/O and file handling classes. The 'main' method sets 'sourceName' to 'in.txt', finds the file, creates a temporary file, and sets up a buffered reader and writer. It then reads the content of 'in.txt' and writes it to the temporary file, converting each character to uppercase. The 'in.txt' file currently contains the text: 'My', 'Name', 'Is', 'Udit' on separate lines.

After :



The screenshot shows an IDE with two files open: `exp2.java` and `in.txt`. The `exp2.java` file contains a Java program that reads from `in.txt` and writes to a temporary file. The `in.txt` file contains the text: `MY`, `NAME`, `IS`, and `UDIT`. The terminal at the bottom shows the execution of the program, which runs successfully with exit code 0.

```
exp2.java x
1 import java.io.BufferedReader;
2 import java.io.BufferedWriter;
3 import java.io.FileReader;
4 import java.io.FileWriter;
5 import java.io.IOException;
6 import java.nio.file.FileSystems;
7 import java.nio.file.Files;
8 import java.nio.file.Path;
9
10 public class exp2 {
11     public static void main(String[] arguments) {
12         String sourceName = "in.txt";
13         try {
14             Path source = FileSystems.getDefault().getPath(sourceName);
15             Path temp = FileSystems.getDefault().getPath("tmp_" + sourceName);
16             FileReader fr = new FileReader(source.toFile());
17             BufferedReader in = new BufferedReader(fr);
18             FileWriter fw = new FileWriter(temp.toFile());
19
20             String line;
21             while ((line = in.readLine()) != null) {
22                 fw.write(line + "\n");
23             }
24             in.close();
25             fw.close();
26         } catch (IOException e) {
27             e.printStackTrace();
28         }
29     }
30 }
```

```
in.txt
1 MY
2 NAME
3 IS
4 UDIT
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[Running] cd "F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code\lab10\" && javac exp2.java && java exp2
[Done] exited with code=0 in 0.649 seconds
[Running] cd "F:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code\lab10\" && javac exp2.java && java exp2
```

EXPERIMENT 10.3

AIM : JDBC (Database connectivity with MS-Access)

Source Code :

```
import java.sql.*;
public class exp3 {
    public static void main(String[] args) {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            Connection conn =
DriverManager.getConnection("jdbc:odbc:Test");
            Statement st = conn.createStatement();
            String sql = "Select * from Table1";
            ResultSet rs = st.executeQuery(sql);
            while (rs.next()) {
                System.out.println("\n" + rs.getString(1) +
"\t" + rs.getString(2) + "\t" + rs.getString(3) + "\t"
+ rs.getString(4));
            }
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}
```

Output :

```
[Running] cd "f:\College Stuff\3rd year\5th Sem\Java\Lab\Source Code\lab10\" && javac exp3.java && jav
sun.jdbc.odbc.JdbcOdbcDriver
ID: 1, Name: Udit, Age: 22, Branch: cse
ID: 2, Name: Pratyaksh, Age: 21, Branch: ece
ID: 3, Name: Ankit, Age: 22, Branch: mae
ID: 3, Name: Ahmad, Age: 20, Branch: it
```


VIVA QUESTIONS :

Q1) What is JDBC in Java?

Ans. JDBC(Java Database Connectivity) is a Java API, which is helpful in interaction with the database to retrieve, manipulate and process the data using SQL. It will make use of JDBC drivers for connecting with the database. By using JDBC, we can access tabular data stored in various types of relational databases such as Oracle, MySQL, MS Access, etc.

Q2). What is a stream and its types and classes?

Ans. A Stream is an abstraction that either produces or consumes information. There are two types of Streams :

Byte Streams: Provide a convenient means for handling input and output of bytes.

Character Streams: Provide a convenient means for handling input & output of characters.

Byte Streams classes: Are defined by using two abstract classes, namely InputStream and OutputStream.

Character Streams classes: Are defined by using two abstract classes, namely Reader and Writer.

Q3). What is the difference between InputStream and OutputStream in Java?

Ans. InputStream is used to read data from sources like File, Socket, or Console, while OutputStream is used to write data into a destination like a File, Socket, or Console.

Q4) What is ODBC Bridge driver ?

Ans. In this, the JDBC–ODBC bridge acts as an interface between the client and database server. When a user uses a Java application to send requests to the database using JDBC–ODBC bridge, it converts the JDBC API into ODBC API and then sends it to the database. When the result is received from the database, it is sent to ODBC API and then to JDBC API. It is platform-dependent because it uses ODBC which depends on the native library of the operating system. In this, JDBC–ODBC driver should be installed in every client system and database must support for ODBC driver. It is easier to use but it gives low performance because it involves the conversion of JDBC method calls to the ODBC method calls.