LAB - 10

Java Programming Lab

Topics Covered

File Handling, JDBC

Syeda Reeha Quasar
14114802719
4C7

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.

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

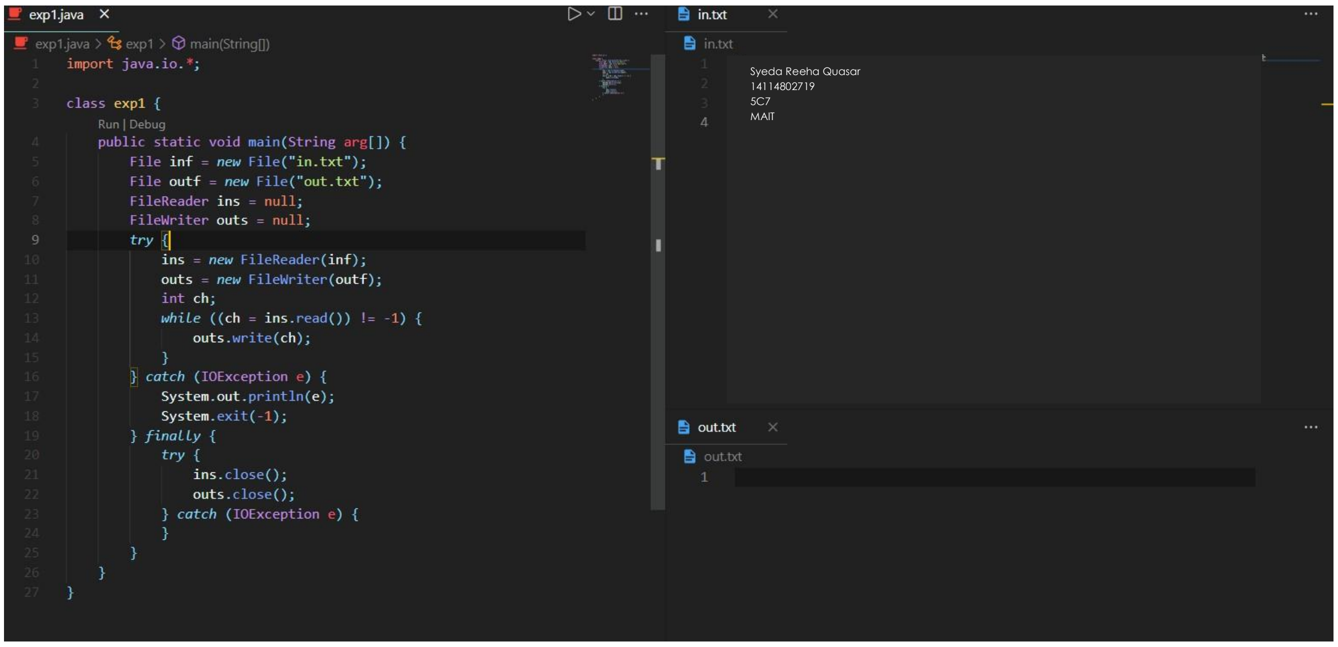
Source Code:

```
package javaapplication1;

import java.io.*;


/**

 *

 * @author reeha

 */
```

```
public class IOFile {
    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 :



```java
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) {
            }
        }
    }
}
```

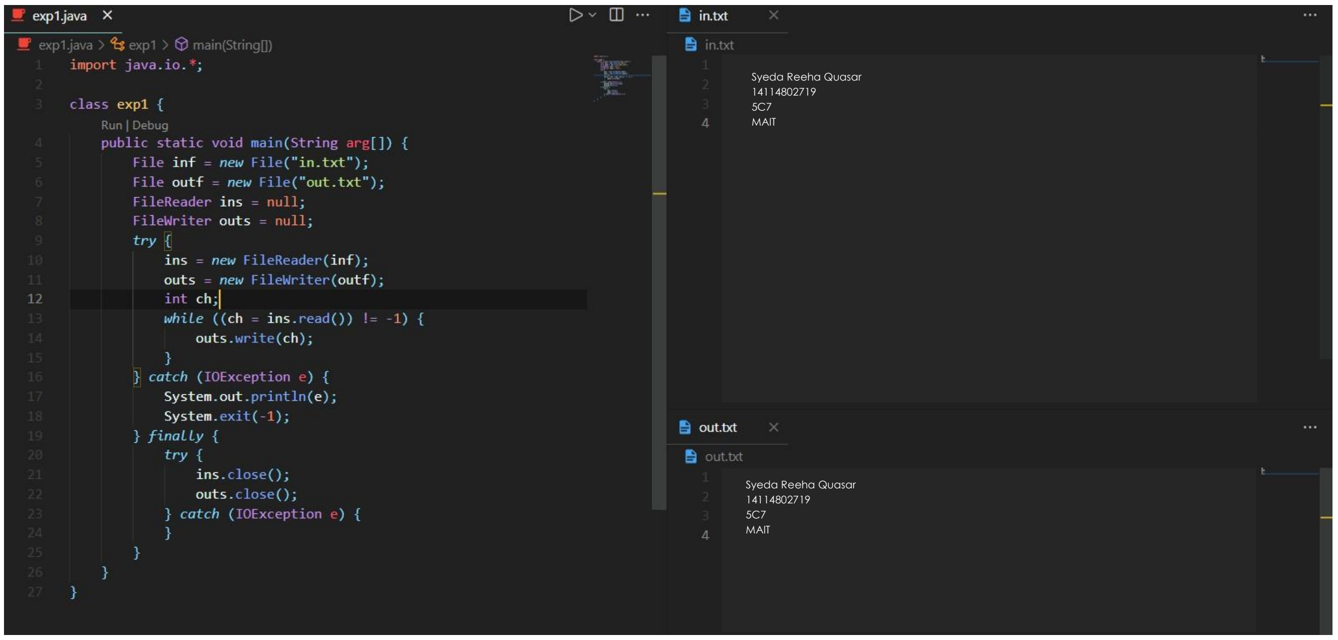
in.txt:

```
Syeda Reeha Quasar
14114802719
5C7
MAIT
```

out.txt:

```
```

After :



```java
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) {
            }
        }
    }
}
```

in.txt:

```
Syeda Reeha Quasar
14114802719
5C7
MAIT
```

out.txt:

```
Syeda Reeha Quasar
14114802719
5C7
MAIT
```

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.

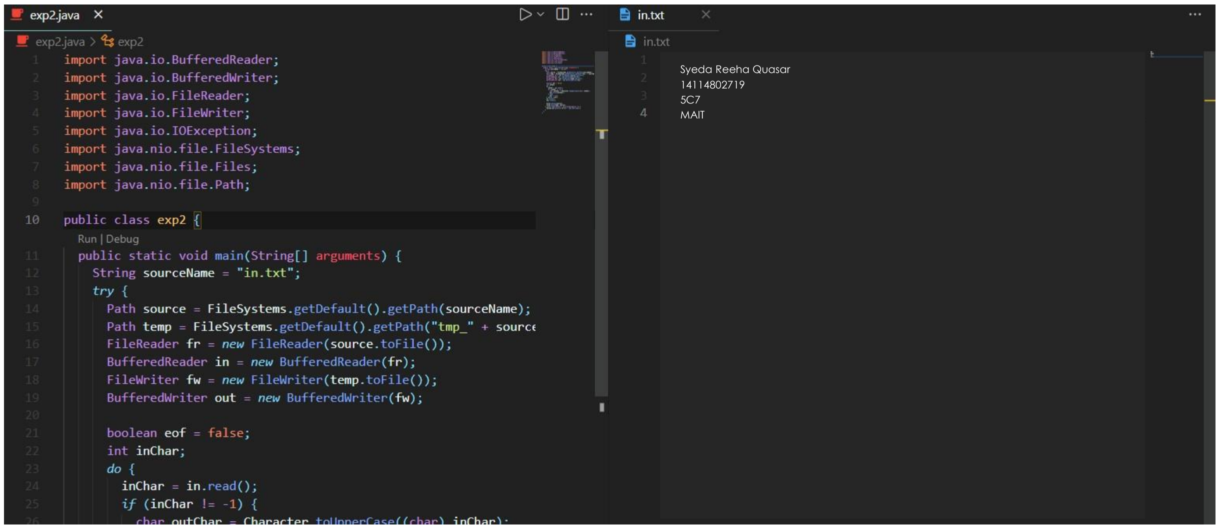
Buffered Writer : Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits Writer class. The buffering characters are used for providing the efficient writing of single arrays, characters, and strings.

Source Code:

```
package javaapplication1;


/**
 *
 * @author reeha
 */


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 :

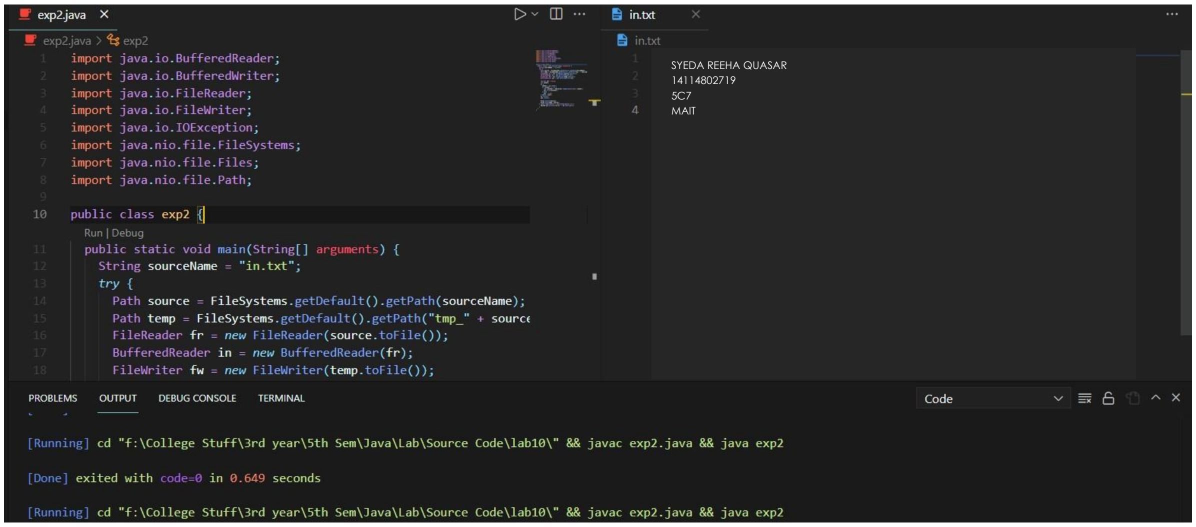


```java
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.nio.file.FileSystems;
import java.nio.file.Files;
import java.nio.file.Path;

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

in.txt

```
Syeda Reeha Quasar
14114802719
5C7
MAIT
```

After :



```java
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 {
  Run | Debug
  public static void main(String[] arguments) {
    String sourceName = "in.txt";
    try {
      Path source = FileSystems.getDefault().getPath(sourceName);
      Path temp = FileSystems.getDefault().getPath("tmp_" + source
      FileReader fr = new FileReader(source.toFile());
      BufferedReader in = new BufferedReader(fr);
      FileWriter fw = new FileWriter(temp.toFile());
```

in.txt

```
SYEDA REEHA QUASAR
14114802719
5C7
MAIT
```

```
[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)

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.

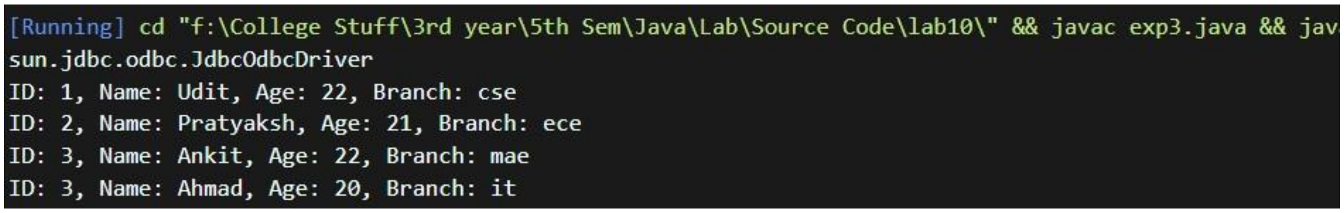
Buffered Writer : Java BufferedWriter class is used to provide buffering for Writer instances. It makes the performance fast. It inherits Writer class. The buffering characters are used for providing the efficient writing of single arrays, characters, and strings.

Source Code:

```
package javaapplication1;

/**
 *
 * @author reeha
 */
```

```java
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

1. 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.

2. 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.

3. 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.

4. 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

5. Describe synchronization in respect to multithreading.

Ans.

With respect to multithreading, synchronization is the capability to control the access of multiple threads to shared resources. Without synchonization, it is possible for one thread to modify a shared variable while another thread is in the process of using or updating same shared variable. This usually leads to significant errors.