Experiment No.7

Title: Implementing the concept of I/O Programming	

Aim: To Study

1. Byte Stream
2. Character stream
3. Buffered Stream
4. Input and Output from command line.

Theory:

I/O Streams

- Byte Streams handle I/O of raw binary data.
- Character Streams handle I/O of character data, automatically handling translation to and from the local character set.
- Buffered Streams optimize input and output by reducing the no. of calls to the native API.
- Scanning and Formatting allows a program to read and write formatted text.
- I/O from the Command Line describes the Standard Streams and the Console object.
- Data Streams handle binary I/O of primitive data type and String values.
- Object Streams handle binary I/O of objects.

File I/O

Java provides a number of classes and methods that allow you to read and write files. In Java, all files are byte-oriented, and Java provides methods to read and write bytes from and to a file. However, Java allows you to wrap a byte-oriented file stream within a character-based object. Two of the most often-used stream classes are FileInputStream and FileOutputStream, which create byte streams linked to files. To open a file, you simply create an object of one of these classes, specifying the name of the file as an argument to the constructor. While both classes support additional, overridden constructors, the following are the forms that we will be using:
FileInputStream(String fileName) throws FileNotFoundException
FileOutputStream(String fileName) throws FileNotFoundException

Here, fileName specifies the name of the file that you want to open. When you create an input stream, if the file does not exist, then FileNotFoundException is thrown. For output streams, if

the file cannot be created, then FileNotFoundException is thrown.When an output file is opened, any preexisting file by the same name is destroyed.When you are done with a file, you should close it by calling close( ). It is defined by both FileInputStream and FileOutputStream, as shown here:

void close( ) throws IOException

To read from a file, you can use a version of read( ) that is defined within FileInputStream. The one that we will use is shown here: int read( ) throws IOException Each time that it is called, it reads a single byte from the file and returns the byte as an integer value. read( ) returns –1 when the end of the file is encountered. It can throw an IOException.
To write to a file, you will use the write( ) method defined by FileOutputStream. Its simplest form is shown here:

void write(int byteval) throws IOException

Basic input and output classes

The java.io package contains a fairly large number of classes that deal with Java input and output.

Most of the classes consist of:

1. Byte streams that are subclasses of InputStream or OutputStream
2. Character streams that are subclasses of Reader and Writer

The Reader and Writer classes read and write 16-bit Unicode characters. InputStream reads 8-bit bytes, while OutputStream writes 8-bit bytes. As their class name suggests, ObjectInputStream and ObjectOutputStream transmit entire objects. ObjectInputStream reads objects; ObjectOutputStream writes objects.

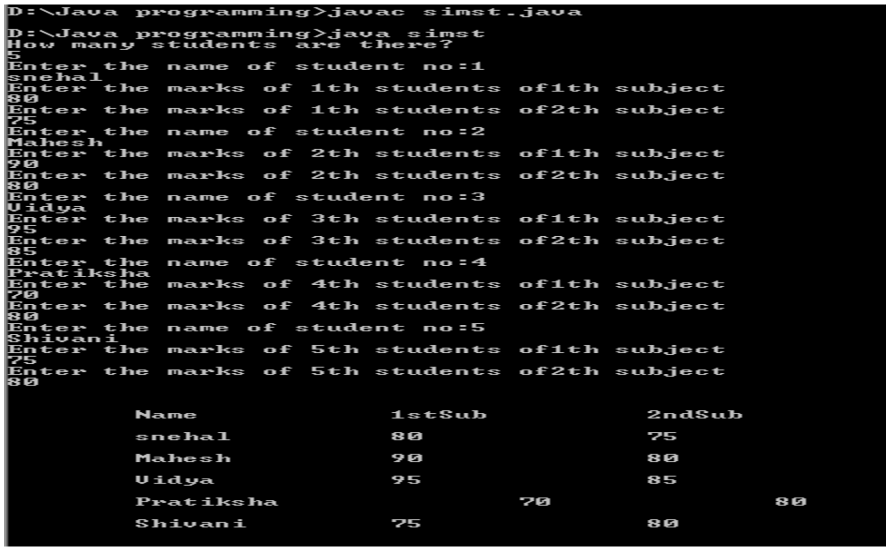
Sample Program:

```
import    java.io.*;
  class simst
  {
          public static void main(String[] args) throws IOException
          {
                  BufferedReader br= new BufferedReader(new InputStreamReader(System.in));
                  String snm[]=new String[100];
                  String a;
                  int n;
                    Int m[][]=new int[10][20];
                    System.out.println("How many students are there?");
                    a=br.readLine();
                   n=Integer.parseInt(a);
                  for(int i=1;i<=n;i++)
                  {
                  System.out.println("Enter     the     name     of     student     no:"+i);
                  snm[i]=br.readLine();
                  for(int j=1;j<=2;j++)
```

```
                    {
                    System.out.println("Enter the marks of "+i+"th students of"+j+"th subject");
                     a=br.readLine();
                    m[i][j]=Integer.parseInt(a);

                    }
        }
                    System.out.println("\n---------------------------------------------------------------");
                    System.out.println("\tName\t\t1stSub\t\t2ndSub");
                    System.out.println("---------------------------------------------------");
                    for(int i=1;i<=n;i++)
                    {
                       System.out.print("\t"+snm[i]);
                       for(int j=1;j<=2;j++)
                     {
                           System.out.print("\t\t"+m[i][j]);
                     }
                           System.out.println("\n");
                    }
                    System.out.println("-----------------------------------");
           }
        }
```

Output:



```
D:\Java programming>javac simst.java

D:\Java programming>java simst
How many students are there?
5
Enter the name of student no:1
snehal
Enter the marks of 1th students of1th subject
80
Enter the marks of 1th students of2th subject
75
Enter the name of student no:2
Mahesh
Enter the marks of 2th students of1th subject
90
Enter the marks of 2th students of2th subject
80
Enter the name of student no:3
Vidya
Enter the marks of 3th students of1th subject
95
Enter the marks of 3th students of2th subject
85
Enter the name of student no:4
Pratiksha
Enter the marks of 4th students of1th subject
70
Enter the marks of 4th students of2th subject
80
Enter the name of student no:5
Shivani
Enter the marks of 5th students of1th subject
75
Enter the marks of 5th students of2th subject
80

        Name            1stSub          2ndSub
        snehal          80              75
        Mahesh          90              80
        Vidya           95              85
        Pratiksha               70              80
        Shivani         75              80
```

Problem Statement:
Take file name as input to your program through command line, if file exists the open and display contents of the file. After displaying contents of file ask user – 1.do you want to add the data at the end of file or 2.replace specified text in file by other text. Based on user’s response, then accept data from user and append it to file. If file in not existing then create a fresh new-file and store user data into it. Also. User should type exit on new line to stop the program. Do this program using Character stream classes.

Conclusion: Thus we studied different Input and output stream.