Experiment No. : 10

Title: Take Student information such as Roll No, Class, age, weight, height, city, phone from user using DataInputStream and and store it in the file using FileOutputStream. Also retrive the data from file using FileInputStream and display the result on the screen.

Objectives: 1. To learn file handling.

Theory:

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 characterbased 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:

To create file-

FileOutputStream fout=new FileOutputStream("studfile.txt",true);

Here studfile is created and that file is attached to FileOutputStream. Then FileOutputStream class object fout writes data to file.

void close( ) throws IOException

To read from a file, you can use a version of **read( )** that is defined within F**ileInputStream**. 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

This method writes the byte specified by *byteval* to the file. Although *byteval* is declared as an integer, only the low-order eight bits are written to the file. If an error occurs during writing, an **IOException** is thrown. The programmer must include extra program statements to determine which event actually occurred. In Java, errors are passed to your program via exceptions, not by values returned by **read( )**. Thus, when **read( )** returns –1, it means only one thing: **the end of the file** has been encountered.

DataStreams:

Data streams support binary I/O of primitive data type values (boolean, char, byte,short, int, long, float, and double) as well as String values. All data streams implement either the DataInput interface or the DataOutput interface. This section focuses on the most widely used implementations of these interfaces, DataInputStream and DataOutputStream. DataInputStream class allows an application to read primitive data from the input stream in a machine-independent way. Java application generally uses the data output stream to write data that can later be read by a data input stream. To read data from keyboard, attach the keyboard to DataInputStream class as-

DataInputStream dis=new DataInputStream(System.in)

Here, System.in represents the keyboard which is linked with DataInputStream object that is dis.

```
ch=(char)dis.read();          //read one character into ch
fout.write(ch);               //write ch into file
```

Key concepts: FileInputStream, FileOutputStream, FileNotFoundException, DataInputStream, DataOutputStream.

Algorithm:

1. Create a class with name StudentInfo.
2. Define a method to get details of the Student from user using DataInputStrean.
3. Create object of FileOutputStream and write the details of the student as a record into the file.
4. Create FileInputStream object, Read and display the details of the student.

Note: Please follow the naming conventions while writing the program.