Aim: Study I/O in java built on streams.

Objectives:

1. Illustration of input and output streams
   (a) W.A.P to create a text file and write data into it.

```
public class filewriter{
    Public static void main (String [] args) throws
                                        IOException {
    Scanner sc = new Scanner (System.in);
    FileWriter f = new FileWriter ("abc.txt");
    SOP ("Enter the number of string");
    int n = sc.nextInt();
    for (int i=0; i<n+1; i++){
        f.write (sc.nextLine() + "\n");
    }
    f.close();
}
```

Explanation - This program will create a text file The 4th line will create a new txt file. The f.write will write into the text file. and f.close will close the file.

2. Illustration of filter stream using Buffered Stream.

(a) WAP to show time efficiency using Buffered stream (I/O) using the buffer.

```
import java.io.*;
public class BufferStream {
Public static void main (String[] args) throws
                                        IOException {
FileInputStream f = new FileInputStream ("C:\\User\\Desktop
                                        \\abc.txt");
BufferedInputStream bf = new BufferedInputStream
                                        (f);
while (bf.available()>0){
    char c=
                        (char)bf.read();
    System.out.print(c);
  }
 }
}
```

(b) WAP to show time efficiency using a buffered stream (I/O) without the buffer.

```
public class File_reader {
public static void main (String[] args) throws
                                        IOException {
FileInputStream f = new FileInputStream ("C:\\User\\
                                        Desktop\\abc.txt");
long st = System.currentTimeMillis();
int i=0;
while ((i=f.read())!=-1)
 {
```

```
        System.out.print((char)i);
    }
    System.out.println();
    long et = System.currentTimeMillis();
    long tot = et - st;
    System.out.print("Total time:" +tot);
    Systm.out.print("start time:" +st);
    System.out.println("End time:" +et);
    f.close();
}
}
```

Exercises:-

1. Explain the operation of the write() method of output stream.

Sol:

The 'write()' method of 'outputStream' is used to write a single byte of data to an output stream.
It takes an integer as input, but only the lowest 8 bits (1 byte) are written.
It may throw an 'IOException' if there's an error during the write operation, and it's typically used in a loop to write multiple bytes sequentially to the stream.

Q2. Mention the fun of the following subclasses of outputStream:

(a) FileOutputStream:
Writes raw bytes to a file.
Functions:
- write (int b) = writes the specified byte to the output stream.
- write (byte[]b) = writes the specified byte array to the output stream.
- flush(): Flushes the output stream, meaning that any buffered output bytes are written to the file.
- close(): Closes the output stream.

(b) Telnet output stream:
Used in Telnet client implementation for remote terminal connections.
Function :-
- write (int b)
- write (byte[]b)
- flush()
- close()

(C) ByteArrayOutputStream:
Writes data to an in memory byte array.
Function:-
- write (int b)
- write (byte[]b)
- toByteArray()
- toString()
- close()

Q3- Compare and contrast the bufferedInputStream and bufferedOutputStream.

Soln

BufferedInputStream:

- Purpose: Improves reading performace from an input stream.
- Functionality: Reads data into an internal buffer.
- Methods: read(), read(byte[]b), reset(),
- Use case: Reading from slow sources.

BufferedOutputStream:

- purpose: Improves writing performance to an output stream.
- Functionality: Writes data to an internal buffer and flushes when necessary.
- Method: write(int b), close(), flush().

Common characteristics:

- Both use internal buffer for improved I/O performance.
- Can be chained with other stream classes.

Q4. W.A.C to read the data from a file in byte and char format.

Sol^n

```
public class readinByte {
public static void main(String) throws IOException {
FileInputStream f = new FileInputStream("C:\\user
SOP("Reading data in byte format"); \\abc.txt");
int data;
while ((data = f.read()) != -1) {
        System.out.println(data);
    }
  }
}
public class readinchar {
public static void main() throw IOException {
FileReader f = new FileReader ("C:\\user\\abc.txt");
SOP("Reading data in character format");
int data;
while ((data = f.read()) != -1) {
        System.out.println((char)data);
    }
}
}
```

Q5- Write a code to append a string to an existing text file.

Sol^n

```
public class main {
    public static void main( ) throws IOException {
FileWriter f = new FileWriter("C:\\user\\abc.txt",
                                        true) ;
 f.write("Hello"); // content to append.
 SOP("Appended successfully");
 f.close();
  }
}
```

FileWriter is used with the constructor parameter 'true' to enable append mode.