22

IO STREAMS

Stream:-

A Stream represents flow of data from one place to another.

There are two types of streams.
1.InputStreams:- They read(or)accept data
2.OutputStreams:-They send (or) write data to some other place.

Here another 2 classifications are also there.
1.byte streams:-They transport data in the form of bits and bytes.
2.text streams:-They transport data in the form of characters.

All streams are represented by classes in java.io package.

Byte streams:-

These classes are derived from the abstract classes:
InputStream and OutputStream

ex:-
ByteArrayInputStream:- It uses a buffer to read bytes.(or)read bytes from array.
ByteArrayOutputStream:-It writes data into byte array.

FileInputStream :-Reads data from file.

FileOutputStream:-Write data into file.

FilterInputStream:-Reads data from another inputStream.

FilterOutputStream:-Sends data to another outputStream.

PipedInputStream:-This should be connected to piped output stream.It reads data that is send to PipedOutputStream.
PipedOutputStream:-Sends data from one program to another program.

ObjectInputStream:-Reads objects from a file.

ObjectOutputStream:-Writes objects into a file.

BufferedInputStream:-Reads a buffer full of data from another inputStream.
BufferedOutputStream:-Writes a buffer full of data(bytes) into another outputStream.
DataInputStream:-Reads primitive data types from another inputStream.

DataOutputStream:-writes primitive datatypes to another outputStream.

PrintStream:-Sends bytes to another Stream.

TextStreams:-

These classes are derived from the abstract classes:
Reader and Writer.

ex:-
BufferedReader:-Reads a buffer full of characters from another inputStream.
BufferedWriter:-Writes a buffer full of characters into another outputStream.
CharArrayReader:-Reads a character array.

CharArrayWriter:-Writes a character array.
InputStreamReader:-It reads bytes and decodes them into character.

InputStreamWriter:-It converts characters into bytes and writes them to another stream.
FileReader:-Reads characters from a file.

FileWriter:-Writes characters into a file.

PrintWriter:-Sends text to another Stream.

FILE I/O

1. **Introduction**
2. **File**
3. **FileWriter**
4. **FileReader**
5. **BufferedWriter**
6. **BufferedReader**
7. **PrintWriter**

Introduction:

In 1995 when java is invented Database concepts are not much popular so we are storing the Data in Files.we may store data using variable and arrays is not a permanent storage this is temporary storage only.To Store Data Permanently we use Files.

File:

File f=new File (“abc.txt”);

This line won’t create any physical file first it will check is there any physical file named with abc.txt is available or not if it is available then f simply refers that file. If it is not available then we are just creating java file object to represent the name abc.txt

Program 1:

```
import java.io.*;
class Test1
{
public static void main(String[] args)throws Exception
{
File f=new File("klutest.txt");
System.out.println(f.exists());
f.createNewFile();
System.out.println(f.exists());

}
}
```

Output:

```
D:\KLUniversity\Files>javac Test1.java
D:\KLUniversity\Files>java Test1
false
true

D:\KLUniversity\Files>java Test1
true
true
```

We can use java file object to represent directory also.
Note:in UNIX everything is treated as file java file IO concepts is implemented based on UNIX operating System.Hence java file object can be used to represent both files and directories.
Program 2:

```
import java.io.*;
class Test2
{
public static void main(String[] args)throws Exception
{
File f=new File("yellaswamy");
System.out.println(f.exists());
f.mkdir();
System.out.println(f.exists());

}
}
```

Output:

```
D:\KLUniversity\Files>javac Test2.java
D:\KLUniversity\Files>java Test2
false
true

D:\KLUniversity\Files>java Test2
true
true
```

File class Constructors:

1. File f=new File(String name) ;

creates a java file object to represent name of the file or directory in current working directory.

2. File f=new File(String subdir,String name);
   Creates a Java file object to represent name of the file or directory present in specified sub directory.
3. File f=new File(File Substr,String name);

Program3:Write code to create a file named with abc.txt in current working directory.

```
import java.io.*;
class Test1
{
public static void main(String[] args)throws Exception
{
File f=new File("abc.txt");
System.out.println(f.exists());
f.createNewFile();
System.out.println(f.exists());

}
}
```

Output:

```
D:\KLUniversity\Files>javac Test1.java
D:\KLUniversity\Files>java Test1
false
true

D:\KLUniversity\Files>
```

Program4: Write code to create a directory named with klu123 in current working directory and create file named with demo.txt in that directory.

```
import java.io.*;
class Test3
{
public static void main(String[] args)throws Exception
{
File f=new File("klu123");
f.mkdir();
File f1=new File("klu123","demo.txt");
System.out.println(f.exists());
f.createNewFile();
System.out.println(f.exists());
```

```
        }
        }
        Output:

        D:\KLUniversity\Files>javac Test3.java

        D:\KLUniversity\Files>java Test3
        true
        true
```

Program5:Write code to create a file named with abc.txt in D://KLUniversity//Files//klutestdir directory.

```
import java.io.*;
class Test4
{
        public static void main(String args[])throws Exception
        {
                File myfile=new File("D://KLUniversity//Files//klutestdir","abc.txt");
                myfile.createNewFile();
        }
}
```

output:

```
D:\KLUniversity\Files>java Test4

D:\KLUniversity\Files>cd klutestdir
D:.
   abc.txt

No subfolders exist
```

Directory Structure:

```
D:
|
├──klutestdir
          abc.txt
```

Note: Assume that D://KLUniversity//Files//klutestdir is already available in our system.

Important Methods Present in File class:

- **public boolean exists();**
  - Returns true if the specified file or directory available
- **public boolean createNewFile();**
  - First this method will check whether the specified file is already available or not. If it is already available then this methods returns False with out creating any physical file.
  - If the File is not already available then this method creates new File and returns true.
- **public boolean mkdir();**
  - Same as **createNewFile();**
- **public boolean isFile();**
  - Returns true if the specified file Object pointing to Physical file.
- **public boolean isDirectory();**
  - Returns true if pointing to directory
- public java.lang.String[] list();
  - This method returns the names of all files and sub directories present in specified directory.
- **public long length();**
  - Returns no of characters present in the specified file.
- **public boolean delete();**
  - To delete specified file or directory

Program6: Write a Program to display all names and directories present in D://KLUniversity

```
import java.io.*;
class Test5
{
        public static void main(String args[])throws Exception
        {
                int count=0;
                File myfile=new File("D://KLUniversity");
                String[] s=myfile.list();
                for (String s1:s)
                {
                        count++;
                        System.out.println(s1);
                }
                System.out.println(count);
        }
}
```

Output:

```
D:\KLUniversity\Files>javac Test5.java
```

```
D:\KLUniversity\Files>java Test5
FILE IO.docx
Files
InductionProgram
MailsMSRPrasad
Phd2016
~$ILE IO.docx
6
```

TO DISPLAY ONLY FILE NAMES:

```
import java.io.*;
class Test6
{
	public static void main(String args[])throws Exception
	{
		int count=0;
		File myfile=new File("D://KLUniversity");
		String[] s=myfile.list();
		for (String s1:s)
		{
			File f1=new File(myfile,s1);
			if(f1.isFile())
			{
			count++;
			System.out.println(s1);
			}
		}
		System.out.println("The Total No of Files="+count);
	}
}
```

Output:

```
D:\KLUniversity\Files>javac Test6.java
D:\KLUniversity\Files>java Test6
FILE IO.docx
~$ILE IO.docx
The Total No of Files=2
```

TO DISPLAY ONLY Directory NAMES:

```
import java.io.*;
class Test7
{
	public static void main(String args[])throws Exception
	{
		int count=0;
		File myfile=new File("D://KLUniversity");
		String[] s=myfile.list();
		for (String s1:s)
		{
```

```
                        File f1=new File(myfile,s1);
                        if(f1.isDirectory())
                        {
                        count++;
                        System.out.println(s1);
                        }
                }
                System.out.println("The Total No of Files="+count);
        }
}
```

Output:

```
D:\KLUniversity\Files>javac Test7.java
D:\KLUniversity\Files>java Test7
Files
InductionProgram
MailsMSRPrasad
Phd2016
The Total No of Files=4
```

FileWriter:

We can use FileWriter to write character data to the file.

Constructors:

1. FileWriter fw=new FileWriter(String fname);
2. FileWriter fw=new FileWriter(File f);

   The above File Writers meant for overriding of existing data. Instead of overriding we want append operation then we have to create FileWriter by using the following Constructors.

3. FileWriter fw=new FileWriter(String fname,boolean append);
4. FileWriter fw=new FileWriter(File f,boolean append);

Note: If the specified file is not already available then all the above constructors will create that file.

Methods in FileWriter:

- write(int ch)
  - To write a single character

    Example:

    ```
    Fw.write('a');
    Fw.write(97);
    ```
- write(char[] ch)
  - To write array of characters
- write(String s)
  - To write string to a file
- flush()

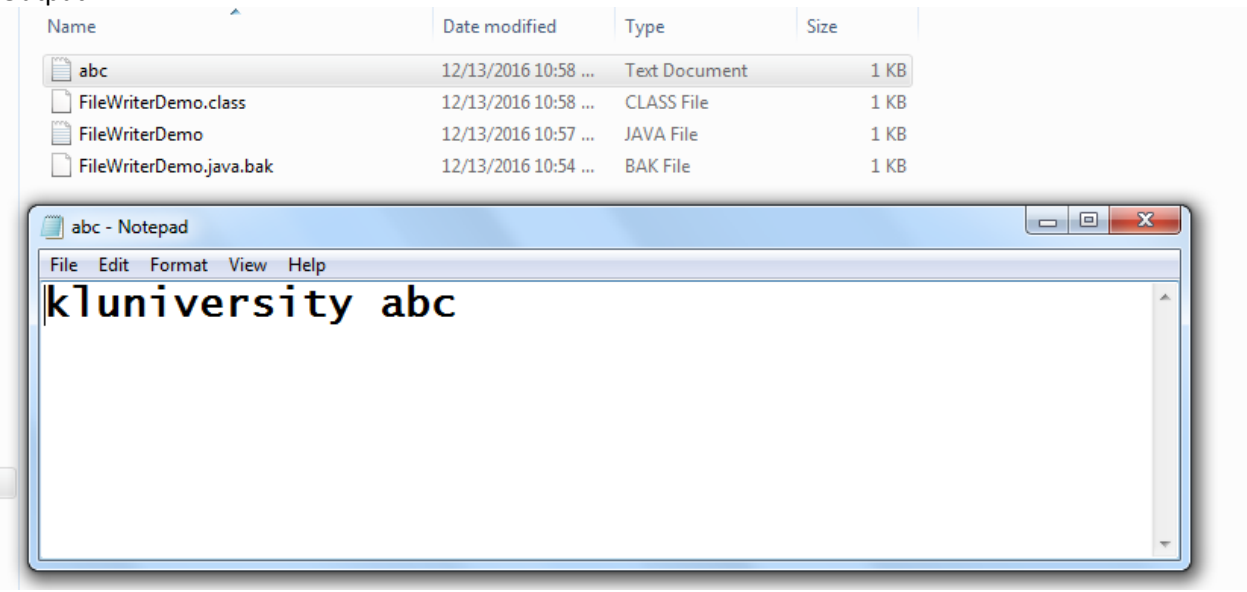
- To give the guarantee that total data including last character will be written to the file.

- close()
  - To close the writer

Example:

```
import java.io.*;
class FileWriterDemo
{
public static void main(String[] args) throws IOException
{
        FileWriter fw=new FileWriter("abc.txt");
        fw.write(107);//write single character
        fw.write("luniversity");
        fw.write('\n');
        char[] ch1={'a','b','c'};
        fw.write(ch1);
        fw.write('\n');
        fw.flush();
        fw.close();

}
}
```

Output:



In the above program FileWriter can perform overriding of existing data.instead of overriding if want append operation then we have to create FileWriter object as follows.

FileWrite fw=new FileWriter(“abc.txt”,true);

Note: The main problem with FileWriter is we have to insert line separator (\n) manually which is varied from system to system.it is the difficulty to the programmer.we can solve this problem by using BufferedWriter and PrintWriter classes.

FileReader:

We can use FileReader to read character data from file.

Constructors:

1. FileReader fr=new FileReader(String fname);
2. FileReader fr=new FileReader(File f);

Methods in FileReader:

- int read();
  - it attempts to read next character from the file and returns its Unicode value if the next character is not available then this methods returns -1.
  - As this method returns Unicode value(int value),at the time of printing we have to perform type casting.

Ex:

```
Int i=fr.read()
System.out.println(i);
```

Output:

Unicode value will be printed

Example:

```
import java.io.*;
class FileReaderEx
{
public static void main(String... args)throws IOException
{
FileReader fr=new FileReader("abc.txt");
int i=fr.read();
while(i!=-1)
{
System.out.print((char)i);
}
}
}
```

Output:

KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK KKKKK

KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
KKKKK
KKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKKK
KKKKK

- int read(char[] ch)
  - it attempts to read enough characters from the file into char array and returns number of characters copied from the file.

Example:

```
import java.io.*;
class FileReaderEx2
{
public static void main(String... args)throws IOException
{
FileReader fr=new FileReader("abc.txt");
char[] ch=new char[10];
fr.read(ch);
System.out.println(ch);
}
}
```

Output:

```
D:\KLUniversity\Files\FileReader>javac FileReaderEx2.java
D:\KLUniversity\Files\FileReader>java FileReaderEx2
KLUniversi
```

```
import java.io.*;
class FileReaderEx3
{
public static void main(String... args)throws IOException
{
File f=new File("abc.txt");
char[] ch=new char[(int)f.length()];
FileReader fr=new FileReader(f);

fr.read(ch);
for(char ch1:ch)
	{

System.out.println(ch);
	}
}
}
```

Output:

```
D:\KLUniversity\Files\FileReader>javac FileReaderEx3.java

D:\KLUniversity\Files\FileReader>java FileReaderEx3
        KLUniversity
        Yellaswamy
        KLUniversity
        Yellaswamy
        KLUniversity
        Yellaswamy
        KLUniversity
        Yellaswamy
        KLUniversity
        Yellaswamy
        KLUniversity
```

- void close()

Usage of FileReader and FileWriter is not recommended because:

- while writing Data by FileWriter we have to Insert Line Separator Manually which varied from System to System.It is Difficult to the programmer
- while reading data by FileReader we have to read Character by Character instead of line by line which is not convenient to the programmer.
- To overcome these limitations we should go for BufferedWriter and BufferedReader concepts

BufferedWriter:
By using BufferedWriter object we can write character data to the file.
Constructors:

```
BufferedWriter bw=new BufferedWriter(Writer w);
BufferedWriter bw=new BufferedWriter(Writer w,int buffersize);
```

Note:
BufferedWriter never communicates directly with the file it should communicates via some writer object.

Methods:

1. write(int ch);
2. write(char[] ch);
3. write(String s);
4. flush();
5. close();
6. newline();

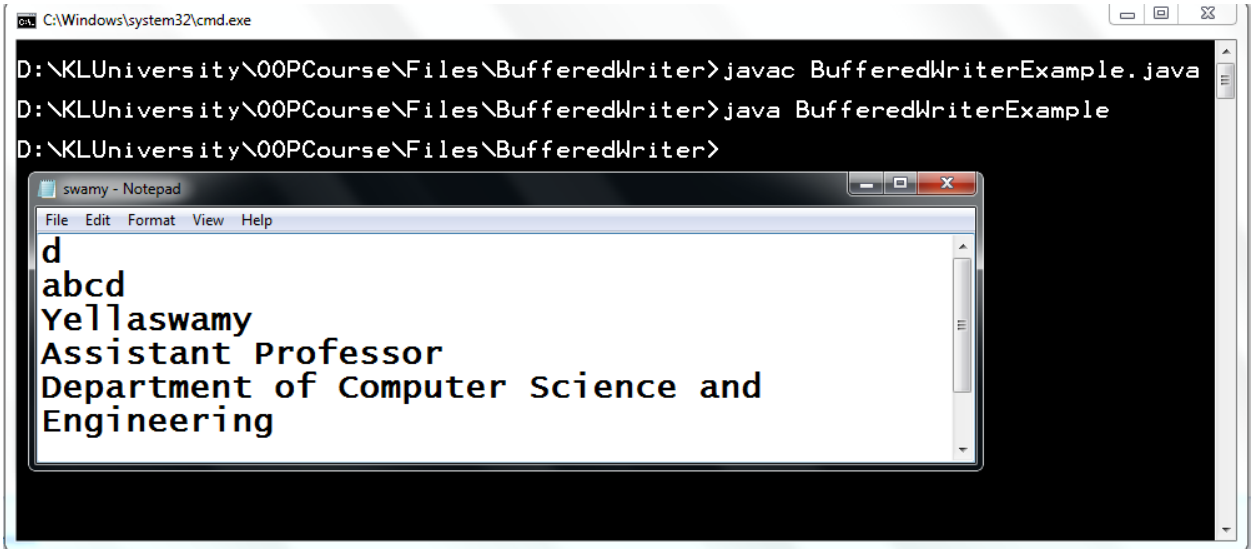
inserting a new line character to the file.

Example:

Program for Buffered Writer

```
import java.io.*;

class  BufferedWriterExample
{
        public static void main(String[] args) throws IOException
        {
                FileWriter fw=new FileWriter("swamy.txt");
                BufferedWriter bw=new BufferedWriter(fw);
                bw.write(100);
                bw.newLine();
                char[] ch={'a','b','c','d'};
                bw.write(ch);
                bw.newLine();
                bw.write("Yellaswamy");
                bw.newLine();
                bw.write("Assistant Professor");
                bw.newLine();
                bw.write("Department of Computer Science and Engineering");
                bw.newLine();

                bw.write("K L University");
                bw.newLine();


                bw.flush();
                bw.close();
        }
}
```

Output:



Note:
When ever we are closing BufferedWriter automatically underlying writer will be closed and we are not close explicitly.

Constructors:

BufferedReader br=new BufferedReader(Reader r);
BufferedReader br=new BufferedReader(Reader r,int buffersize);

Note:
BufferedReader can not communicate directly with the File it should communicate via some Reader object.
The main advantage of BufferedReader over FileReader is We can read data line by line instead of character by character.

Methods:

1. **int** read();
2. int read(char[] ch);
3. String readLine();
   It attempts to read next line and return it,form the file.if the next line is not available then this methods return null.
4. **void close()**

Example:

```
//Program for BufferedReader
import java.io.*;
class BufferedReaderExample
{
public static void main(String[] args)throws IOException
{
FileReader fr=new FileReader("swamy.txt");
BufferedReader br=new BufferedReader(fr);
String line=br.readLine();
```

```
while(line!=null)
{
System.out.println(line);
line=br.readLine();
}
br.close();
}
}
```

Output:

```
D:\KLUniversity\OOPCourse\Files\BufferedReader>javac BufferedReaderExample.java

D:\KLUniversity\OOPCourse\Files\BufferedReader>java BufferedReaderExample
d
abcd
Yellaswamy
Assistant Professor
Department of Computer Science and Engineering
K L University
```

PrintWriter:

1. This is the most enhanced Writer to write text data to the file.
2. By using FileWriter and BufferedWriter we can write only character data to the File but using PrintWriter we can write any type of data to the file.

Constructors:

```
PrintWriter pw=new PrintWriter(String name);
PrintWriter pw=new PrintWriter(File f);
PrintWriter pw=new PrintWriter(Writer w);
```

PrintWriter can communicate either directly to the file or via some Writer object also.

Methods:

1. write(int ch);
2. write(char[] ch);
3. write(String s);
4. flush();
5. close();

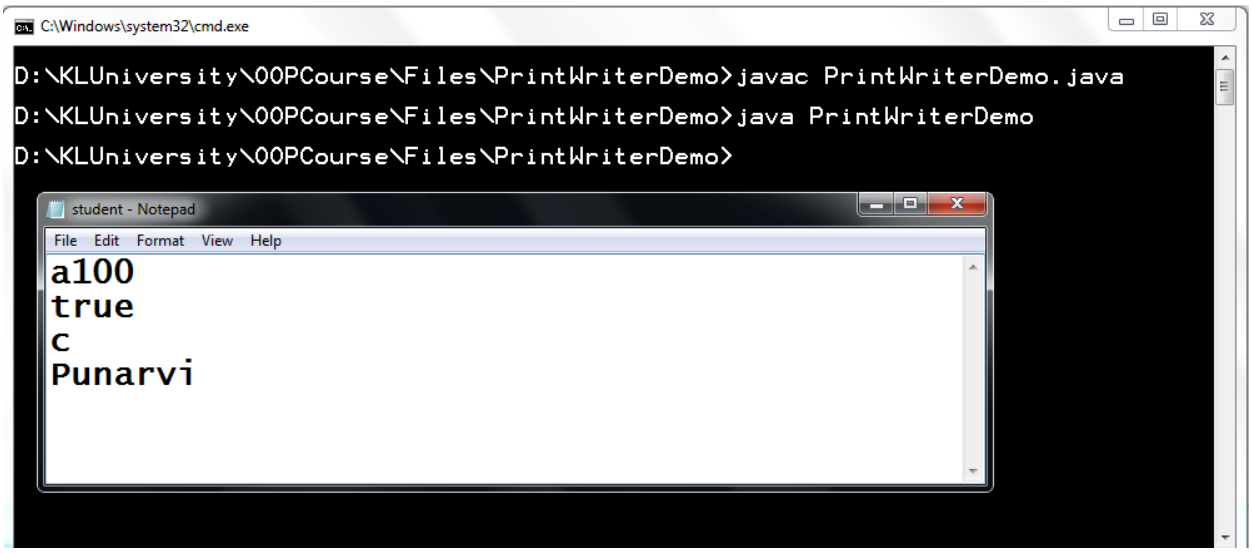
6. print(char ch);
7. print(int i);
8. print(double d);
9. print(boolean b);
10. print(String s);
11. println(char ch);
12. println(int i);
13. println(double d);

14. println(boolean b);
15. println(String s)

Example:
Program for PrinWriter Demo

```
//Program for PrintWriter Usage
import java.io.*;
class PrintWriterDemo
{
public static void main(String[] args)throws IOException
{
FileWriter fw=new FileWriter("student.txt");
PrintWriter out=new PrintWriter(fw);
out.write(97);
out.println(100);
out.println(true);
out.println('c');
out.println("Punarvi");
out.flush();
out.close();

}
}
```

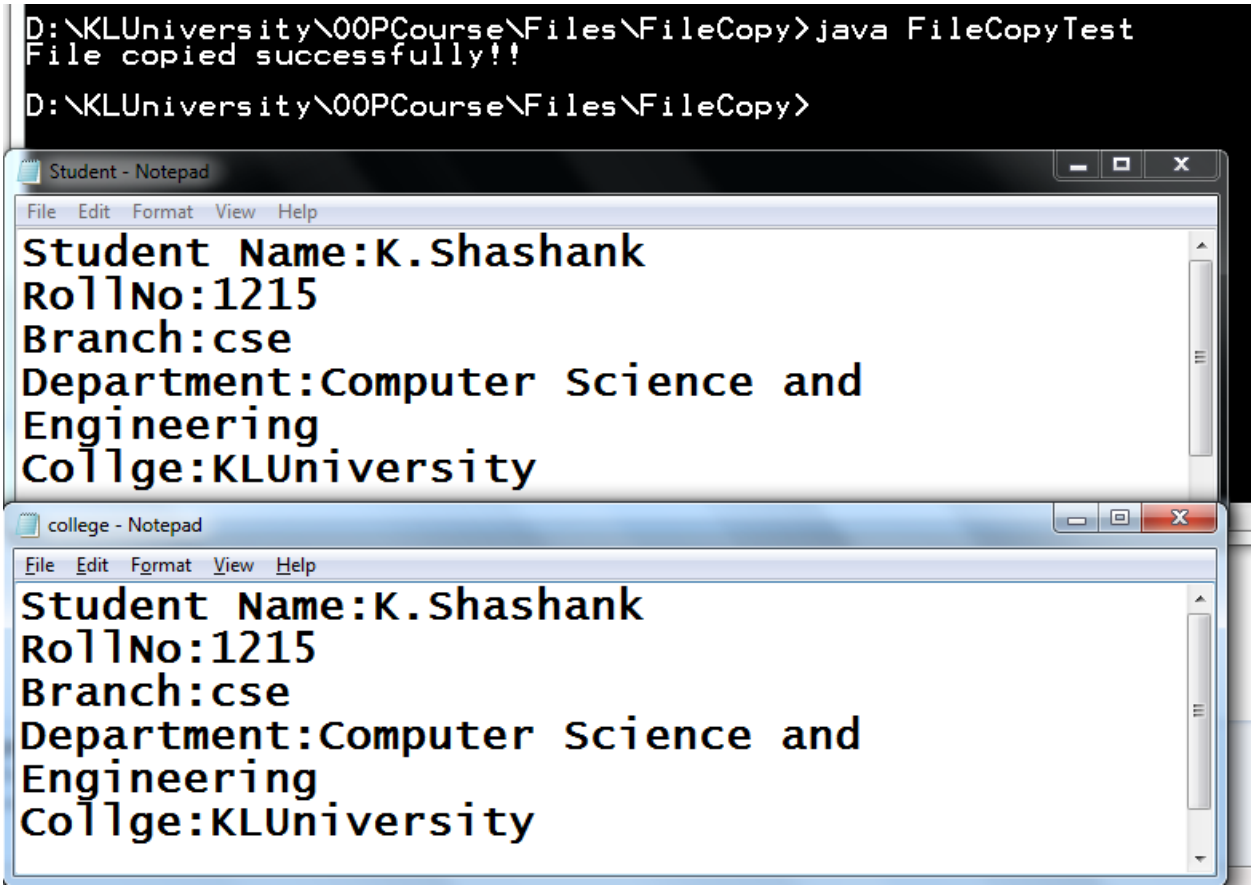
Output:



Example:
//Program for file copy(copying from "student.txt" file to "college.txt")

```
import java.io.*;

public class FileCopyTest
{
   public static void main(String[] args)
   {
        FileInputStream fis = null;
                FileOutputStream fos = null;

        try
                        {
           File inputfile =new File("student.txt");
           File outputfile =new File("college.txt");

           fis = new FileInputStream(inputfile);
           fos= new FileOutputStream(outputfile);

           byte[] buffer = new byte[1024];

           int length;
           /*copying the contents from input stream to
            * output stream using read and write methods
            */
           while ((length = fis.read(buffer)) > 0)
                        {
                fos.write(buffer, 0, length);
           }

           //Closing the input/output file streams
           fis.close();
           fos.close();

           System.out.println("File copied successfully!!");

        }catch(IOException ioe){
                ioe.printStackTrace();
         }
   }
}
```

Output:




```
//program to Read data from keyboard using Console class
import java.io.*;
class ConsoleInputTest
{
	public static void main(String... args)
	{

	Console c=System.console();
	int n;
	System.out.println("Enter a Number");
	n=Integer.parseInt(c.readLine());
	System.out.println("The Given Number:"+n);
	}

}
```

Output:

D:\KLUniversity\OOPCourse\ConsoleInput>java ConsoleInputTest

```
Enter a Number
5
The Given Number:5
```

Java Console and File Input/Output Cheat Sheet

Console Output

```
System.out.print("Hello ");
System.out.println("world");
```

Console Input

```
BufferedReader in = new BufferedReader(new InputStreamReader(System.in));
String text = in.readLine();
```

File Output

```
PrintWriter out = new PrintWriter(new FileWriter("K:\\location\\outputfile.txt")));
out.print("Hello ");
out.println("world");
out.close();
```

File Input

```
BufferedReader in = new BufferedReader(new FileReader("K:\\location\\inputfile.txt"));
String text = in.readLine();
in.close();
```

Converting input data

```
String text  = in.readLine();
int x = Integer.parseInt(text);
double y = Double.parseDouble(text);
```

Reading until EOF

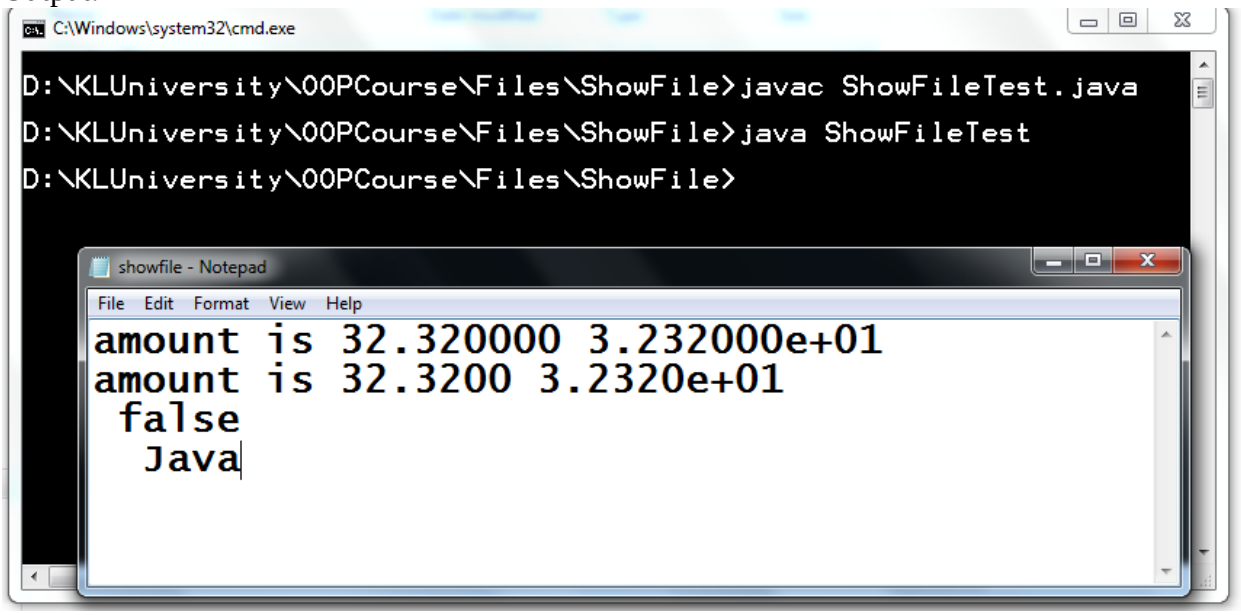
```
while (in.ready()) {
  text = in.readLine();
  System.out.println(text);
}
```

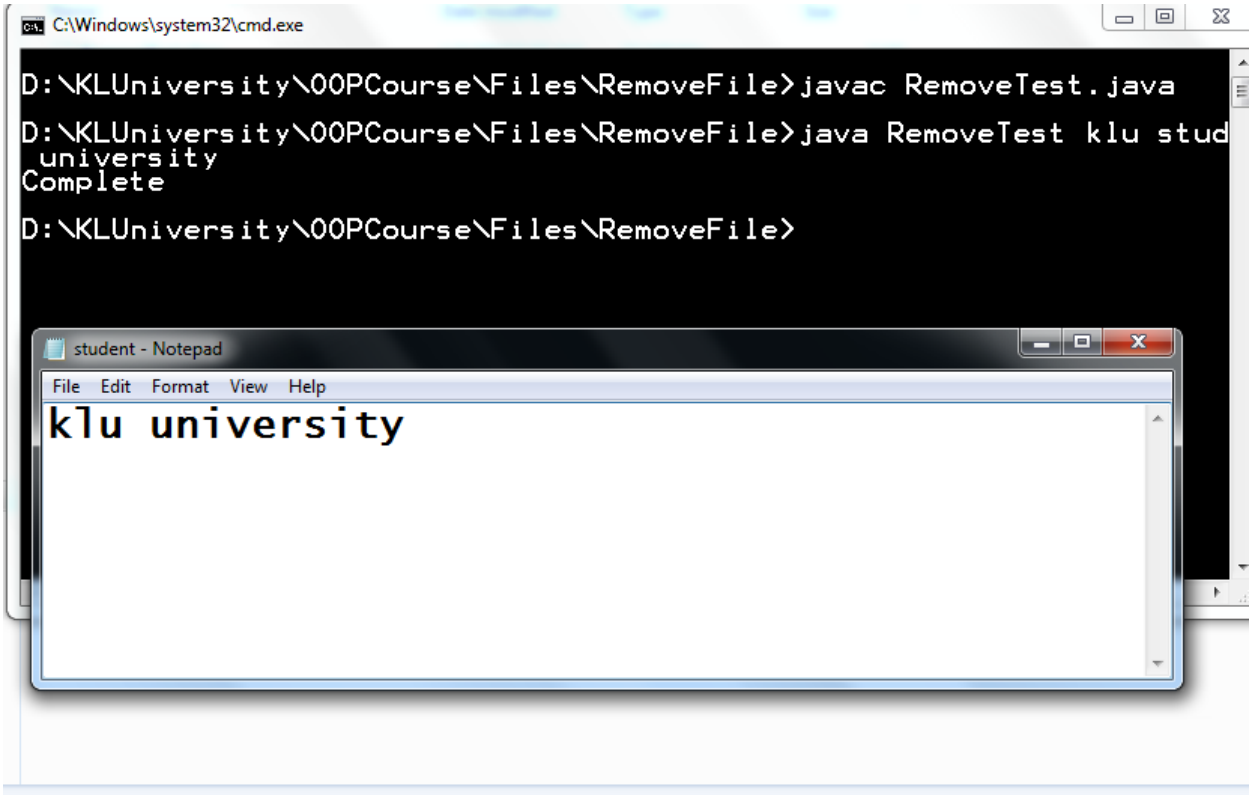
Example:

```
//Show the contents of the file showfile.txt after the following program is executed.
public class ShowFileTest
```

```
{
public static void main(String[] args) throws Exception
{
java.io.PrintWriter output = new java.io.PrintWriter("showfile.txt");
output.printf("amount is %f %e\r\n", 32.32, 32.32);
output.printf("amount is %5.4f %5.4e\r\n", 32.32, 32.32);
output.printf("%6b\r\n", (1 > 2));
output.printf("%6s\r\n", "Java");
output.close();
}
}
```

Output:




C:\Windows\system32\cmd.exe
D:\KLUniversity\OOPCourse\Files\RemoveFile>javac RemoveTest.java
D:\KLUniversity\OOPCourse\Files\RemoveFile>java RemoveTest klu stud
university
Complete
D:\KLUniversity\OOPCourse\Files\RemoveFile>
student - Notepad
File Edit Format View Help
klu university

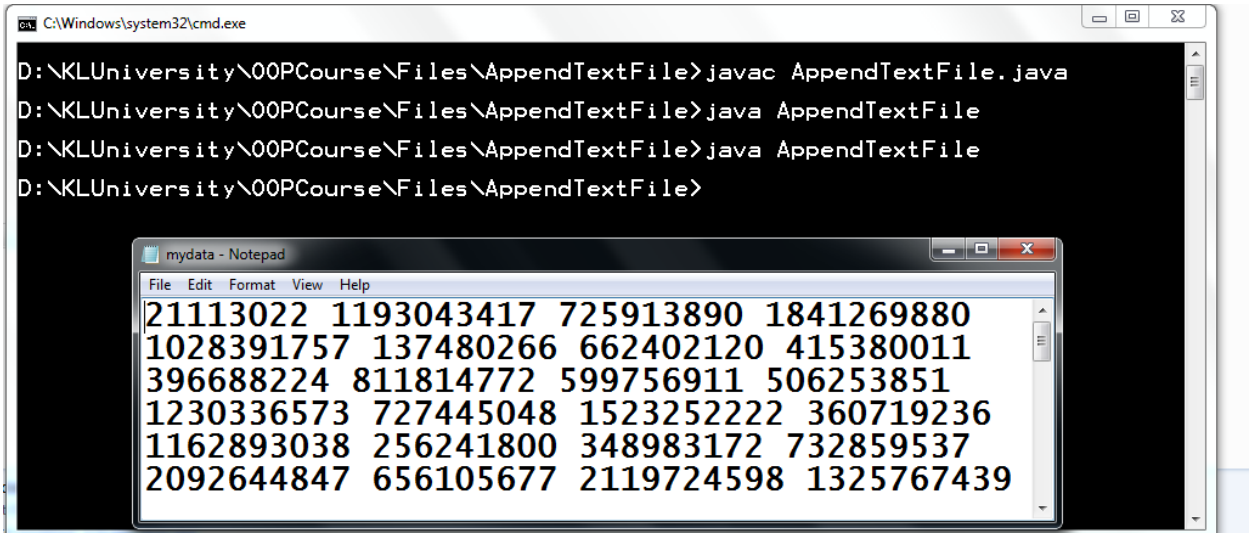
Example:

(Create a text file)

Write a program to create a file named mydata.txt if it does not exist. Append new data to it if it already exists. Write 100 integers created randomly into the file using text I/O. Integers are separated by a space.

```
import java.io.*;

public class AppendTextFile
{

   public static void main(String[] args) throws IOException
   {

      File file = new File("mydata.txt");

      boolean append = file.exists();

      try(PrintWriter out= new PrintWriter(new BufferedOutputStream(new
FileOutputStream(file, append))))
                                                                  {


         for (int i = 0; i < 100; i++)
                                                                        {
            out.write((int) (Math.random() * (Integer.MAX_VALUE + 1L)) + " ");
         }
      }

   }
}
```

OutPut:


C:\Windows\system32\cmd.exe
D:\KLUniversity\OOPCourse\Files\AppendTextFile>javac AppendTextFile.java
D:\KLUniversity\OOPCourse\Files\AppendTextFile>java AppendTextFile
D:\KLUniversity\OOPCourse\Files\AppendTextFile>java AppendTextFile
D:\KLUniversity\OOPCourse\Files\AppendTextFile>
mydata - Notepad
File Edit Format View Help
21113022 1193043417 725913890 1841269880
1028391757 137480266 662402120 415380011
396688224 811814772 599756911 506253851
1230336573 727445048 1523252222 360719236
1162893038 256241800 348983172 732859537
2092644847 656105677 2119724598 1325767439