(Java programming - 22412)

Unit VI : Managing Input/Output Files in Java.

(Weightage - 08 marks)

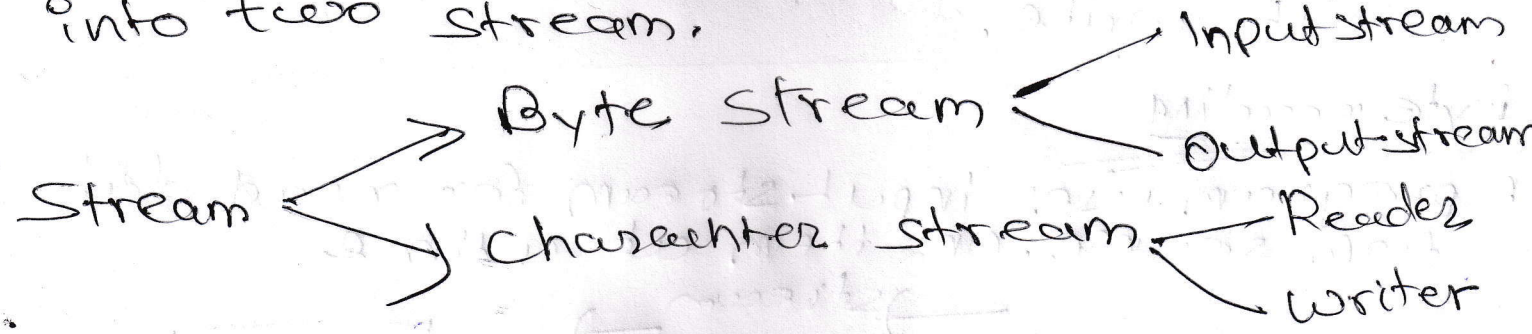
Pranjal Save (5

File Handling (file exists, not exists, Operations.....)

File handling defines how we can read and write data on a file. Java IO package contains all the classes through which we can perform all input & output operations.

Stream

Stream is a sequence of data. On the basis of java.io package all the classes divided into two stream.



File handling methods

1. canRead()
2. canWrite()
3. createNewFile()
4. Delete()
5. Exists()
6. length.
7. getName()
8. List()
9. Read()
10. write()
11. mkdir()
12. RenameTo().

File handling class

1. File - super class all file class
2. File Reader - data read
3. File writer - data write
4. File Input-stream - data read in form of Byte
5. File Output-stream - data write in file in form of Byte
6. Buffered Input-stream - read
7. Buffered output stream - write

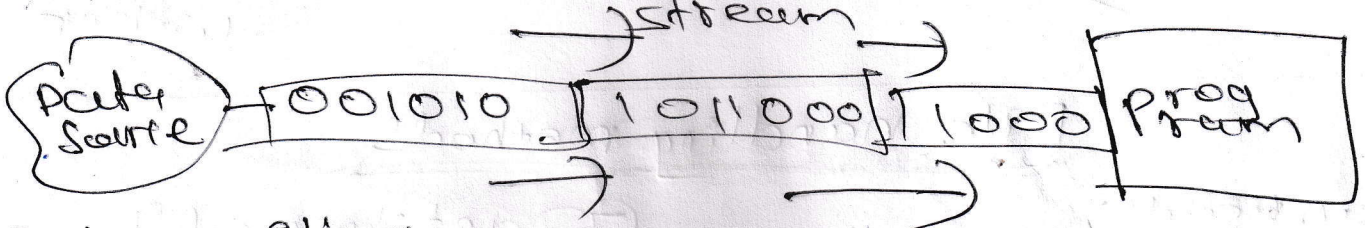
Buffered - Temporary memory To perform I/O operation in it.

Basic operations in file

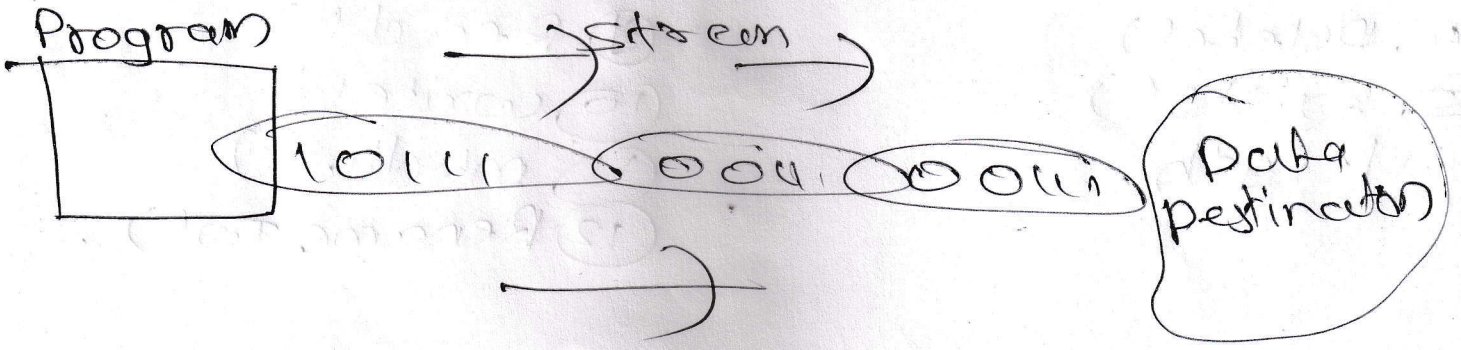
1. create a file
2. Get file information
3. read
4. write.

Byte reading

A program uses input-stream for read data from source. One item at a time



Byte writing - uses output stream.



Byte Stream classes

Byte streams as the name implies handle reading and writing of bytes.

They are abstracted by classes InputStream and OutputStream respectively.

Byte Stream classes

- InputStream
- OutputStream

methods

- int available()
- void close()
- int read()
- int read(byte buffer[])
- int read(byte buffer[], int offset, int nBytes)
- long skip(long nBytes)
- protected void finalize()

methods

- void close()
- void flush()
- void write(int b)
- void write(byte buffer[])

Charecter Stream classes

- Reader
- writer.

methods

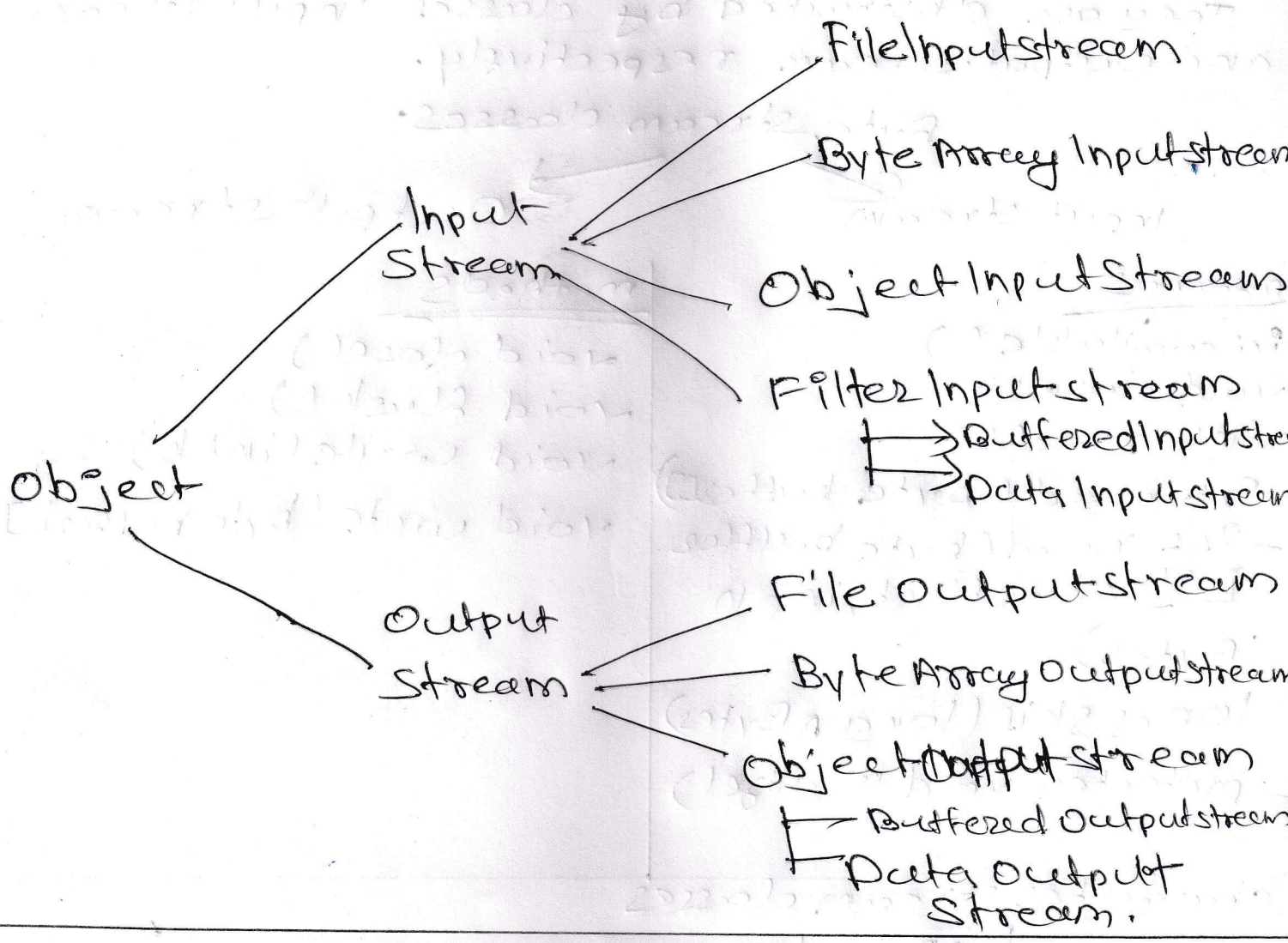
Reader

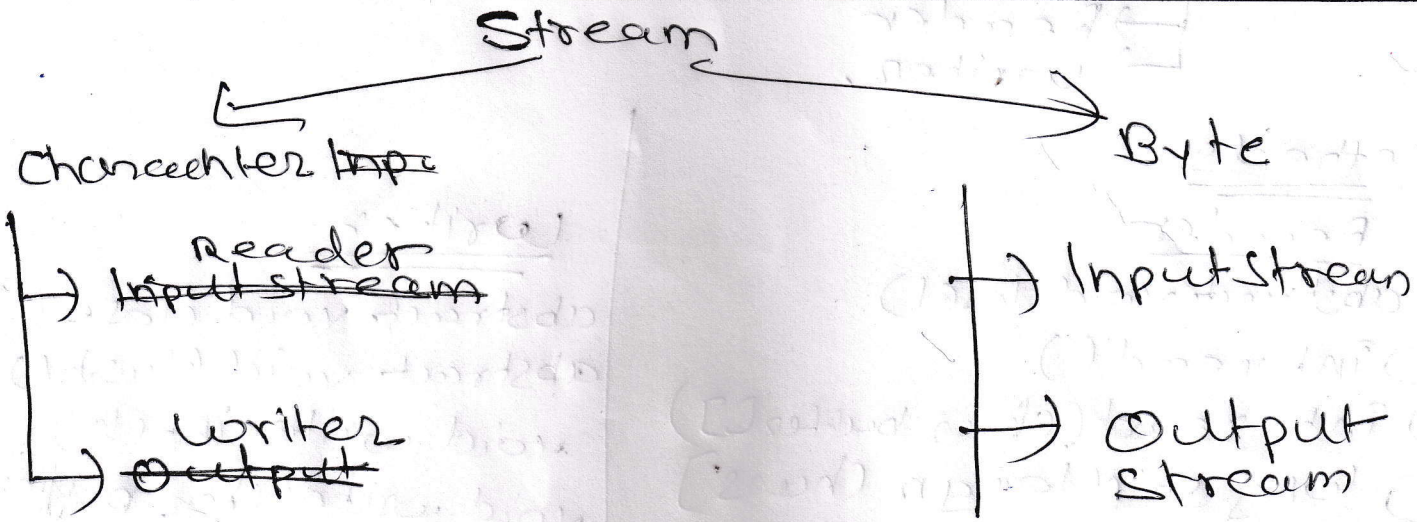
1. abstract void close()
2. int read()
3. int read(char buffer[])
4. long skip(long n chars)
5. abstract int read(char buffer[], int offset, int n chars)

Writer

- abstract void close()
- abstract void flush()
- void write(int ch)
- void write(char buffer[])
- void write(String str)

hierarchy of stream class





Byte stream Class	Character Stream class
Byte stream access file byte by byte (8 bit).	A character stream reads a file character by character. (16 bits).
Classified into 1. Input Stream classes 2. Output stream classes.	Classified into 1. Reader class 2. writer class
Handle raw binary data	Handle character based data
Read/write binary files.	Read/write text files
Sending/receiving binary data	Sending/receiving ~~character~~ data
Not specifically designed for string operations.	Convient methods for string operations.
Low level interface ~~for~~ byte data	Higher-level abstraction for text data.
Byte stream classes end with suffix InputStream and OutputStream.	Character stream end with suffix Reader and writer.
InputStream / OutputStream class is byte oriented	Reader/writer class is character oriented
It is possible to translate byte stream into character stream with InputStream Reader.	It is possible to translate character stream into byte stream with OutputStream writer.
The methods for byte Streams generally work with byte data type.	The methods for character streams generally accept parameter of data type char parameters.
Binary data, images, audio video.	Text based data, Strings, characters.

First program of file creating

```
import java.io.*;
Class createfile
  {
public static void main (String a[])
   {
File f= new File("C:\Users\Veneora\Desktop\LC.txt");
 try
   {
     if(f.createNewFile())
       {
     System.out.println("File Sucessfully created..");
       }
     else
       {
     System.out.println("File exists");
       }
   }
 catch(IoException i)
    {
   System.out.println("Exception handled");
    }
} }
```

O/P
File Sucessfully created..

else, no try catch

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

Java program to display the file information

```
java.io.*;
Class FileInfo
  {
  public static void main(String args[])
    {

  File f = new File("C:\Users\lenovo\Desktop\LC.txt");

    if(f.exist())
    {
 System.out.println("File Name: "+f.getName());
System.out.println("File location:" +f.getAbsolutePath());
System.out.println("File writeable:"+f.canWrite());
System.out.println("File Readable:"+f.canread());
System.out.println("File Size: "+f.length());
   } // f.delete() for deleting.
 else
   {
 System.out.println("File doesn't exists");
   }

  }
}
```

Output

abcd

```
File Name: LC.txt
File Location: C:\Users\lenovo\Desktop\LC.txt
File writeable: true
File Readable: true
File size: 4
```

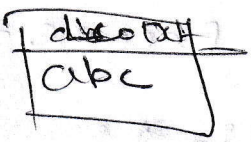
Program to write on File

```
import java.Io.*;
Class FileWriter
 {
  public static void main(String args[])
    {

      try
      {
    Filewriter f = new Filewriter("C:\\Users\\
    lenevo\\Desktop\\LC.txt");
     try
     {
     f.write("Java Programming");
     }
    finally
     {f.close();
     }
System.out.println("Successfully wrote");
}
catch(IOException i)
  {
System.out.println(i);
  }


}
}
```

program to read a text from file




```
import java.io.*;
class fileReader
  {
public static void main(String args[])
   {
    try
    {
  FileReader r = new FileReader("C:\Users\lenevo\
  Dekstop\LC.txt");
     try
     {
       int i;
     while((i=r.read())!=-1)
       {
     System.out.println((char)i);
        }
   }
  Finally
    {
    r.close();
  } }
catch(IOException e)
    {
System.out.println("Exception Handled");
    }

}
}
```

Output

abc

Linewise output
.readline()

Program to Rename the File

```
import java.io.*;
class RenameFile
  {
public static void main(String args[])
    {
File f= new File("C:\\Users\\lenevo\\Desktop
                \\abc.txt");
File r = new File("C:\\Users\\lenevo\\Desktop
                \\psc.txt");

  if(f.exists())
    {
System.out.println(f.renameTo(r));
    }
  else
    {
System.out.println("File doesn't exists...!");
    }

  }
}
```

imp copy one file data to another.


```
import java.io.*;
class copyFile
{
public static void main(String args[])throws
{ IOException.
FileInputstream r= new FileInputstream
("C:\\Users\\leneovo\\Desktop\\abc.txt");
FileOutputstream w= new FileOutputstream
("C:\\Users\\leneovo\\Desktop\\pys.txt");
int i;
while((i=r.read())!=-1)
{
w.write((char)i);
}
System.out.println("Data Copied Sucessfully"
}
}
```

Program Count Number of words/line/paragraph.

```
package java.io.*;
public class Count
{
public static void main(String args[])
throws IOException
{
String filename="C:\\Program Files\\Java\\abc.txt"};
int wordCount=0
int lineCount=0
int paraCount=0
```

```
try (BufferedReader r = new BufferedReader(new FileReader
(fileName)))
  {
  String line;
  while((line = reader.readLine()) != null)
   {
     lineCount++;
    if(line.trim().isEmpty())
      {
      paraCount++;
      }
    else
      {
  String[] words = line.split("\\s+");
    wordCount+= words.length;
    }
  }
}
catch (IOException e)
  {
  S.O.P(e.getMessage());
  return;
  }
S.O.P(wordCount);
S.O.P(lineCount);
SO.P(paraCount);

  }

}
```