IOStreams

Q. What are streams?

When we transfer data from one location to another location called streams.
Means we can send data from one computer to another computer in network or send data from one memory to another memory called as streams

If we think about iostream we can store our java program output in file permanently or read file from hard disk to our java program

If we want to work with streaming in JAVA we have one inbuilt package name as java.io
And this package provide some classes and interfaces also to work with files

When we want to send data to some location or get data from some location we need to know the path

If we want to work with path we have File class provided by java.io package

File class from java.io package

File class is used for access the drive list from computer, create folder , create file , delete folder , delete files, access the total space drive, free space of drive, list of files and list folders from specified path

How to access drive list or partition list from hard disk

If we want to check the number of partitions of your hard disk or number of drives in a hard disk then we have the listRoots() method of File class.

Syntax: static File [] listRoots(): this method helps us to return all drive lists from your hard disk.
Error! Filename not specified.

Access total space of drive and free space of drive

long getTotalSpace(): this method help us to return the total space of drive in bits format
long getFreeSpace(): this method help us to return the free space of drive in bits format
Error! Filename not specified.

How to create a folder using file class?

If we want to create folder using a File class we have method
boolean mkdir(): this method is used for creating a folder using File class and if the folder created successfully returns true otherwise returns false.
Example: we want to create folder on D drive name as augsepoct
Error! Filename not specified.
Now we want to execute same program and check the output
Error! Filename not specified.

How to check folder or file exist on specified path

boolean exists(): this method can check if a folder or file is present or not on a specified path if present return true otherwise return false.
Error! Filename not specified.

How to fetch all files and folder from particular path

If we want to fetch all files and folder from particular path we have method name as listFiles()

Syntax: File ref[] listFiles(): this method is used for fetching all files and folders using some specified path.

Error! Filename not specified.
If we think about above output we fetch all files and folders but we want to fetch only Files or folder from particular path.

boolean isFile(): this method checks if the path is file or not if file return return otherwise return false.

boolean isDirectory() : this method checks if the path is a folder or not if the path is a folder return true otherwise return false.

Example: Fetch only files from specified path

Error! Filename not specified.
Example: Fetch only folder from specified path
Error! Filename not specified.
How to create file using a File class

If we want to create file using a File class we have method name as
boolean createNewFile(): this method can create a file and file is created return true otherwise return false.

Error! Filename not specified.

Once we create file we can store data in file by using java program or read data from file using java program

How to store data in file using JAVA Program

If we want to store data in a file using Java Program we have two ways.

1. **Using Text format :** if we want to work with document file, text file etc format then we can store data in file using text format
2. **Using Byte Format :** if we want to work on image , audio , video file then we can use byte format.

How to work with Text format

If we want to work with text format in JAVA we have Reader and Writer classes
Means those classes name ends with Reader or Writer they support to text format

How to work with byte format

If we want to work with byte format in JAVA we have stream classes
Means those classes ends with stream word they support to byte format
Like as FileOutputStream,FileInputStream etc

Now we want to discuss about the Writer classes

Writer classes help us to write data in file using a text format.
If we want to work with Writer classes we have following class Hierarchy

Error! Filename not specified.

If we think about the above diagram, the Writer class is parent of all text data writing classes and it is an abstract class which contains some abstract methods which help us to write data in file and this file contains write() method and it is an overloaded method which helps us to write data in file.

Methods of Writer class

void write(char): this method can write single character at time in file
void write(char[]): this method can write character array in file
void write(String): this method can write string data in file
void write(char[],int offset, int length): this method can write specified length of data from character array in file

char[]: this parameter contain actual data
Int offset: this parameter decide from which index data want to write
Int length: this parameter decide how much data want to write in file'
void write(String,int offset,int length): this method can write specified length of data from string in file

String : this parameter contain actual data
Int offset: this parameter decide from which index data want to write
Int length: this parameter decide how much data want to write in file'

void close(): this method can close the file

FileWriter class

FileWriter class is used for write textual format data in file or text formatted data in file
Constructor of FileWriter class

Error! Filename not specified. Error! Filename not specified.
Error! Filename not specified.
Error! Filename not specified.

Example: we want to create a file and store text data in it.
Error! Filename not specified.
Note: if we think about above code we write data using FileWriter class in file with append mode.
When we add data in file then new data added only on same line
Means using FileWriter class we cannot write data on new line in file
So if we want to write data on new line in file we have BufferedWriter class

Now we want to discuss about BufferedWriter class

Constructor of BufferedWriter class

Error! Filename not specified.

Note: BufferedWriter class contains newLine() method which is used for writing data on new lines in a file.
Error! Filename not specified.

How to read textual file data

If we want to read file data using textual format we have following class hierarchy
Error! Filename not specified.

Reader class

Reader class is abstract class and it contain some abstract methods which help us to read data from file

int read(): this method is used for reading a single single character and returning ascii code from file and return -1 when file is finished or when file has no data.

int read(char[]): this method is used for read file data and store in character array and return its length and file has no data or end then return -1

int read(char[],int offset,int length): this method can read specified length of file data from file.

Parameter details

char [ ]: this parameter contain actual data

int offset/index: this parameter decide from which index we want to read data from file and index start from 0th location in file

int length: this parameter decides how much data you want to read from the file.

void mark(int index): set pointer from data want to read from file or set index location from data want to read from file

FileReader class

constructor of FileReader class

FileReader(String path): accept the file path using string format and use read mode by default

FileReader(File path): accept the file path using file class reference and use read mode by default.

Example using FileReader with read() method

Error! Filename not specified.

Example using read(char[]);

```
import java.io.*;
public class ReadAPP
{  public static void main(String x[])throws Exception
        {   File f=new File("D:\\augsepoct\\third.txt");
                FileReader fr= new FileReader(f);
                int data;
                char ch[]=new char[(int)f.length()];
                  fr.read(ch);
                  for(char c:ch)
                  {  System.out.print(c);
                  }
        }
}
```

Note: if we use the FileReader class then we can read single single character data from file But if we want to read line by line data from a file we have to use the BufferedReader class. BufferedReader class uses readLine() method which is used for reading single single line data from file and returning null when file is finished.

Constructor of BufferedReader

BufferedReader(Reader): we can pass reference of any child of Reader or Reader itself

Error! Filename not specified.

Stream classes

Stream classes are used for writing data in file using byte format.
If we want to work with Stream classes we have two types of stream class

OutputStream : OutputStream is a class which is parent of all stream classes and which is used for writing data in file.

InputStream : InputStream is a class which is parent of all stream classes and which is used for read byte format data from file.

Now we want to discuss about OutputStream class

Error! Filename not specified.

If we think about the above diagram we have OutputStream is a parent of all stream classes and it contains some methods which help us to write data in file using byte format.

public abstract void write(int) throws java.io.IOException: this function is used for write single byte at time in file
public void write(byte[]) throws java.io.IOException: this function is used for write byte array data in file
public void write(byte[], int offset, int length) throws java.io.IOException: this function for write byte array with some specified size
public void flush() throws java.io.IOException: this function can flush the buffer
public void close() throws java.io.IOException: this function can close the file.

FileOutputStream class

This class is used for writing byte data in a file.

Constructor of FileOutputStream class

FileOutputStream(String path): this constructor can accept file paths using a string format and by default use write mode for writing data in file
FileOutputStream(File path): this constructor can accept file path using a file class reference for accept file path and use by default write mode for writing data in file
FileOutputStream(String path,boolean mode): accept the file path using string format and if we pass true then file can use append mode and if we pass false then file can use write mode.
FileOutputStream(File path,boolean mode): this constructor accepts file path reference using file class reference and if user passes true then file can append otherwise file can write mode.

Example: WAP to input string data from keyboard and store in file using byte format.
Error! Filename not specified.

InputStream classes

InputStream classes are used for reading data from files using byte format.

Hierarchy of InputStream classes

Error! Filename not specified.
Note: here InputStream is an abstract class and it contains some abstract methods which help us to read data from file using byte format.

int read(): this method is used for reading single single byte data from file and return -1 when file is finished or when file has no data.

int read(byte[]): read complete data from file and store in byte array and return its length and if file has no data or finished then return -1

int read(byte[],int offset,int length): read the specified size of byte data from file

void mark(int index): this method can mark the index location and read data from specified index.
void close(): this method can close the reading object or file.
FileInputStream class

Constructor of FileInputStream class

FileInputStream(String path): this constructor accept file class reference using string format and use read mode by default

FileInputStream(File path): this constructor accepts file path using file class reference and uses read mode by default.

Error! Filename not specified.

Example: we want to copy the image from particular location and paste on some other location

```
import java.io.*;
import java.util.*;
public class ImgCopyPasteApp
{   public static void main(String x[])throws Exception
        { FileInputStream fr = new FileInputStream("D:\\OCT2024\\er.jpg");
          int data;
          FileOutputStream fw =new FileOutputStream("D:\\augsepoct\\er.jpg");
          while((data=fr.read())!=-1)
          { fw.write(data);
          }
          fw.close();
          fr.close();
          System.out.println("SAVE");
        }
}
```

Serialization and Deserialization

Serialization : Serialization is process to store user define object in file called as serialization
When we store user define object in file then internally JVM encrypt the object content and store in file means when some open the file and check the file content then file content is not human readable format means we provide security to user file
As well as when we share in network from one machine to another machine in the form java object and if we serialize object then data send via network in encrypted format

Steps to work with Serialization

---

1. **Add java.io package in application**
   Syntax: import java.io.*;
2. **Create POJO class and implement Serializable interface in it.**

Note: Serializable is marker interface in JAVA

Example with source code

```
import java.io.*;

class Employee implements Serializable
{
   private int id;
        private String name;
        private long sal;

        public void setId(int id)
        { this.id=id;
        }
        public int getId(){
          return id;
        }
        public void setName(String name)
        { this.name=name;
        }
        public String getName(){
          return name;
        }
        public void setSal(int sal)
        { this.sal=sal;
        }
        public int getSal()
        { return sal;
        }
}
```

3. **Create Object of ObjectOutputStream class**
   Constructor of ObjectOutputStream
   Syntax:
   ObjectOutputStream(OutputStream): this constructor accepts reference of OutputStream means we can pass reference of any child of OutputStream.

Example

```
FileOutputStream fout=new FileOutputStream("D:\\augsepoct\\ser.txt");
ObjectOutputStream oout=new ObjectOutputStream(fout);
```

4. **Call writeObject() method of OutputStream class**
   Syntax: void writeObject(Object): this method can save object of any class.

Example with source code

```
import java.io.*;

class Employee implements Serializable
```

```
{
   private int id;
        private String name;
        private long sal;

        public void setId(int id)
        { this.id=id;
        }
        public int getId(){
          return id;
        }
        public void setName(String name)
        { this.name=name;
        }
        public String getName(){
          return name;
        }
        public void setSal(int sal)
        { this.sal=sal;
        }
        public long getSal()
        { return sal;
        }
}
public class ObjSerApp
{
   public static void main(String x[])throws Exception
        {
                FileOutputStream fout=new FileOutputStream("D:\\augsepoct\\ser.txt");
                ObjectOutputStream oout=new ObjectOutputStream(fout);

                Employee emp = new Employee();
                emp.setId(1);
                emp.setName("ABC");
                emp.setSal(100000);

                oout.writeObject(emp); //write employee object in file and encrypt its data
                oout.close();
                System.out.println("Save");
        }
}
```

Deserialization

Deserialization means reading object data from a file called deserialization.
If we want to read object data from file we have class name as ObjectInputStream and its one method name as readObject()

Constructor

ObjectInputStream(InputStream): this constructor accepts file path using inputstream reference.

```
import java.io.*;
class Employee implements Serializable
{          private int id;
           private String name;
           private long sal;
           public void setId(int id)
           { this.id=id;
           }
           public int getId(){
             return id;
           }
           public void setName(String name)
           { this.name=name;
           }
           public String getName(){
             return name;
           }
           public void setSal(int sal)
           { this.sal=sal;
           }
           public long getSal()
           { return sal;
           }
}
public class ObjDSerApp
{
    public static void main(String x[])throws Exception
           {          FileInputStream fin=new FileInputStream("D:\\augsepoct\\ser.txt");
                      ObjectInputStream ooin=new ObjectInputStream(fin);
                       Object obj=ooin.readObject();
                       if(obj!=null)
                       {Employee emp=(Employee)obj;
                        System.out.println(emp.getId()+"\t"+emp.getName()+"\t"+emp.getSal());
                       }
                       else{
                                 System.out.println("No Data");
                       }
           }
}
```

transient: if we want to avoid some data store in file in serialization process then we can use the transient non access specifier with variable