Chapter 1.5
File I/O

Files and Directories

- Storing and managing data using files is known as file processing, which includes tasks such as Creating files, updating files and manipulating data.
- A file is collection of similar kinds of records.
- Generally, file handling mechanism involves the following steps.
  1. Opening the file.
  2. Performing Read or write operation.
  3. Closing the file
- In order to work with file object, we must import java.io.File.
- Some common constructors that can be used to create file object are
  1. File (String DirectoryPath)
  2. File (String DirectoryPath, String FileName)
- A directory is a file that contains a list of other files and directories. i.e. placeholder for file and subdirectories is called directories.
- Some of the important properties of files and directories are
  1. public boolean exists(); // check if file/directory exist or not
  2. public boolean isDirectory()// test if the instance is directory or not
  3. public boolean isFile() // test if the instance is file or not
  4. public File [] listFiles()// Return array of files from the specified directory
  5. public String [] list()// Return array of string containing name of file and directories from specified path.
  6. public String getPath()// Return Location of this file with name
  7. public String getName()// Return name of this file
  8. publc boolean delete() // delete the file/directory
  9. public boolean renameTo(File newname) .// rename a file

Example:

```
import java.io.File;
public class Fl
{
        public static void main(String [] args)
        {
                File da=new File("E:\\Ram");  //directory
                File f=new File ("demo.txt");  //file
                // or File f1=new File("C:\\","demo.txt");
                boolean b=f.exists();
                boolean d=f.isDirectory();
                boolean g=f.isFile();
                System.out.println(b);
                System.out.println(d);
                System.out.println(g);
                File nf=new File("Bhesh.txt");
                f.renameTo(nf); // nf must of file type not string
                nf.delete();
        }
}
```

I/O Stream Classes

- A stream is a general name given to a flow of data.

- A stream is a data abstraction for input and output to and from the program.
- Different streams are used to represent different kinds of data flow.
- Java provides several stream classes to read input and write output different I/O primitives such as keyboard, file, network, memory buffer etc.

- Basically java has two stream classes.
  1. Byte Streams: InputStream and OutputStream
  2. Character Streams: Reader and Writer

1. Byte Stream

- The byte stream class provide rich environment for handling byte oriented I/O.
- Bytes means reading or writing binary data.
- A program uses input stream to read data from a source one at a time and output stream to write data to a destination one at a time
- Three common input byte stream classes are
  - **FileInputStream**
  - DatInputStream
  - BufferedInputStream
- Three common output byte stream classes are
  - **FileOutputStream**
  - DataOutputStream
  - BufferedOutputStream

FileInputStream and FileOutputStream

- FileInputStream class creates an InputStream that can be used to read bytes from a file.
- FileOutputStream class creates and outputStream that can be used to write bytes to a file.
- Some commonly used constructors used are
  - FileInputStream(String Filename)
  - FileOutputStream(String Filename)
  - FileOutputStream(String Filepath, boolean append)

- Some commonly used methods are
  - int read() : Used to read a byte of data defined in FileInputStream
  - void write(byte [] b): Used to write specified b length byte in a file defined in FileOutputStream
  - void close(): Used to close the file operation

1. Write a program to create a file welcome.txt and write text "Welcome to Java Lab" to it.

```
import java.io.*;
class FileOutputStreamDemo
{
        public static void main(String args[])
        {
                try
                {
                        FileOutputStream fout=new FileOutputStream("welcome.txt");
                        String s="Welcome to Java Lab";
                        //fout.write(s.getBytes()); // getBytes() returns the byte array of string
                        byte [] b=s.getBytes();
                        fout.write(b);
```

```
                fout.close();
                System.out.println("successfully written to file");
            }

            catch(Exception e)
            {
                System.out.println(e);
            }
        }
}
```

2. Write a program to input name and roll of 10 students and write it into file named "std.txt"

```
import java.io.*;
import java.util.*;
class Demo
{
    public static void main(String args[])
    {
        try
        {
            Scanner sc=new Scanner(System.in);
            FileOutputStream fout=new FileOutputStream("std.txt");
            System.out.println("Enter name and roll of 10 students");
            for(int i=0;i<10;i++)
            {
                String nm=sc.nextLine();
                int rl=sc.nextInt();
                String data= nm+"\t"+ rl+"\n";
                fout.write(data.getBytes());
                //sc.nextLine(); //Flush if necessary
            }
            fout.close();
            System.out.println("successfully written to file");
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

3. WAP to read the above file and display the content in console.

```
import java.io.*;
class FileInputStreamDemo
{
    public static void main(String args[]) throws Exception
    {
            FileInputStream fis=new FileInputStream("std.txt");
            //int c=fis.read(); // c will get Ascii code of first character stored in file
            int c;
            while( (c=fis.read())!=-1)  //i.e in.read() return -1 if EOF is encountered
```

```
                {
                        System.out.print((char)c);
                }
                fis.close();
        }
}
```

4. **Wap to copy content of file "abc.txt" to another new file.**

```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
public class F
{
        public static void main(String [] args) throws Exception
        {
                FileInputStream in=new FileInputStream("abc.txt");
                FileOutputStream out=new FileOutputStream("def.txt"); //create new file
                int c;
                while((c=in.read())!=-1) // i.e in.read() return -1 if EOF is encountered
                {
                        out.write(c);
                }
                in.close();
                out.close();
        }
}
```

5. **Write a program to input name, roll and marks in three different subject for a student. Create a file named std.txt and store all this information along with total and percentage.**

```
import java.io.File;
import java.io.FileOutputStream;
import java.util.Scanner;
public class Demo
{
        public static void main(String [] args) throws Exception
        {
                Scanner sc=new Scanner(System.in);
                System.out.println("Enter the name");
                String nm=sc.nextLine();
                System.out.println("Enter the roll");
                int rol=sc.nextInt();
                System.out.println("Enter the marks in 3 different subject");
                double m1=sc.nextDouble();
                double m2=sc.nextDouble();
                double m3=sc.nextDouble();
                double tot=m1+m2+m3;
                double pct=tot/3;
                String stf=nm+" " +rol +" " +m1 + " " +m2+" " +m3+" " +tot+" "+pct;
                FileOutputStream out=new FileOutputStream("Std.txt"); //create new file
                out.write(stf.getBytes());
                out.close();
                System.out.println("Successfully saved to file");
```

```
	}
}
```

6. **WAP to read Name and Address of student and write it to file using tab delimiter.**

```
import java.io.File;
import java.io.FileOutputStream;
import java.util.Scanner;
public class F
{
	public static void main(String [] args) throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the Name");
		String nm=sc.nextLine();
		System.out.println("Enter the Address");
		String ad=sc.nextLine();
		String stf=nm+"\t"+ad;

			FileOutputStream out=new FileOutputStream("Std.txt"); //create new file
			out.write(stf.getBytes());
			out.close();
			System.out.println("Successfully saved to file");
	}
}
```

7. **Wap to copy the content of file "abc.txt" and append it in the existing file "def.txt".**

```
import java.io.*;
public class F
{
	public static void main(String [] args) throws Exception
	{
			FileInputStream in=new FileInputStream("abc.txt");
			FileOutputStream out=new FileOutputStream("def.txt",true); //open in append mode
			int c;
			while((c=in.read())!=-1) // i.e in.read() return -1 if EOF is encountered
			{
				out.write(c);
			}

			in.close();
			out.close();
	}
}
```

8. **WAP to read the content of file "abc.txt" and count the number of digits, vowel, consonants, space and words.**

```
	import java.io.*;
	class FileOutputStreamDemo
	{
		public static void main(String args[])
		{
```

```
                try
                {
                        FileInputStream fis=new FileInputStream("abc.txt");
                        int c;
                        int cdig=0,cvow=0,ccons=0, cspac=0, cword=0 ;
                        while((c=fis.read())!=-1)
                        {
                                if(c>='0' && c<='9')
                                {
                                        cdig++;
                                }
                                else if(c=='a' || c=='e' ||c=='i' ||c=='o' ||c=='u' ||c=='A' || c=='E' ||c=='I'
                                                                                ||c=='O' ||c=='U' )
                                {
                                        cvow++;
                                }
                                else if((c>='a' && c<='z') ||(c>='A' && c<='Z'))
                                {
                                        ccons++;
                                }
                                else if(c==' ')
                                {
                                        cspac++;
                                }
                        }
                        fis.close();
                        cword=cspac+1;
                        System.out.println("Total number of Digits="+cdig);
                        System.out.println("Total number of Vowels="+cvow);
                        System.out.println("Total number of Consonants="+ccons);
                        System.out.println("Total number of Space="+cspac);
                        System.out.println("Total number of Words="+cword);
                }
                catch(Exception e)
                {
                        System.out.println(e);
                }
        }
}
```

9. WAP to delete all the vowel characters in the file named “abc.txt”.

```
import java.io.*;
class FileOutputStreamDemo
{
        public static void main(String args[])
        {
                try
                {
                        FileInputStream fis=new FileInputStream("abc.txt");
                        FileOutputStream fos=new FileOutputStream("tmp.txt");
                        int c;
                        while((c=fis.read())!=-1)
```

```
                {

                    if(!(c=='a' || c=='e' ||c=='i' ||c=='o' ||c=='u' ||c=='A' || c=='E' ||c=='I' ||c=='O' ||c=='U' ))
                        {
                            fos.write(c);
                        }
                }
                fis.close();
                fos.close();
                File f1=new File("abc.txt");
                f1.delete();
                File f2=new File("tmp.txt"); //old name
                File f3=new File("abc.txt"); //new name
                f2.renameTo(f3);
            }
            catch(Exception e)
            {
                System.out.println(e);
            }
        }
}
```

2. Character Stream

- A byte stream access the file byte by byte. Java programs use byte streams to perform input and output of 8-bit bytes. It is suitable for any kind of file, however not quite appropriate for text files. For example, if the file is using a unicode encoding and a character is represented with two bytes, the byte stream will treat these separately and you will need to do the conversion yourself.
- So the main problem in byte stream class is that they can't work directly with Unicode character, since one of the main purpose of java is to support 'Write once Run anywhere' paradigm, it is necessary to direct support for characters.
- Character stream classes provides a convenient means of handling input and output of character.
- Character streams are more efficient than byte stream. They use Unicode and hence can be internationalized.
- A byte stream read the file byte by byte while the character stream read the file character by character.
- Character stream classes are defined by using the two classes Reader and Writer that define several key methods that other stream classes implement.
- Two of the important methods are
    1. read()
    2. write()
- One of the commonly used character stream class is FileReader for input and FileWriter for output.

File Reader

The FileReader is a subclass of Reader Class that can be used to read the content of a file.

We use the following constructor

- FileReader(String Filepath)
- FileReader(File fileobject)

FileWriter

FileWriter class creates Writer that can be used to write a file.

We can use the following constructors.

- FileWriter(String filepath)
- FileWriter(String filepath,boolean append)

- FileWriter(File Fileobj)
- FileWriter(File Fileobj, boolean append)

Example:

1. **Wap to display the text of file "Bhesh.txt" in standard output Stream.**

```
import java.io.*;
public class Bh
{
        public static void main(String args []) throws IOException
        {

                FileReader fr=new FileReader("Bhesh.txt");
                int c;
                 while((c=fr.read())!=-1)
                    {
                        System.out.print((char)c);
                     }
        }
}
```

Or

```
        import java.io.*;
        public class Bh
        {
                public static void main(String args []) throws IOException
                {
                        FileReader fr=new FileReader("Bhesh.txt");
                        BufferedReader br=new BufferedReader(fr);
                        String s;
                        while((s=br.readLine())!=null)
                        {
                                System.out.println(s);
                        }
                }
        }
```

2. **Wap to copy the content of one file to another file using FileReader and FileWriter character strem**

```
import java.io.*;
public class CopyFile
{
        public static void main(String [] args) throws IOException
        {
                FileReader fr=new FileReader("abc.txt");
                FileWriter fw=new FileWriter("new.txt");
                int c;
                while((c=fr.read())!=-1)
                {
                        fw.Write(c);
                }
```

```
                fr.close();
                fw.close();
        }
    }
```

3. WAP to count the number of line in file "abc.txt".

```
import java.io.*;
public class Bh
{
    public static void main(String args []) throws IOException
    {
        FileReader fr=new FileReader("abc.txt");
        BufferedReader br=new BufferedReader(fr);
        int count=0;
        while((br.readLine())!=null)
        {
            count++;
        }
        System.out.println("Total number of line= "+count);
    }
}
```

4. WAP to input name and save in file abc.txt

```
import java.io.*;
import java.util.Scanner;
public class Bh
{
    public static void main(String args []) throws IOException
    {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter your name");
        String nm=sc.nextLine();
        FileWriter f=new FileWriter("abc.txt");
        f.write(nm);
        f.close();
    }
}
```

Object Stream

- An object stream is a sequence of objects that supports various methods which can be pipelined to produce the desired result.
- Just as data streams support I/O of primitive data types, object streams support I/O of objects.

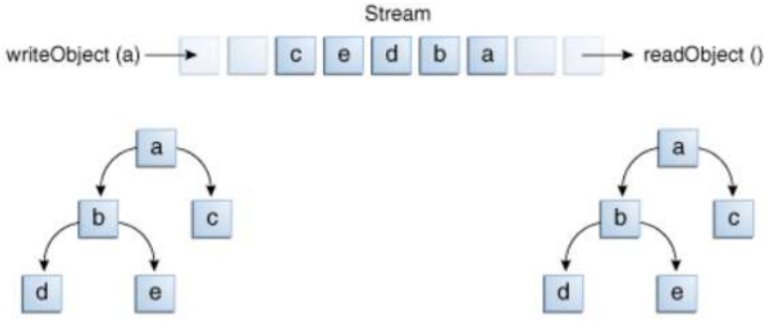

Fig: Object Stream I/O

- The object stream classes are **ObjectInputStream** and **ObjectOutputStream**. These classes implement ObjectInput and ObjectOutput, which are subinterfaces of DataInput and DataOutput.
- Only objects that support **the java.io.Serializable** interface can be written or read from streams.

ObjectOutputStream Class

- The OutputStream class of the java.io package is an abstract superclass that represents an output stream of bytes.
- The ObjectOutputStream class of the java.io package can be used to write objects that can be read by ObjectInputStream. It extends the OutputStream abstract class.
- Basically, the ObjectOutputStream encodes Java objects using the class name and object values. And, hence generates corresponding streams. This process is known as serialization.
- Those converted streams can be stored in files and can be transferred among networks.
- The ObjectOutputStream class provides implementations for different methods present in the OutputStream class. Some methods includes
    1. write() - writes a byte of data to the output stream
    2. writeBoolean() - writes data in boolean form
    3. writeChar() - writes data in character form
    4. writeInt() - writes data in integer form
    5. writeObject() - writes object to the output stream

ObjectInputStream

- The InputStream class of the java.io package is an abstract superclass that represents an Input stream of bytes.
- The ObjectInputStream class of the java.io package can be used to read objects that were previously written by ObjectOutputStream. It extends the InputStream abstract class.
- Basically, the ObjectOutputStream converts Java objects into corresponding streams. This is known as serialization. Those converted streams can be stored in files or transferred through networks.
- Now, if we need to read those objects, we will use the ObjectInputStream that will convert the streams back to corresponding objects. This is known as deserialization.
- The ObjectInputStream class provides implementations of different methods present in the InputStream class.
    1. read() - reads a byte of data from the input stream
    2. readBoolean() - reads data in boolean form
    3. readChar() - reads data in character form
    4. readInt() - reads data in integer form
    5. readObject() - reads the object from the input stream
- Serialization in Java allows us to convert an Object to stream that we can send over the network or save it as file or store in DB for later usage. Deserialization is the process of converting Object stream to actual Java Object to be used in our program.

Create a class Student with data member roll and name. Write java program

a. To write object of class Student in file named Student.txt.

```
import java.io.*;
class Student implements Serializable
{
    private String name;
    private int roll;
    public void setName(String name)
    {
            this.name=name;
    }
    public void setRoll(int roll)
```

```
    {
        this.roll=roll;
    }
    public String getName()
    {
        return(name);
    }
    public int getRoll()
    {
        return(roll);
    }
}
class Demo1
{
    public static void main(String [] args) throws Exception
    {
        Student s=new Student();
        s.setName("Ram Thapa");
        s.setRoll(5);
        FileOutputStream fos=new FileOutputStream("student.txt");
        ObjectOutputStream os=new ObjectOutputStream(fos);
        os.writeObject(s);
        os.writeObject(null); //Null will be treated as EOF while reading the  stream
        System.out.println("Done");
    }
}
```

b. To write N object of class Student in file named Student.txt.

```
    import java.io.*;
    import java.util.*;
    class Student implements Serializable
    {
        private String name;
        private int roll;
        public void setName(String name)
        {
            this.name=name;
        }
        public void setRoll(int roll)
        {
            this.roll=roll;
        }
        public String getName()
        {
            return(name);
        }
        public int getRoll()
        {
            return(roll);
        }
    }
    class Demo
```

```
{
	public static void main(String [] args) throws Exception
	{
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the number of students");
		int n=sc.nextInt();
		FileOutputStream fos=new FileOutputStream("student.txt");
		ObjectOutputStream os=new ObjectOutputStream(fos);
		for(int i=0;i<n;i++)
		{
			Student s=new Student();
			System.out.println("Enter name and roll of students");
			//sc.nextLine();
			String nm=sc.nextLine();
			int roll=sc.nextInt();
			s.setName(nm);
			s.setRoll(roll);
			os.writeObject(s);
		}
		os.writeObject(null); //Null will be treated as EOF while reading the  stream
		System.out.println("Done");
	}
}
```

c. To read first two students objects from the file named Student.txt and display them in console.

```
	import java.io.*;
	class Student implements Serializable
	{
		private String name;
		private int roll;
		public void setName(String name)
		{
			this.name=name;
		}
		public void setRoll(int roll)
		{
			this.roll=roll;
		}


		public String getName()
		{
			return(name);
		}
		public int getRoll()
		{
			return(roll);
		}

	}
	class Demo
	{
```

```
	public static void main(String [] args) throws Exception
	{
		Student s=new Student();
		FileInputStream fis=new FileInputStream("student.txt");
		ObjectInputStream is=new ObjectInputStream(fis);
		//Read First Object
		//s= (Student)is.readObject();
		//System.out.println(s.getName());
		//System.out.println(s.getRoll());
		for(int i=0;i<2;i++)
		{
			s= (Student)is.readObject();
			System.out.println("Name="+s.getName());
			System.out.println("Roll="+s.getRoll());
		}
	}
}
```

d. To read all the students objects from the file named Student.txt and display them in console.

```
import java.io.*;
class Student implements Serializable
{
	private String name;
	private int roll;
	public void setName(String name)
	{
		this.name=name;
	}
	public void setRoll(int roll)
	{
		this.roll=roll;
	}


	public String getName()
	{
		return(name);
	}
	public int getRoll()
	{
		return(roll);
	}

}
class Demo
{
	public static void main(String [] args) throws Exception
	{
		Student s=new Student();
		FileInputStream fis=new FileInputStream("student.txt");
		ObjectInputStream is=new ObjectInputStream(fis);

		//Read First Object
		//s= (Student)is.readObject();
		//System.out.println(s.getName());
```

```
                    //System.out.println(s.getRoll());

                    while((s= (Student)is.readObject())!=null)//Assuming the last object written is null
                    {
                            System.out.println("Name="+s.getName());
                            System.out.println("Roll="+s.getRoll());
                    }
            }
    }
```

e. To display all the student objects from the file whose roll number is greater than 10 named Student.txt and display them in console.

```
    import java.io.*;
    class Student implements Serializable
    {
            private String name;
            private int roll;
            public void setName(String name)
            {
                    this.name=name;
            }
            public void setRoll(int roll)
            {
                    this.roll=roll;
            }


            public String getName()
            {
                    return(name);
            }
            public int getRoll()
            {
                    return(roll);
            }

    }
    class Demo
    {
            public static void main(String [] args) throws Exception
            {
                    Student s=new Student();
                    FileInputStream fis=new FileInputStream("student.txt");
                    ObjectInputStream is=new ObjectInputStream(fis);
                    while((s= (Student)is.readObject())!=null)//Assuming the last object written is null
                    {
                            if(s.getRoll()>10)
                            {
                                    System.out.println("Name="+s.getName());
                                    System.out.println("Roll="+s.getRoll());
                            }
                    }
            }
```

```
        }
```

f. To count all the student objects from the file named Student.txt and display it in console.

```
import java.io.*;
class Student implements Serializable
{
        private String name;
        private int roll;
        public void setName(String name)
        {
                this.name=name;
        }
        public void setRoll(int roll)
        {
                this.roll=roll;
        }
        public String getName()
        {
                return(name);
        }
        public int getRoll()
        {
                return(roll);
        }
}
class Demo
{
        public static void main(String [] args) throws Exception
        {
                Student s=new Student();
                FileInputStream fis=new FileInputStream("student.txt");
                ObjectInputStream is=new ObjectInputStream(fis);
                int count=0;
                while((s= (Student)is.readObject())!=null)//Assuming the last object written is null
                {
                        count++;
                }
                System.out.println("Total Number of Student="+count);
        }
}
```

Random Access File

- All the I/O stream we studied yet are one-way stream and provide **sequential access only**.
- One-way stream means, they are either read only input stream or write only output stream.
- And sequential access means reading or writing data in serial order.
- Random Access File allows us to read as well as modify existing records at same time and also provide random access to a disk file.
- A random-access data file is a file in which we can read or write information anywhere in the file
- Two ways to create a random access file are
  - RandomAccessFile f1=new RandomAccessFile(“filename”, ”r”); // i.e. read only access

- RandomAccessFile f2=new RandomAccessFile("filename", "rw");// i.e. read-write access

- It provides following methods
  - void seek(long position); // seek the file byte pointer to specified position for next read/write operation
  - long getFilePointer(); // get the position of the current file pointer
  - long length(); // get the lengths of file
  - int readInt(), char readChar(), boolean readBoolean(), String readLine() etc for reading primitives type from the file
  - writeInt(int), writeChar(char), writeBoolean(boolean), writeBytes(String) etc for writing primitives type to the file

Example

Write the output of the following program

Notes: 2 byte for char, 4 byte for int , 4 byte for float, 8 byte for double , 1byte for boolean

```
import java.io.*;
class A
{
    public static void main(String [] args) throws IOException
    {
        RandomAccessFile raf=new RandomAccessFile("A.txt","rw");
        raf.writeChar('X');
        raf.writeInt(444);
        raf.writeDouble(3.1416);
        raf.writeBoolean(true);
        raf.writeChar('Y');
        raf.writeInt(677);
        System.out.println("The length of file in bytes is " +raf.length());

        //reading serially
        raf.seek(0); // seek to start position
        System.out.println(raf.readChar());
        System.out.println(raf.readInt());
        System.out.println(raf.readDouble());
        System.out.println(raf.readBoolean());
        System.out.println(raf.readChar());
        System.out.println(raf.readInt());

        //reading by jumping to specific location
        raf.seek(0);
        System.out.println(raf.readChar());
        raf.seek(2);
        System.out.println(raf.readInt());
        raf.seek(6);
        System.out.println(raf.readDouble());
        raf.seek(14);
        System.out.println(raf.readBoolean());
        raf.seek(15);
        System.out.println(raf.readChar());
        raf.seek(17);
        System.out.println(raf.readInt());
        raf.close();
    }
```

```
}
```

Output:

```
The length of file in bytes is 21.
X
444
3.1416
True
Y
677
X
444
3.1416
True
Y
677
```

3. **WAP to read a string from keyboard and append it in a file.**

```
import java.io.*;
import java.util.*;
class Append
{
	public static void main(String [] args) throws Exception
	{
		RandomAccessFile ra=new RandomAccessFile("A.txt","rw");
		Scanner sc=new Scanner(System.in);
		System.out.println("Enter the text to be appended");
		ra.seek(ra.length());// seek to last position
		String txt=sc.nextLine();
		ra.writeBytes(txt);
		ra.close();
	}
}–
```