**PRACTICAL # 06**

**OBJECT:**

**Files I/O in Java.**

**THEORY:**

The java.io package contains classes to perform input and output (I/O) in Java.

A stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.

## **Byte Streams**

Java byte streams are used to perform input and output of 8-bit bytes. The most frequently used byte stream classes are, **FileInputStream** and **FileOutputStream**.

**Program:**

import java.io.\*;

public class CopyFile {

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

{

FileInputStream in = null;

FileOutputStream out = null;

try {

in = new FileInputStream("input.txt");

out = new FileOutputStream("output.txt");

int c;

while ((c = in.read()) != -1) {

out.write(c);

}

}finally {

if (in != null) {

in.close();

}

if (out != null) {

out.close();

}

}

}

}

## **Character Streams**

Java **Character** streams are used to perform input and output for 16-bit unicode. The most frequently used classes are, **FileReader** and **FileWriter.**. Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time.

The above example re-written to makes use of these two classes to copy an input file into an output file:

import java.io.\*;

public class CopyFile {

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

{

FileReader in = null;

FileWriter out = null;

try {

in = new FileReader("input.txt");

out = new FileWriter("output.txt");

int c;

while ((c = in.read()) != -1) {

out.write(c);

}

}finally {

if (in != null) {

in.close();

}

if (out != null) {

out.close();

}

}

}

}

## **Standard Streams**

Following is a simple program which creates **InputStreamReader** to read standard input stream until the user types a "q":

import java.io.\*;

public class ReadConsole {

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

{

InputStreamReader cin = null;

try {

cin = new InputStreamReader(System.in);

System.out.println("Enter characters, 'q' to quit.");

char c;

do {

c = (char) cin.read();

System.out.print(c);

} while(c != 'q');

}finally {

if (cin != null) {

cin.close();

}

}

}

}

## **Directories in Java**

A directory is a File which can contains a list of other files and directories. **File** object can create directories, list down files available in a directory and many other functions.

**Creating Directories**

**mkdir( )** method creates a directory, returning true on success and false on failure. Failure indicates that the path specified in the File object already exists, or that the directory cannot be created because the entire path does not exist yet.

**mkdirs()** method creates both a directory and all the parents of the directory.

Following example creates "/tmp/java/bin" directory:

import java.io.File;

public class CreateDir {

public static void main(String args[]) {

String dirname = "/tmp/java/bin";

File d = new File(dirname);

// Create directory now.

d.mkdirs();

}

}

## **Listing Directories**

**list( )** method provided by **File** lists down all the files and directories available in a directory:

import java.io.File;

public class ReadDir {

public static void main(String[] args) {

File file = null;

String[] paths;

try{

// create new file object

file = new File("/tmp");

// array of files and directory

paths = file.list();

// for each name in the path array

for(String path:paths)

{

// prints filename and directory name

System.out.println(path);

}

}catch(Exception e){

// if any error occurs

e.printStackTrace();

}

}

}

**ACTIVITIES**

**Activity 1**

Write and execute the above programs.

**Activity 2**

Write a program that writes prime numbers upto 100 in a text file.

**Activity 3**

Write a program to read a text file, invert case of each character and write back to another text file.

**REVIEW QUESTIONS**

1. What number shows end of input file stream ?
2. Which package contains all the java I/O functionality?
3. What is the difference between file and directory?

1. What will happen if a file is unable to be read or written?