Creating a File

Lets start working with file by creating a new file. To create a file, we are using createNewFile method that takes filename as an argument. This method creates a new file if file is not already exist. See the example below.

import java.io.File;

import java.io.IOException;

public class FileCreateDemo1

{

public static void main(String[] args)

{

try

{

File Obj = new File("FileDemo.txt");

if (Obj.createNewFile()) {

System.out.println("\*\*\*\*\*\*File created\*\*\*\*\*\*");

System.out.println("Name of the file = " + Obj.getName());

}

else{

System.out.println("File already exists.");

}

}

catch (IOException e){

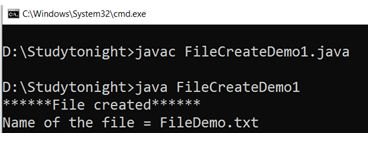
e.printStackTrace();

}

}

}

Copy



Example: Writing in a file

After creating a file, now will write data to the file. To write data, a method of Filewrite class is used. It will write the data but may throw exception so make sure to handle the excecptions as well.

import java.io.FileWriter;

import java.io.IOException;

public class FileWriteDemo1

{

public static void main(String[] args)

{

try

{

FileWriterobj = new FileWriter("FileDemo.txt");

obj.write("Welcome to studytonight.com.");

obj.close();

System.out.println("File is Updated.");

}

catch (IOException e)

{

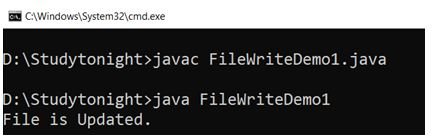
e.printStackTrace();

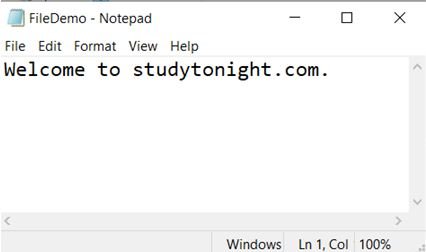
}

}

}

Copy





Example : Reading a file

To read data from the file we used File and Scenner classes both are used to handle input and output of system resources. Here we are using two method hasNextLine() and nextLine() to read the data sequentially.

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

public class FileReadDemo1

{

public static void main(String[] args)

{

try

{

File Obj = new File("FileDemo.txt");

Scanner obj1 = new Scanner(Obj);

while (obj1.hasNextLine())

{

String obj2 = obj1.nextLine();

System.out.println(obj2);

}

obj1.close();

}catch (FileNotFoundException e)

{

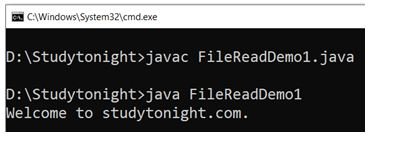
e.printStackTrace();

}

}

}

Copy



Example : Copying a file

Lets take another example to copy data of one file to another. Here we are using fileinput and output streams to read and write data. Although this is just a procedure to copy one file data to another whereas Java provides built-in methods to direct copy one file data to another file. See the below example.

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class FileCopyDemo1 {

public static void main(String[] args) {

FileInputStream a = null;

FileOutputStream b = null;

try {

File obj\_in = new File("FileDemo.txt");

File obj\_out = new File("FileDemo1.txt");

a = new FileInputStream(obj\_in);

b = new FileOutputStream(obj\_out);

byte[] buffer = new byte[1024];

int length;

while ((length = a.read(buffer)) > 0) {

b.write(buffer, 0, length);

}

a.close();

b.close();

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

} catch (IOException e) {

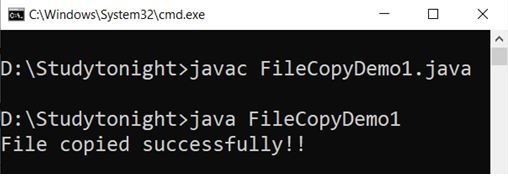
e.printStackTrace();

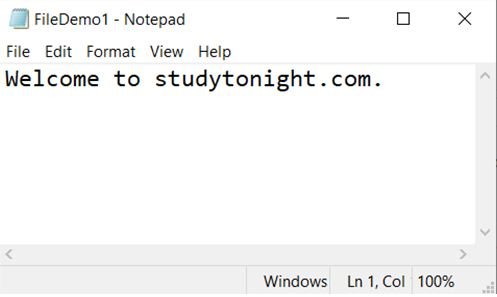
}

}

}

Copy





File permissions

We can check permissions like: reading, writing, deleting etc of a file by using the built-in methods. The File class provides methods canRead(), canWrite() etc to check whether the operation is permissible or not.

import java.io.\*;

public class FilePermissionDemo1 {

public static void main(String[] args) {

File a = new File("FileDemo1.txt");

boolean b = a.exists();

if (b == true) {

System.out.println("Executable: " + a.canExecute());

System.out.println("Readable: " + a.canRead());

System.out.println("Writable: " + a.canWrite());

} else {

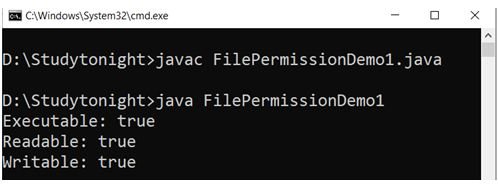
System.out.println("File not found.");

}

}

}

Copy



Retrieving file information

This is one of the important part of file handling, getting metadata of a file is necessary to keep the information about the file like: type of file, location of file, permissions of file etc. in this example, we are using some built-in methods of File class to get information about the file. See the below example

import java.io.File;

public class FileInfoDemo1

{

public static void main(String[] args)

{

File Obj = new File("FileDemo1.txt");

if (Obj.exists())

{

System.out.println("File name= " + Obj.getName());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Absolute path= " + Obj.getAbsolutePath());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Writeable= " + Obj.canWrite());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("Readable= " + Obj.canRead());

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.out.println("File size in bytes= " + Obj.length());

}

else

{

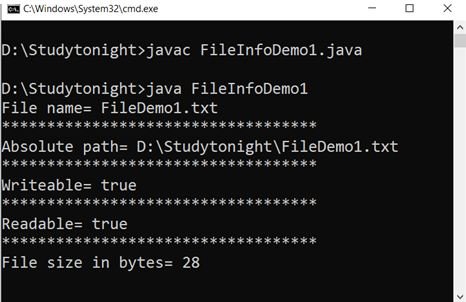
System.out.println("file does not exist.");

}

}

}

Copy



Deleting a file

In case, we need to delete a file then Java provides the method delete() that helps to delete a file using a code. In this example, we are deleting file. The method returns a boolean value to ensure that file has deleted successfully.

import java.io.File;

public class FileDeleteDemo1 {

public static void main(String[] args) {

File Obj = new File("FileDemo.txt");

if (Obj.delete()) {

System.out.println(Obj.getName() + " has been deleted");

} else {

System.out.println("Failed");

}

}

}

Copy

