National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Fall 2023, Lab Manual – 11

Course Code: CL-217	**Course : Object Oriented Programming Lab**
Instructor(s) :	**Shafique Rehman**

LAB - 11

File Handling in Java

File Handling:

File handling in java comes under IO operations. Java IO package java.io classes are specially provided for file handling in java. We will discuss some operations on filing as:

- Create file
- Delete file
- Read file
- Write file
- Change file permissions

Different streams are used for file IO operations. ;ets discuss them first then we will proceed to IO operations.

Streams in Java:

In Java, a sequence of data is known as a stream. This concept is used to perform I/O operations on a file. There are two types of streams namely input Streams & output streams

Input Streams:
The Java InputStream class is the superclass of all input streams. The input stream is used to read data from numerous input devices like the keyboard, network, etc. We will use different methods of InputStream like read and close.

```
InputStream obj = new FileInputStream(f1);
```

Output Streams:
The output stream is used to write data to numerous output devices like the monitor, file, etc. We will use different methods of OutputStream like Write and close.

```
OutputStream obj2 = new FileOutputStream(f1);
```

Create a file:

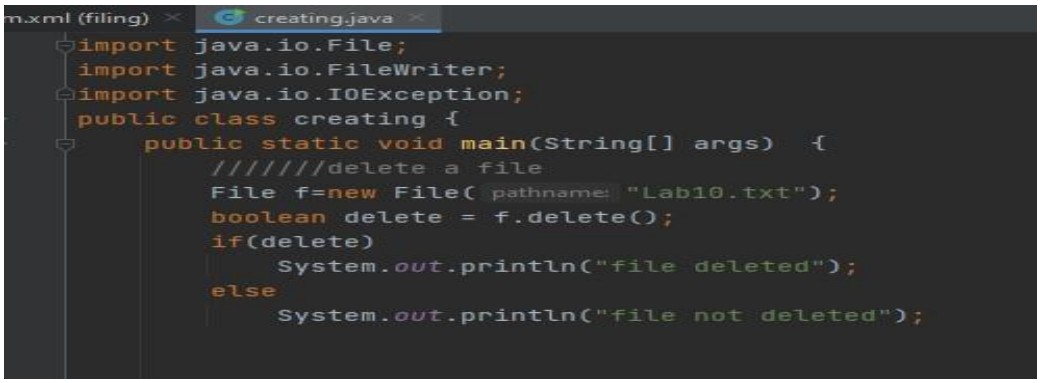
We can use File class createNewFile() method to create new file. This method returns true if file is successfully created, otherwise it returns false.

```
public class Main {
    public static void main(String[] args) {
        File file = new File( pathname: "D:\\1. Fast Data\\Fall 2023\\3. OOP BSSE-AI\\Labs\\Lab 11 - Generic and Fillng\\Lab11.txt");
       // File file = new File("Lab11.txt");
        try {
            if(file.createNewFile()){
                System.out.println("File Succesfully Created");
            }
        } catch (IOException e) {
            throw new RuntimeException("File not Created");
        }

        //System.out.println("Hello world!");
    }
}
```

Delete a file:

We can use File class delete() method to remove file.



Write a file:

We have four options for writing to a file in Java.

- FileWriter
- BufferedWriter (uses internal buffer to write data, used when you have more write operations)
- FileOutputStream (use for writing raw stream data to be written into file)
- Files (Internally, it's using OutputStream to write byte array into file).

We will uae FileWriter as FileWriter is the simplest way to write a file in Java. It provides overloaded write method to write int, byte array, and String to the File. You can also write part of the String or byte array using FileWriter. FileWriter writes directly into Files and should be used only when the number of writes is less.

Writing using BufferedWriter

```
//create a file
File files=new File( pathname: "newjavafile.txt");
files.createNewFile();
//create a Buffered writer Stream
BufferedWriter bw=new BufferedWriter(new FileWriter(files));
//write to a file
bw.write( str: "Mr. H. Potter");
bw.write( str: "4, Privet Drive");
bw.write( str: "\nLittle Winging Street,surrey");
//close the file
bw.close();
```

Read a file:

We use BufferedWriter to read file and print to console.

```
//create input stream
BufferedReader br=new BufferedReader(new FileReader( fileName: "newjavafile.txt"));
//to read the text till end of file
String buffer;
while((buffer = br.readLine())!= null){
    System.out.println(buffer);
}
    br.close();
}
```

Changing Permissions of a file:

To set and check the status for the Read Write & eXecutable permissions we can do:

```
    File f=new File( pathname: "abc.txt");
    f.createNewFile();

    f.setReadable(true);
    f.setExecutable(true);
    f.setWritable(true);

    System.out.println(f.canRead());
    System.out.println(f.canExecute());
    System.out.println(f.canWrite());
}
```