OBJECT ORIENTED PROGRAMMING USING JAVA

Lecture 10: Managing Input/ Output Files in Java

Java FileOutputStream Class: Java FileOutputStream is an output stream used for writing data to a file.

If you have to write primitive values into a file, use FileOutputStream class.

You can write byte-oriented as well as character-oriented data through FileOutputStream class.

But, for character-oriented data, it is preferred to use FileWriter than FileOutputStream

Faculty in Charge: **Prof Amey J. Shenvi Khandeparkar**

OBJECT ORIENTED PROGRAMMING USING JAVA

Lecture 10: Managing Input/ Output Files in Java

FileOutputStream class declaration:

Let's see the declaration for Java.io.FileOutputStream class:

public class FileOutputStream extends OutputStream

OBJECT ORIENTED PROGRAMMING USING JAVA

Lecture 10: FileOutputStream class methods:

Method	Description
protected void finalize()	It is used to clean up the connection with the file output stream.
void write(byte[] ary)	It is used to write ary.length bytes from the byte array to the file output stream.
void write(byte[] ary, int off, int len)	It is used to write len bytes from the byte array starting at offset off to the file output stream.
void write(int b)	It is used to write the specified byte to the file output stream.
FileChannel getChannel()	It is used to return the file channel object associated with the file output stream.
FileDescriptor getFD()	It is used to return the file descriptor associated with the stream.
void close()	It is used to closes the file output stream.

Lecture 10: Managing Input/ Output Files in Java

Java FileOutputStream Example 1: write byte

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
    public static void main(String args[]){
          try{
            FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
            fout.write(65);
            fout.close();
            System.out.println("success...");
           }catch(Exception e){System.out.println(e);}
    }
}
```

Output: Success...

The content of a text file testout.txt is set with the data A.

testout.txt, gives A

Lecture 10: Managing Input/ Output Files in Java

Java FileOutputStream example 2: write string

```
import java.io.FileOutputStream;
public class FileOutputStreamExample {
    public static void main(String args[]){
          try{
            FileOutputStream fout=new FileOutputStream("D:\\testout.txt");
            String s="Welcome to GEC.";
            byte b[]=s.getBytes();//converting string into byte array
            fout.write(b);
            fout.close();
            System.out.println("success...");
           }catch(Exception e){System.out.println(e);}
     }
}
```

Lecture 10: Managing Input/ Output Files in Java

Output:

Success...
The content of a text file testout.txt is set with the data Welcome to GEC.

testout.txt

Welcome to GEC.

Lecture 10: Managing Input/ Output Files in Java

Java FileInputStream Class:

Java FileInputStream class obtains input bytes from a file. It is used for reading byte-oriented data (streams of raw bytes) such as image data, audio, video etc.

You can also read character-stream data. But, for reading streams of characters, it is recommended to use FileReader class.

Java FileInputStream class declaration:

Let's see the declaration for java.io.FileInputStream class:

public class FileInputStream extends InputStream

OBJECT ORIENTED PROGRAMMING USING JAVA

Lecture 10: Java FileInputStream class methods

Method	Description
int available()	It is used to return the estimated number of bytes that can be read from the input stream.
int read()	It is used to read the byte of data from the input stream.
int read(byte[] b)	It is used to read up to b.length bytes of data from the input stream.
int read(byte[] b, int off, int len)	It is used to read up to len bytes of data from the input stream.
long skip(long x)	It is used to skip over and discards x bytes of data from the input stream.
FileChannel getChannel()	It is used to return the unique FileChannel object associated with the file input stream.
FileDescriptor getFD()	It is used to return the FileDescriptor object.

Lecture 10: Java FileInputStream class methods

Method	Description
protected void finalize()	It is used to ensure that the close method is call when there is no more reference to the file input stream.
void close()	It is used to closes the stream.

Lecture 10: Managing Input/ Output Files in Java

Java FileInputStream example 1: read single character:

```
import java.io.FileInputStream;
public class DataStreamExample {
    public static void main(String args[]){
        try{
          FileInputStream fin=new FileInputStream("D:\\testout.txt");
          int i=fin.read();
          System.out.print((char)i);

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

Lecture 10: Managing Input/ Output
Files in Java

Note: Before running the code, a text file named as "testout.txt" is required to be created. In this file, we are having following content:

Welcome to GEC.

After executing the above program, you will get a single character from the file which is 87 (in byte form). To see the text, you need to convert it into character

Output:

W

Java FileInputStream example 2: read all characters:

```
package com.javatpoint;

import java.io.FileInputStream;
public class DataStreamExample {
    public static void main(String args[]){
        try{
          FileInputStream fin=new FileInputStream("D:\\testout.txt");
          int i=0;
          while((i=fin.read())!=-1){
           System.out.print((char)i);
          }
          fin.close();
        }catch(Exception e){System.out.println(e);}
       }
      }
```

Output:

```
Welcome to GEC
```