File Handling

Contents

- Overview and Objectives
- File Definition
- Streams
- Class File Methods
- File Operations
- Sample Programs

Overview and Objectives

This lesson covers file handling, its operations and sample code

- Define file and its components
- Enumerate different file operations
- Know the importance of file handling in the program
- Generate program codes

File Definition

- In Java, a **File** is an abstract data type. A named location used to store related information is known as a **File.**
- With the help of File Class, we can work with files. This File Class is inside the java.io package. The File class can be used by creating an object of the class and then specifying the name of the file.

Why File Handling is Required?

- File Handling is an integral part of any programming language as file handling enables us to store the output of any particular program in a file and allows us to perform certain operations on it.
- In simple words, file handling means reading and writing data to a file.

Streams

- A series of data is referred to as **a stream**
- In Java, the concept Stream is used in order to perform I/O operations on a file

Types of Streams

- Input Stream
  - 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. InputStream is an abstract class, and because of this, it is not useful by itself.

Cont.: Types of Streams

- Output Stream
  - The output stream is used to write data to numerous output devices like the monitor, file, etc. OutputStream is an abstract superclass that represents an output stream. OutputStream is an abstract class and because of this, it is not useful by itself. However, its subclasses are used to write data.

Cont.: Types of Streams – Subclasses of Input Stream

- AudioInputStream
- ByteArrayInputStream
- FileInputStream
- FilterInputStream
- StringBufferInputStream
- ObjectInputStream

Cont.: Types of Streams – Methods in Input Stream

read()	Reads one byte of data from the input stream.
read(byte[] array)()	Reads byte from the stream and stores that byte in the specified array.
mark()	It marks the position in the input stream until the data has been read.
available()	Returns the number of bytes available in the input stream.
markSupported()	It checks if the mark() method and the reset() method is supported in the stream.
reset()	Returns the control to the point where the mark was set inside the stream.
skips()	Skips and removes a particular number of bytes from the input stream.
close()	Closes the input stream.

Cont.: Types of Streams – Subclasses of Output Stream

- ByteArrayOutputStream
- FileOutputStream
- StringBufferOutputStream
- ObjectOutputStream
- DataOutputStream
- PrintStream

Cont.: Types of Streams – Methods in Output Stream

write()	Writes the specified byte to the output stream.
write(byte[] array)	Writes the bytes which are inside a specific array to the output stream.
close()	Closes the output stream.
flush()	Forces to write all the data present in an output stream to the destination.

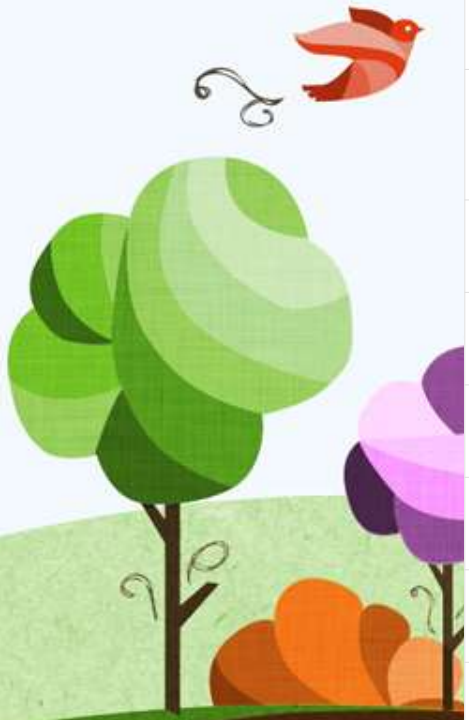
Cont.: Types of Streams Based on Data Type

- **Byte Stream:**
  - This stream is used to read or write byte data. The byte stream is again subdivided into two types which are as follows:
  - **Byte Input Stream:** Used to read byte data from different devices.
  - **Byte Output Stream:** Used to write byte data to different devices.

Cont.: Types of Streams Based on Data Type

- **Character Stream:**
  - This stream is used to read or write character data. Character stream is again subdivided into 2 types which are as follows:
  - **Character Input Stream:** Used to read character data from different devices.
  - **Character Output Stream:** Used to write character data to different devices.

Class File Methods



canRead()	It tests whether the file is readable or not.	Boolean
canWrite()	It tests whether the file is writable or not.	Boolean
createNewFile()	It creates an empty file.	Boolean
delete()	It deletes a file.	Boolean
exists()	It tests whether the file exists or not.	Boolean
length()	Returns the size of the file in bytes.	Long
getName()	Returns the name of the file.	String
list()	Returns an array of the files in the directory.	String[]
mkdir()	Creates a new directory.	Boolean
getAbsolutePath()	Returns the absolute pathname of the file.	String

File Operations

Create a File	• In order to create a file in Java, you can use the createNewFile() method • If the file is successfully created, it will return a Boolean value true and false if the file already exists
Read from a File	Use the Scanner class in order to read contents from a file
Write to a File	Use the FileWriter class along with its write() method in order to write some text to the file
Delete a File	Use the delete() method in order to delete a file

Sample Programs

```java
// Create a file
// Import the File class
import java.io.File;

// Import the IOException class to handle errors
import java.io.IOException;

public class GFG {
    public static void main(String[] args)
    {

        try {
            File Obj = new File("myfile.txt");
            if (Obj.createNewFile()) {
                System.out.println("File created: "
                                   + Obj.getName());
            }
            else {
                System.out.println("File already exists.");
            }
        }
        catch (IOException e) {
            System.out.println("An error has occurred.");
            e.printStackTrace();
        }
    }
}
```

Cont.: Sample Programs

```java
// Read from a file
// Import this class for handling errors
import java.io.FileNotFoundException;

// Import the Scanner class to read content from text files
import java.util.Scanner;

public class GFG {
    public static void main(String[] args)
    {
        try {
            File Obj = new File("myfile.txt");
            Scanner Reader = new Scanner(Obj);
            while (Reader.hasNextLine()) {
                String data = Reader.nextLine();
                System.out.println(data);
            }
            Reader.close();
        }
        catch (FileNotFoundException e) {
            System.out.println("An error has occurred.");
            e.printStackTrace();
        }
    }
}
```

Cont.: Sample Programs

```java
// Write to a file
// Import the IOException class for handling errors
import java.io.IOException;

public class GFG {
    public static void main(String[] args)
    {
        try {
            FileWriter Writer
                = new FileWriter("myfile.txt");
            Writer.write(
                "Files in Java are seriously good!!");
            Writer.close();
            System.out.println("Successfully written.");
        }
        catch (IOException e) {
            System.out.println("An error has occurred.");
            e.printStackTrace();
        }
    }
}
```

Cont.: Sample Programs

```java
// Delete a file
// Import the File class
import java.io.File;

public class GFG {
    public static void main(String[] args)
    {
        File Obj = new File("myfile.txt");
        if (Obj.delete()) {
            System.out.println("The deleted file is : "
                               + Obj.getName());
        }
        else {
            System.out.println(
                "Failed in deleting the file.");
        }
    }
}
```

Cont.: Sample Programs

```
// Read from a file and Write
import java.io.*;

public class ReadFromFileAndWrite
{
  public static void main(String args[])
  {
    InputStream istream;
    OutputStream ostream;

    File inputFile = new File("Data.txt");

    int c;
    final int EOF = -1;

    ostream = System.out;

    try
     {//provides the capability to read from a file
       istream = new FileInputStream(inputFile);

       System.out.println("Data from a File:\n");

       try
         {  while ((c= istream.read()) != EOF)
            ostream.write(c);
                }
catch(IOException e)
        {
           System.out.println(e.getMessage());
        }

      finally
      {
         try{
              istream.close();
              ostream.close();
             }
         catch(IOException e)
             {
              System.out.println("File did not close");
              }
      }
    }
    catch(FileNotFoundException e)
    {
       System.exit(1);
    }
  }
}
```

Cont.: Sample Programs

```
// Read and Write to a File
import java.io.*;

public class ReadAndWriteToFile
{
  public static void main(String args[])
  {
    InputStream istream;  // class containing methods for performing
input
    OutputStream ostream; // class containing methods for
performing output

    File outputFile = new File("Data.txt"); // class for file object
    int c;
    final int EOF = -1;

    istream = System.in;

    try
     {//provides the capability to write to file
       ostream = new FileOutputStream(outputFile);

       System.out.println("Input some names:\nPress Ctrl-Z to
terminate inputs...\n");

       try
         {  while ((c = istream.read()) != EOF)
            ostream.write(c);
                }
catch(IOException e)
        {
           System.out.println(e.getMessage());
        }

      finally
      {
         try{
              istream.close();
              ostream.close();
             }
         catch(IOException e)
             {
               System.out.println("File did not close");
             }
      }
    }
    catch(FileNotFoundException e)
    {
       System.exit(1);
    }
  }
}
```

Cont.: Sample Programs

```
// Read a File using Scanner
import java.io.*;
import java.util.*;

public class ReadingAFileScanner
{
 public static void main(String args[])
 {
   try
   { File myFile = new File("Data.txt");
     Scanner fileScan = new Scanner (myFile);

     while(fileScan.hasNext())
     {
       System.out.println(fileScan.nextLine());
     }
   }
   catch(FileNotFoundException e)
   { System.out.println(e.getMessage());
   }
 }
}
```

Cont.: Sample Programs

```
// Write to a File using BufferedWriter
import java.io.*;

public class WritingToFileBufferedWriter
{
  public static void main(String args[])
  {
    try
      {
        BufferedWriter bw = new BufferedWriter(new FileWriter("Data.txt",true));
        bw.newLine();
        bw.write("rachel a. nayre");
        bw.newLine();
        bw.write("elias a. austria");
        bw.close();
        System.out.println("Check your file!");
      }
    catch(IOException e)
      {
        System.out.println(e.getMessage());
      }
  }
}
```

Thank you very much!!!

Q & A