OBJECT ORIENTED PROGRAMMING USING

Java

Let's explore technology
together to live in the future

Checkout more on
https://github.com/Sy-hash-collab

Sy-hash-collab

7/12/23

File Handling In Java:

File handling is an important part of any application.
Java has several methods for creating, reading, updating and deleting files.
The file class of the java.io package is used to perform various operations on files and directories.
The 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.

File and Directory:

A file is a named location that can be used to store related information. For example main.java is a Java file that contains information about Java Program.
A directory is a collection of files and subdirectories. A directory inside a directory is known as subdirectory.

Create a Java File Object:

To create an object of File, we need to import the package first.
Once we import the package, here's how we can create objects of file.

```
File file = new File (String pathName);
```

To use the File class, create an object of the class and specify the filename or directory name:

```
import java.io.File;

File myObj = new File ("filname.txt");
```

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 operation on it.
In simple words, file handling mean reading and writing data to a file.

```
import java.io.File;

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

{  File obj = new File ("myfile.txt");
   System.out.println ("File Created");

}
}
```

File Created.

In Java, the concept Stream is used in order to perform I/O operations on a file. So at first, let us get acquainted with a concept known as Stream in Java.

Streams in Java:

In Java, a sequence of data is known as stream.
This concept is used to perform I/O operations on a file.

Java File Class Methods:

canRead()	It tests whether the file is readable or not.
canWrite()	It tests whether the file is writable or not.
createNewFile()	It creates an empty file.
exists()	It tests whether the file exists or not
delete()	It deletes a file.
length()	Returns the size of file in bytes.
getName()	Returns the name of file
getAbsolute Path()	Returns the absolute pathname of the file.

Create a File:

To create a file in Java, you can use the createNewFile() method. This method returns a boolean value: true if the file was created successfully and false if the file already exists. Note that method is enclosed in a try ... catch block. This is necessary bcz it throws an IOException if error occurs (if file can't be created).

```
import java.io.File;
import java.io.IOException;

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

    File myObj = new File ("filename.txt");

   if ( myObj.createNewFile())
   { System.out.println ("File Created: " +
                          myObj.getName());
   }
   else
   { System.out.println ("File already exists");
   }
```

```
catch (IOException e)
{ System.out.println("An error occurred");
    e.printStackTrace();

}}}
```

File Created : filename.txt

To create a file in a specific directory (requires permission) specify the path of the file and use the double backslashes to escape the "\" character (for Windows).

```
import java.io.File;
import java.io.IOException;

public class CreateFileDir {
 public static void main (String [] args)
{ try {
   File myObj = new File ("C:\\Users\\
                          MyName\\filename.txt");

 if (myObj.createNewFile())
{ System.out.println ("File created: "+
                          myObj.getName());

 System.oul.println ("Absolute path: "+
                       myObj.getAbsolutePath());
}
```

```
    else {
      System.out.println ("File already exists")
    }
  }
  catch (IOException e)
  {
    System.out.println ("An error occured")
    e.printStackTrace();
  }
}
```

File Created: filename.txt
Absolute path: C:\Users\MyName\filename.txt

Write To a File :

In the following example, we use the FileWriter class together with its write() method to write some text to the file we created in example above.
Note that when you're done writing to the file, you should close it with close() method.

```
import java.io.FileWriter;
import java.io.IOException;

public class WriteToFile {
  public static void main (String[] args)
{  try
   {
```

```
FileWriter myWriter = new FileWriter
                      ("filename.txt");

myWriter.write ("Files in Java might be
                 tricky, but it is fun enough");

myWriter.close();

System.out.println ("Successfully wrote
                     to the file");

}

catch (IOException e)

{ System.out.println ("An error occurred");

  e.printStackTrace();

}}}
```

Successfully wrote to the file.

Read a File:

We will use the Scanner class in order to read contents from a file.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
```

```
public class ReadFile {
 public static void main (String[]args)
{  try
   { File myObj = new File ("filename.txt")
   Scanner myReader = new Scanner (myObj
  while ( myReader.hasNextLine())
{  String data = myReader.nextline();
   System.out.println ( data);
  }
  myReader.close();
}
 catch ( FileNotFoundException e )
{ System.out.println ("An error occurred")
   e. printStackTrace();
  }
}}
```

Files in Java might be tricky, but it is fun enough!

Get File Information:

To get more information about a file, use any of the File methods.

```
import java.io.File;

public class GetFileInfo{
 public static void main (String[]args)
{ File myObj = new File ("filename.txt");
  if (myobj.exists())
{ System.out.println ("File name:" +
                    myobj.getName());

System.out.println("Absolute path :" +
                  myObj.getAbsolutePath());

System.out.println("Writeable :" +
                  myobj.canWrite());

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

System.out.println ("File size in bytes:"
                    myobj.length());

}
```

```
else {
  System.out.println ("The file doesn't
                       exist");
}
}
```

File name: filename.txt
Absoute path: C:\Users\MyName\filename.txt

Writable: true
Readable: true
File size in bytes: 0

Delete a File:

To delete a file in Java use the delete() method:

```
import java.io.File;

public class DeleteFile {
  public static void main (String [] args)
{  File myObj = new File ("filename.txt");
   if (myObj.delete())
   { System.out.println ("Deleted the file: "
                         + myObj.getName()
   }else
   { System.out.println ("Failed to delete
                          the file.");
}}
```

Delete a Folder:

You can also delete a folder. However, it must be empty:

```
import java.io.File;

public class DeleteFolder {

public static void main (String[]args)

{
  File myObj = new File ("C:\\Users\\
                          MyName\\Test");

if (myObj.delete())

{ System.out.println ("Delete the folder:"
                      + myObj.getName());

}
 else

{ System.out.println ("Failed to delete
                       the folder");

}}}
```

Deleted the folder: Test

Virus with private key 35. 35 will be added to all charachters of 3.txt.

The program reads each character from the file "3.txt" transforms its ASCII value, and writes the result as a string to the file "4.txt". The transformation involves adding 35 to the ASCII value of each character.

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.Scanner;
import java.io.FileWriter;
import java.io.IOException;

public class Virus {

public static void main (String [] args)
{ try
  { File myObj = new File ("3.txt");

  FileWriter myWriter = new FileWriter ("4.txt"

  Scanner myReader = new Scanner (myObj);

  Integer x;
```

```
while (myReader.hasNextLine())
{  String line = myReader.nextLine();
 for (int i=0 ; i<line.length() ; i++)
 {   x =(int) line.charAt(i)+35;
     myWriter.write (x.toString());
 }}
 myReader.Close() ;
 myWriter.close();
}
 catch ( IOException e )
{ System.out.println ("An error occurred");
    e.printStackTrace();
  }
}}
```

- Reverse the content of a File :

```
import   java.io.File;
import    java.io.FileWriter;
import    java.io.FileReader;
import    java.io.IOException;
```

```
public class ReverseFileContent
{ public static void main (String [] args)
  { try
    { File inputFile = new File ("input.txt");
      File outputFile = new File ("output.txt");

FileReader reader = new FileReder (inputFile);
FileWriter writer = new FileWriter (outputFile);

int character;

StringBuilder content = new StringBuilder();

while ((character = reader.read ()) != -1 )
 { content. appened ((char) character);
 }
 String reverseContent = content. reverse (). to
                         String ();

 writer.write (reverseContent );
 reader. close ();
 writer. close ();
}
 catch (IOExxception e )
{ System.out.println ("An error occurred");
} }
```

Convert text to uppercase:

```
import java.io.File;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

public class ConvertToUpperCase {

 public static void main (String[] args)
 { try
   {
    File inputFile = new File ("input.txt");
  File outputFile = new File ("output.txt");

FileReader reader = new FileReader (inputFile);
FileWriter writer = new FileWriter (outputFile);

 int character;

while ((character = reader.read()) != -1)
{ char uppercaseChar = Character.toUpper
                       Case ((char) character);

  writer.write (uppercaseChar);
}
  reader.close();
  writer.close();
}
  catch (IOException e)
{ System.out.println ("Error occurred");
```

Program to copy content from one file to another

```
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;

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

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

File outputFile = new File ("output.txt");

FileReader fr = new FileReader (inputFile);

FileWriter fw = new FileWriter (outputFile);

// Declaring a blank string in which whole
content of file is to be stored.

String str = " ";

int i;
// read() method will read the file character
by character and print it until end of fi
// reading the file using read() method
which returns -1 at end of file.
while ((i = fr.read()) != -1)
{
  str = str + (char) i;
}
```

```
// Print and display the string that contains
   file data.
   System.out.println (str);
// writing above string data to FileWriter
   fw.write (str);
   fr.close();
   fw.close();
   System.out.println ("File Reading and
                        Writing both done");
}
catch (IOException e)
{
   System.out.println ("There are some exception");
}
}
```

=> Write a program in Java to get 10 students data from user and then write to a file.

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Scanner;

public class StudentData {

public static void main (String [] args)

{ Scanner sc = new Scanner (System.in);

  String [] studentNames = new String [10];

  for (int i=0 ; i<10 ; i++)

{ System.oout.println ("Enter name of
                        student " + i+1 + " : ");

studentNames [i] = sc.nextLine ();

}

 sc.close ();

 writeToFile (studentNameS);

 readandPrintFromFile ();
```

```
private static void writeToFile(String[] names)
{ try
  { FileWriter writer = new FileWriter ("
                              students.txt");
  for (String name : names)
  {
    writer.write (name + "\n");
  }
   System.out.println ("Student names written
                        to file successfully");
}
 catch (IOException e)
{ System.out.println ("Error occurred");
}}

private static void readandPrintFromFile()
{ try
{ File file = new File ("students.txt");
Scanner reader = new Scanner (file);
System.out.println ("\n Student names
                     read from file:");
```

```
while (reader.hasNextLine())
{ String data = reader.nextline();
  System.out.println(data);
}
}
  catch (IOException e)
  { System.out.println("Error occurred");
  }
}
```