Lesson 11

File I/O

By the end of this lesson you will learn:

- Introduction to File Class
- Methods of File Class
- Creating a File
- File Write Operation
- File Read Operation

The File class from the java.io package, allows us to work with files. To use the File class, create an object of the class, and specify the filename or directory name:

import java.io.File; // Import the File class

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

- ✓ Here “filename.txt” is the file that will be created.
- ✓ This file will be created in the folder where you have kept the rest of the code.

These are some built in method of file class.

Method	Type	Description
canRead()	Boolean	Tests whether the file is readable or not
canWrite()	Boolean	Tests whether the file is writable or not
createNewFile()	Boolean	Creates an empty file
delete()	Boolean	Deletes a file
exists()	Boolean	Tests whether the file exists
getName()	String	Returns the name of the file
getAbsolutePath()	String	Returns the absolute pathname of the file
length()	Long	Returns the size of the file in bytes
list()	String[]	Returns an array of the files in the directory
mkdir()	Boolean	Creates a directory

Creating a file:

How to 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 successfully created, and false if the file already exists. The method is enclosed in a try...catch block. This is necessary because it throws an IOException if an error occurs (if the file cannot be created for some reason)

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

Following is the syntax to create a file :

File myObj = newFile("C:\\Users\\MyName\\filename.txt");

Writing in a file:

We use the FileWriter class together with its write() method to write some text to the file we created. Note that when you are done writing to the file, you should close it with the close() method.

Following is the code to write in a file:

```
public void writeInFile(String s)
    {
        try
        {
            file = new File("History.txt");
            file.createNewFile();
            writer = new FileWriter(file, true);
            writer.write(s+"\r"+"\n");
            writer.flush();
            writer.close();
        }
        catch(IOException ioe)
        {
```

```
                ioe.printStackTrace();
            }
        }
```

- ✓ Here in the first line “history” file is declared.
- ✓ Then in the next line If the file does not exists, creates and opens the file. else, just opens the file.
- ✓ After that we are using writer object to write in the file.
- ✓ After that we are writing a string s in the file.
- ✓ After writing, we need to flush to indicate that we have completed writing
- ✓ Finally we need to close the file to save writing in the file.

Reading from file: We create object of FileReader class to read some text from a file we created. We create BufferedReader object using the FileReader class object to read the file content. Note that when you are done reading to the file, you should close it with the close() method.

Following is the code to read from a file:

```
public void readFromFile()
    {
        try
        {
            reader = new FileReader(file);
            bfr = new BufferedReader(reader);
            String text="", temp;
            while((temp=bfr.readLine())!=null)
            {
                text=text+temp+"\n"+"\r";
            }
            System.out.print(text);
            reader.close();
        }
```

```
        catch(IOException ioe)
        {
            ioe.printStackTrace();
        }
    }
```

- ✓ We need to create the reader object to read from a file.
- ✓ Then creating the BufferedReader object using the reader object to read the file content.
- ✓ After that reading one line from the file, storing it in the variable temp and checking whether it is null or not.
- ✓ Then storing the temp string in text by concating it with text and "n" and "\r" is used to go to a newline.
- ✓ After that the whole line is printed in console. Finally we have to close the file.