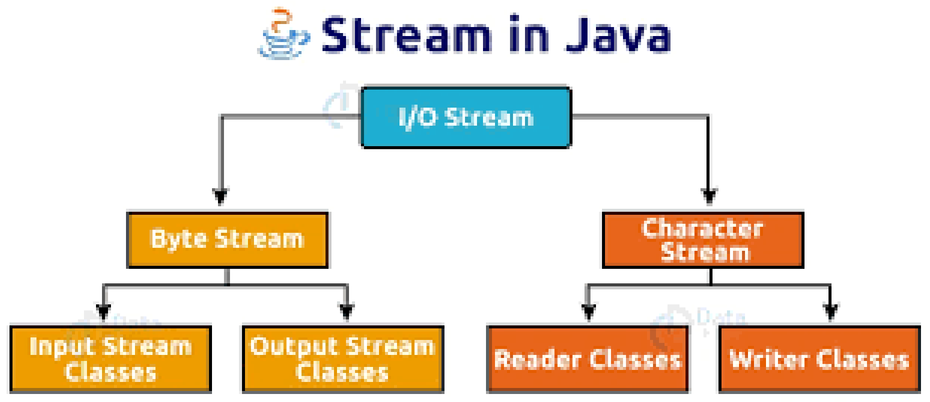
# File Handling in JAVA

DR. KRISHNENDU GUHA
ASSISTANT PROFESSOR/ LECTURER
SCHOOL OF COMPUTER SCIENCE/ INFORMATION TECHNOLOGY
UNIVERSITY COLLEGE CORK

# Contents

- Introduction
- Create files
- Write to a file
- Read files
- Delete files



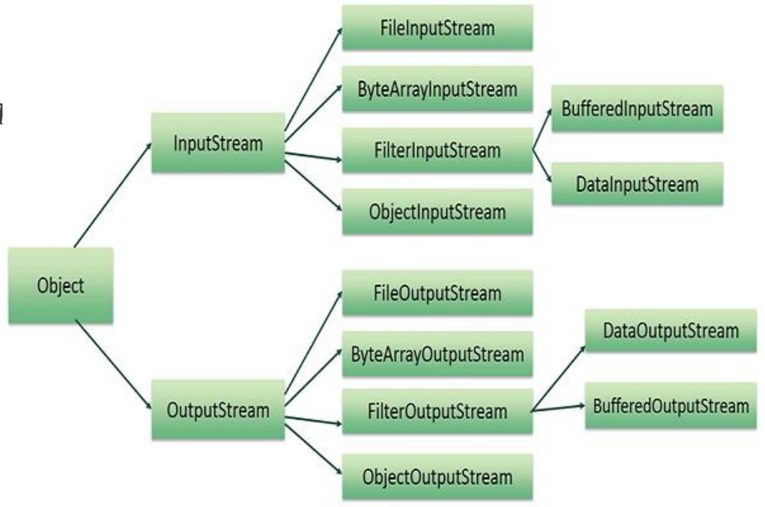
# Introduction

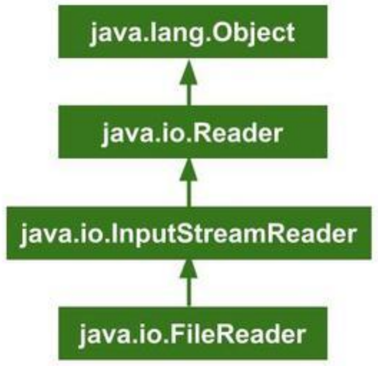
- A **file** is a ***named location that can be used to store related information***.
- For example, **main.java** is a Java file that *contains information about the Java program*.

- A **directory** is a *collection of files and subdirectories*.
- A *directory inside a directory* is known as **subdirectory**.

- 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:

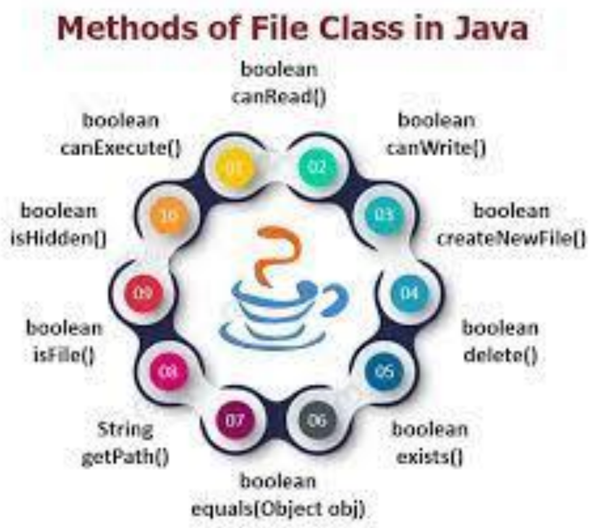
```
import java.io.File; // Import the File class
File myObj = new File("filename.txt"); // Specify the filename
```





**Useful methods in the File class for creating and getting information about files**

| **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 |



# Create

- To *create a file in Java*, we 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)

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

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();
                }
        }
}
```

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

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

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.out.println("Absolute path: " + myObj.getAbsolutePath());
						}
						else {
								System.out.println("File already exists.");
						}
				}
				catch (IOException e) {
						System.out.println("An error occurred.");
						e.printStackTrace();
				}
		}
}
```

## Write to a File

The different ways of writing into a file in Java are

- Using writeString() method
- Using BufferedWriter Class
- Using FileOutputStream Class
- Using FileWriter Class

## Using writeString() method

```
// Importing required classes
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;

// Main class
public class Main {

    // Main driver method
    public static void main(String[] args)
        throws IOException
    {
        // Assigning the content of the file
        String text
            = “This is UCC";

        // Defining the file name of the file
        Path fileName = Path.of(
            "/Users/UCC/Desktop/demo.docx");

// Writing into the file
        Files.writeString(fileName, text);

        // Reading the content of the file
        String file_content = Files.readString(fileName);

        // Printing the content inside the file
        System.out.println(file_content);
    }
}
```

## Using BufferedWriter Class

```
// Importing java input output libraries
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;

// Main class
public class Main {
    // Main driver method
    public static void main(String[] args)
    {
        // Assigning the file content
        // Note: Custom contents taken as input to
        // illustrate
        String text
            = “This is UCC";
// Try block to check for exceptions
        try {
            // Step 1: Create an object of BufferedWriter
            BufferedWriter f_writer
                = new BufferedWriter(new FileWriter(
                    "/Users/UCC/Desktop/demo.docx"));
            // Step 2: Write text(content) to file
            f_writer.write(text);

            // Step 3: Printing the content inside the file
            // on the terminal/CMD
            System.out.print(text);

            // Step 4: Display message showcasing
            // successful execution of the program
            System.out.print(
                "File is created successfully with the content.");

            // Step 5: Close the BufferedWriter object
            f_writer.close();
        }

        // Catch block to handle if exceptions occurs
        catch (IOException e) {

            // Print the exception on console
            // using getMessage() method
            System.out.print(e.getMessage());
        }
    }
}
```

## Using FileOutputStream Class

```
// Importing java input output classes
import java.io.FileOutputStream;
import java.io.IOException;
 
  public class Main {
 
      // Main driver method
    public static void main(String[] args)

    {
        // Assign the file content
        String fileContent = “This is UCC";
        FileOutputStream outputStream = null;
 
          // Try block to check if exception occurs
        try {
 
              // Step 1:  Create an object of FileOutputStream
            outputStream = new FileOutputStream("file.txt");
 
              // Step 2: Store byte content from string
            byte[] strToBytes = fileContent.getBytes();
 
              // Step 3: Write into the file
            outputStream.write(strToBytes);
        // Print the success message (Optional)
        System.out.print(
            "File is created successfully with the content.");
    }

    // Catch block to handle the exception
    catch (IOException e) {

        // Display the exception/s
        System.out.print(e.getMessage());
    }

    // finally keyword is used with in try catch block
    // and this code will always execute whether
    // exception occurred or not
    finally {

        // Step 4: Close the object
        if (outputStream != null) {
```

```
// Note: Second try catch block ensures that
                // the file is closed even if an error
                // occurs
                try {

                    // Closing the file connections
                    // if no exception has occurred
                    outputStream.close();
                }

                catch (IOException e) {

                    // Display exceptions if occurred
                    System.out.print(e.getMessage());
                }
            }
        }
    }
}
```

We use the FileWriter class together with its write() method to write some text to the file that has been created.

Note that after we are done writing to the file, we should close it with the close() method:

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

public class WriteToFile {
        public static void main(String[] args) {
                try {
                        FileWriter myWriter = new FileWriter("filename.txt");
                        myWriter.write(“This is UCC");
                        myWriter.close();
                        System.out.println("Successfully wrote to the file.");
                }
                catch (IOException e) {
                        System.out.println("An error occurred.");
                        e.printStackTrace();
                }
        }
}
```

## Read

The different ways of reading a text file in Java are

- Using BufferedReader class
- Using File Reader class
- Reading the whole file in a List
- Read a text file as String
- Using Scanner Class

- Using **Buffered Reader Class**

BufferedReader in = new BufferedReader(Reader in, int size);

```
// Importing input output classes
import java.io.*;
 // Main class
public class Main {
    // main driver method
   public static void main(String[] args) throws Exception
   {
       // File path is passed as parameter
      File file = new File(
          "C:\\Users\\UCC\\Desktop\\test.txt");
     // Creating an object of BufferedReader class
      BufferedReader br
          = new BufferedReader(new FileReader(file));

// Declaring a string variable
      String st;
      // Condition holds true till
      // there is character in a string
      while ((st = br.readLine()) != null)

          // Print the string
          System.out.println(st);
    }
}
```

Using **File Reader Class**

- *Convenience class for reading character files*. The constructors of this class assume that the default character encoding and the default byte-buffer size are appropriate.
- Constructors defined in this class are as follows:

1. **FileReader(File file):** Creates a new FileReader, given the File to read from
2. **FileReader(FileDescriptor fd):** Creates a new FileReader, given the FileDescriptor to read from
3. **FileReader(String fileName):** Creates a new FileReader, given the name of the file to read from

```
// Importing input output classes
import java.io.*;
public class Main {
       // Main driver method
    public static void main(String[] args) throws Exception
    {
        // Passing the path to the file as a parameter
        FileReader fr = new FileReader(
            "C:\\Users\\UCC\\Desktop\\test.txt");
        // Declaring loop variable
        int i;
        // Holds true till there is nothing to read
        while ((i = fr.read()) != -1)

            // Print all the content of a file
            System.out.print((char)i);
    }
}
```

## Reading whole file in a list

```java
import java.util.*;
import java.nio.charset.StandardCharsets;
import java.nio.file.*;
import java.io.*;
public class ReadFileIntoList
{
  public static List<String> readFileInList(String fileName)
  {

    List<String> lines = Collections.emptyList();
    try
    {
      lines =
       Files.readAllLines(Paths.get(fileName), StandardCharsets.UTF_8);
    }
catch (IOException e)
    {

      // do something
      e.printStackTrace();
    }
    return lines;
  }
  public static void main(String[] args)
  {
    List l = readFileInList("C:\\Users\\UCC\\Desktop\\test.java");

    Iterator<String> itr = l.iterator();
    while (itr.hasNext())
      System.out.println(itr.next());
  }
}
```

## Read a text file as String

```
package io;

import java.nio.file.*;;

public class ReadTextAsString {

 public static String readFileAsString(String fileName)throws Exception
 {
  String data = "";
  data = new String(Files.readAllBytes(Paths.get(fileName)));
  return data;
 }


 public static void main(String[] args) throws Exception
 {
  String data = readFileAsString("C:\\Users\\UCC\\Desktop\\test.java");
  System.out.println(data);
 }
}
```

**Scanner** class to read the contents of the text file we created

```
import java.io.File; // Import the File class
import java.io.FileNotFoundException; // Import this class to handle errors
import java.util.Scanner; // Import the Scanner class to read text files

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();
                }
        }
}
```

## Delete Files

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

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

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

- We 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("Deleted the folder: " + myObj.getName());
                }
                else {
                        System.out.println("Failed to delete the folder.");
                }
        }
}
```

Thank You
ANY QUESTIONS?