

Working with **Files** in JAVA

- A **file** is a named location that can be used to store related information.

For example:

“**Sample.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**.
- In Java, with the help of **File Class**, we can work with files.
This **File Class** is inside the **java.io** package.

File operations in Java

- The following are the several operations that can be performed on a file in Java :
 - **Create a File**
 - **Read from a File**
 - **Write to a File**
 - **Delete a File**
- To create perform **File operations**, we need to **create object** of File.

Format to create File Object:

```
File fileobj = new File(String path/FileName);
```

Example:

```
File obj = new File("newFile.txt");
```

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.
- **Note** that 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).

Example Program:

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

public class FileCreationDemo {
    public static void main(String[] args)
    {
        // creating File object
        File my_obj = new File("Samplefile.txt");
        try {
            if (my_obj.createNewFile()) {
                System.out.println("File Successfully created : "+ my_obj.getName());
            }
            else {
                System.out.println("File already exists.");
            }
        }
        catch (IOException e) {
            System.out.println("An error has occurred.");
            e.printStackTrace();
        }
    }
}
```

Read from a File

- To read data from the file, we can use either **Scanner** or **FileReader** class.
- To read data from the file using file **FileReader** class ,You can use **read()** method. It reads the data character by character.

```
FileReader obj = new FileReader("Samplefile.txt");  
// Reads characters  
obj.read(my_array);
```

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

```
File my_obj = new File("Samplefile.txt");  
Scanner sc_R = new Scanner(my_obj);
```

Example Program: Scanner Class

```
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 FileReadingDemo {
    public static void main(String[] args) {
        try {
            File my_obj = new File("Samplefile.txt");
            Scanner sc_R = new Scanner(my_obj);
            while (sc_R.hasNextLine()) {
                String data = sc_R.nextLine();
                System.out.println(data);
            }
            sc_R.close();
        } catch (FileNotFoundException e) {
            System.out.println("An error occurred.");
            e.printStackTrace();
        }
    }
}
```

Example Program: **FileReader** Class

```
// importing the FileReader class
import java.io.FileReader;

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

        char[] my_arr = new char[200];
        try {
            // Creates a reader using the FileReader
            FileReader f_read = new FileReader("Samplefile.txt");

            // Reads characters
            f_read.read(my_arr);
            System.out.println("Data in the file:");
            System.out.println(my_arr);

            // Closes the reader
            f_read.close();
        }
        catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

Write to a File

- We use the **FileWriter** class along with its **write()** method in order to write some text to the file.

```
// Creates a writer using the FileWriter
FileWriter f_wr = new FileWriter("Samplefile.txt");
// to write the text
f_wr.write("Working with Files in Java!!");
// to close the writer
f_wr.close();
```

Note : You should close it with the **close()** method

Example Program: **FileWriter** Class

```
// Import the FileWriter class
import java.io.FileWriter;
// Import the IOException class for handling errors
import java.io.IOException;

public class FileWritingDemo {
    public static void main(String[] args)
    {
        try {
            // Creates a writer using the FileWriter
            FileWriter f_wr = new FileWriter("Samplefile.txt");
            // to write the data
            f_wr.write("Working with Files in Java!!");
            // to close the writer
            f_wr.close();
            System.out.println("Data Written Successfully.");
        }
        catch (IOException e) {
            System.out.println("An error has occurred.");
            e.printStackTrace();
        }
    }
}
```

Delete to a **File**

- We can use the **delete()** method of the **File** class to delete the specified **file** or **directory**.
- It returns
 true if the file is deleted.
 false if the file does not exist.

Note: We can only delete empty directories.

Example Program:

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

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

File Operation Methods :

Operation	Method	Package
To create file	<code>createNewFile()</code>	<code>java.io.File</code>
To read file	<code>read()</code>	<code>java.io.FileReader</code>
To write file	<code>write()</code>	<code>java.io.FileWriter</code>
To delete file	<code>delete()</code>	<code>java.io.File</code>