# CSCI 2120:
# Software Design & Development II

*UNIT3: I/O management*

*io api*

**FileWriter**

## Overview

## Introduction

- **FileWriter in Java** is an output stream that writes data in the form of characters into the text file.

- In other words, `FileWriter` is a character-based output stream that writes characters into a text file using the platform’s default character encoding and buffer size.

## Introduction

`FileWriter` is useful when we want:

- to write text into a specific file.
- to append text in a file.
- to copy text from one file to another file.

For writing characters in files, Java provides `FileWriter` class.

# FileWriter class declaration

FileWriter class is a subclass of OutputStreamWriter class that extends Writer class. It implements Closeable, Flushable, Appendable, and AutoCloseable interfaces.

The general syntax to declare FileWriter class in Java is given below:

```
public class FileWriter
       extends OutputStreamWriter
       implements Closeable, Flushable, Appendable, AutoCloseable
```

Appendable is present in java.lang package. It defines an object to which characters can be added by using the append( ) method.

## FileWriter class declaration

The inheritance diagram for `FileWriter` class is as follows:

```
java.lang.Object
   java.io.Writer
       java.io.OutputStreamWriter
           java.io.FileWriter
```

# FileWriter Constructors

**1. FileWriter(File file):**
This constructor creates a `FileWriter` object with the specified `File` to write, using the platform's default charset. Here, file specifies the `File` object that describes the file.

The syntax to create a `FileWriter` object with the specified `File` object is as follows:

```
File file = new File("myfile.txt");
FileWriter fw = new FileWriter(file);
```

## FileWriter Constructors

**2. FileWriter(FileDescriptor fd):**
This constructor creates a `FileWriter` object with a specified file descriptor, using the platform's default charset.

# FileWriter Constructors

**3. FileWriter(File file, boolean append):**
This constructor creates a `FileWriter` object with the specified `File` to write and a `boolean` indicating whether to append the data written, using the platform's default charset.

If `append` is `true`, then the contents of a preexisting file will be preserved and the output is appended to the end of the file. This is useful when we want to add to an existing file.

Otherwise, when the `append` is `false`, the contents of any preexisting file with the same name will be destroyed.

The syntax to create a `FileWriter` object with a specified file object is as:

```
File file = new File("myfile.txt");
FileWriter fw = new FileWriter(file, true);
```

## FileWriter Constructors

**4. FileWriter(File file, Charset charset):**

This constructor creates a `FileWriter` object with the specified `File` to write and `charset`.

## FileWriter Constructors

**5. FileWriter(File file, Charset charset, boolean append):**
This form of constructor creates a `FileWriter` object with given the `File` to write, `charset`, and a `boolean` indicating whether to append the data written.

## FileWriter Constructors

**6. FileWriter(String fileName):**
This constructor creates a `FileWriter` object with the specified file name, using the platform's default charset. Any text written in the file is overwritten.

The general syntax to create a `FileWriter` with a specified file name is as follows:

```
FileWriter fw = new FileWriter(String fileName);

//For example:
FileWriter fw = new FileWriter("myfile.txt");
```

# FileWriter Constructors

**7. FileWriter(String fileName, boolean append):**

This constructor creates a `FileWriter` object with a file name and a boolean indicating whether to append the data written, using the platform’s default charset.

If the append is true, the new text is appended to the existing content of the file rather than overwriting them by setting the second argument true.

The general syntax to create `FileWriter` object is as follows:

```
FileWriter fw = new FileWriter(String fileName, boolean append);

//For example:
FileWriter fw = new FileWriter("myfile.txt", true);
```

## FileWriter Constructors

**8. FileWriter(String fileName, Charset charset):**
This constructor creates a `FileWriter` object with given a file name and `charset`.

# FileWriter Constructors

**9. FileWriter(String fileName, Charset charset, boolean append):**
This overloaded form of constructor creates a `FileWriter` object with given a file name, charset, and a boolean indicating whether to append the data written.

If the named file does not exist, this form of constructor throws an exception named `IOException`.

## FileWriter Constructors

**Note**:

The `FileWriter` class constructor internally creates a `FileOutputStream` to write bytes to the specified file. With the help of its superclass `OutputStreamWriter,` it converts Unicode characters written to the stream into bytes using the default encoding of default locale.

## FileWriter Methods

`FileWriter` class does not define any methods of its own. It simply inherits them from its superclass.

1. **Methods** inherited from class `java.io.OutputStreamWriter`:

   `flush, getEncoding, write, write, write`

2. **Methods** inherited from class `java.io.Writer`:

   `append, append, append, close, nullWriter, write, write`

# Example 1:  Write String to File

1. Let’s create a simple program where we will write a text into a file. We will use `write()` method to write into a file inherited from `Writer` class.

## Example 1: Write String to File

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

public class FileWriterTester1 {
   public static void main(String[] args) throws IOException {
       // Create a FileWriter object to open the file.
       FileWriter fw = new FileWriter("./src/myfile.txt");

       // To write text to the file, call write() method inherited from Writer class.
       fw.write("Welcome to UNO CSCI \n");
       fw.write("I love Java Programming");

       fw.close(); // Closing the file.
       System.out.println("Successfully written...");
   }
}
```

## Example 1:  Write String to File

**Output:**

```
Successfully written...
```

**myfile.txt:**

```
Welcome to UNO CSCI
I love Java Programming.
```

**Explanation:**

In this program, we have created a `FileWriter` object, specifying the name of file in the form of either `String` or `File` reference. The `write()` method inherited from `Writer` class has been used to write lines of text.

**Note:**

If you are writing multiple lines of text, must use newline. Without the newline, the next string would start on the same line, immediately after the previous one.

## Example 2:  Write String[ ] Array to File

2. Let’s create another program where we will write an array of strings to a file. Look at the program source code to understand better.

# Example 2:  Write String[ ] Array to File

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

public class FileWriterTester2 {
   public static void main(String[ ] args) throws IOException {
       String strs[ ] = {
               "This is a dog",
               "This is a cat",
               "This is an elephant",
               "This is a lion"
       };

       // Create a FileWriter object to open file.
       FileWriter fw = new FileWriter("./src/myfile.txt");

       // To write an array of Strings to the file, call write() method inherited from Writer class.
       for(int i = 0; i < strs.length; i++) {
           fw.write(strs[i]); // write line to file
           fw.write("\n"); // output a newline
       }
       fw.close(); // Closing the file.
       System.out.println("Successfully written...");
   }
}
```

# Example 2: Write String[ ] Array to File

### Output:

```
Successfully written...
```

### myfile.txt:

```
This is a dog
This is a cat
This is an elephant
This is a lion
```

### Explanation:

In this program, we have created a `FileWriter` object, specifying the name of file in the form of either `String` or `File` reference. The `write()` method inherited from `Writer` class has been used to write lines of text.

### Note:

If you are writing multiple lines of text, must use newline. Without the newline, the next string would start on the same line, immediately after the previous one.

## Example 3:  Write File to File

3. Let’s create a program where we will copy lines of text from one file to another file in a very simple process. The contents written in the input.txt file are as follows:

- This is an apple.
- This is an orange.
- This is papaya.
- This is a mango.

# Example 3:  Write File to File

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

public class FileWriterTester3 {
   public static void main(String[] args) throws IOException {
       File inFile = new File("./src/input.txt");
       File outFile = new File("./src/output.txt");

       FileReader fr = new FileReader(inFile);
       FileWriter fw = new FileWriter(outFile);

       // Read and write till the end.
       int ch;
       while((ch = fr.read()) != -1) {
           fw.write(ch);
       }
       fr.close();
       fw.close();
   }
}
```

# Example 3: Write File to File

## Output:

```
```

## output.txt:

```
This is an apple.
This is an orange.
This is papaya.
This is a mango.
```

## Explanation - (Part 1):

In this simple program, we have created two file objects inFile and outFile and initializes them with `“input.txt”` and `“output.txt”` respectively using the following code:

```
File inFile = new File("./src/input.txt");
File outFile = new File("./src/output.txt");
```

Then, we have created two file stream objects `fr` and `fw`, and connected them to the named files using the following code:

```
FileReader fr = new FileReader(inFile);
FileWriter fw = new FileWriter(outFile);
```

Connecting of `inFile` with FileReader and `outFile` with FileWriter means files `“input.txt”` and `“output.txt”` are opened.

# Example 3:  Write File to File

## Output:

## output.txt:

```
This is an apple.
This is an orange.
This is papaya.
This is a mango.
```

## Explanation - (Part 2):

The statement `ch = fr.read();` reads a character from the inFile through input stream fr and assigns it to the variable ch.

Similarly, the statement `fw.write(ch);` writes the character stored in the variable `ch` to the `outFile` through the output stream `fw`.

The character `-1` indicates the end of file. When the end of file is reached, the statements `fr.close()` and `fw.close()` close the input and output streams.

Thus, we have performed reading and writing characters using file streams and file objects in this program.

## Example 3: Write File to File

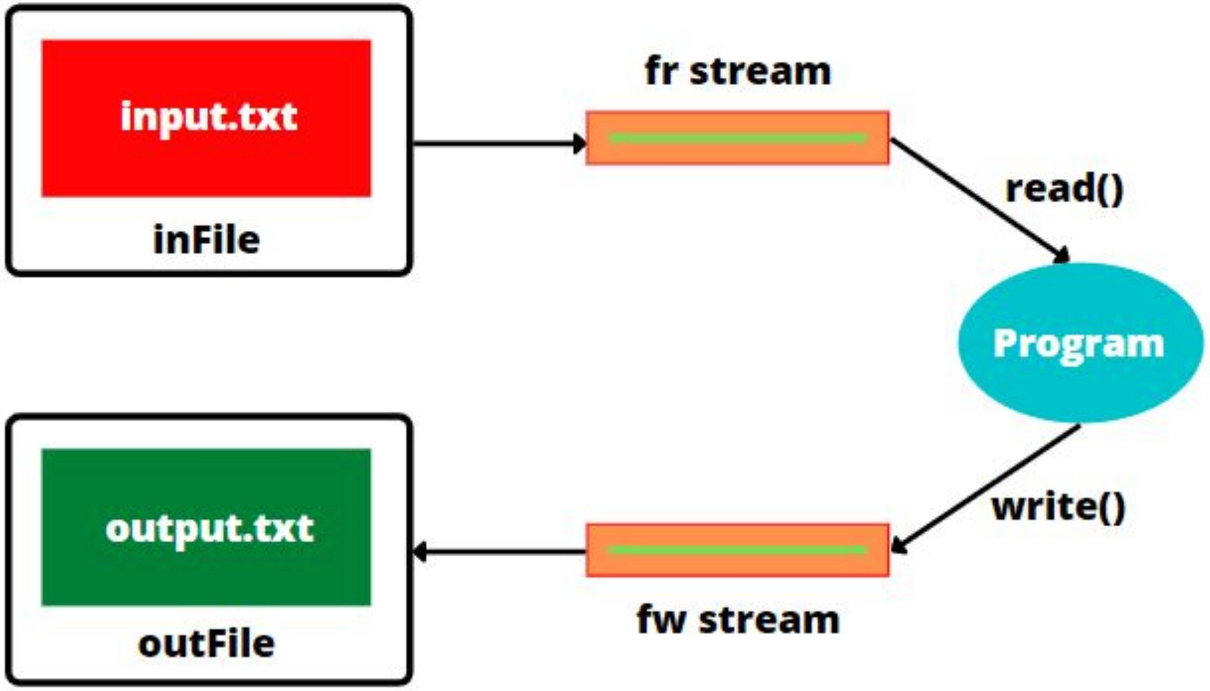

Fig: Reading from and writing to files

END