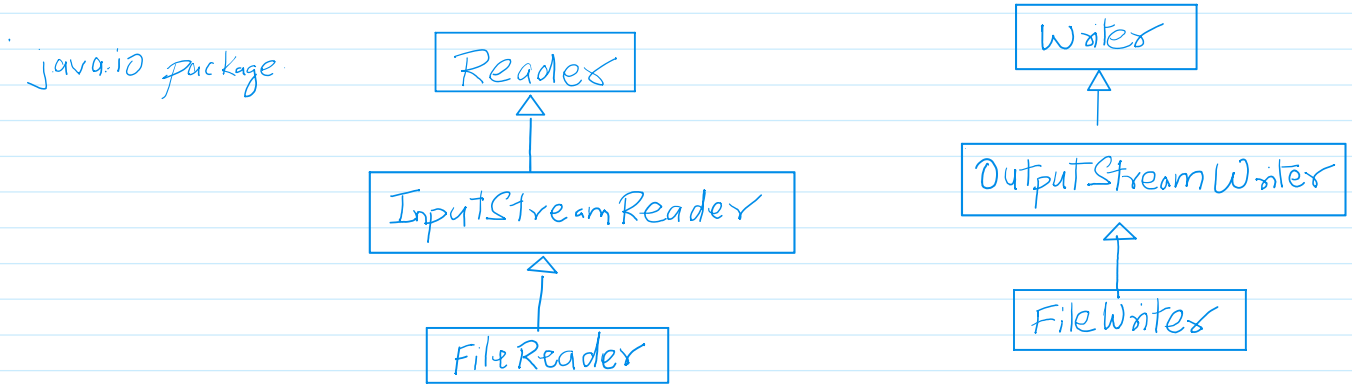
FileReader and FileWriter

22 September 2025 08:29

In Java, **FileReader** and **FileWriter** are classes provided in the java.io package. To handle character streams for reading and writing.

- **FileReader:** Used to read data from file character by character.
- **FileWriter:** He used to write data to a file character by character.

These classes are specially designed for text files (That is. Files containing character data like .txt, .csv).



FileReader class:- The file reader class is used for reading character data from a file.

- Reads character-by-character or character arrays.
- Best suited for reading text files.
- Returns data in the form of int (ASCII code of the character.) When reading character by character.

Constructors

Constructor	Description
`FileReader(File file)`	Creates a new FileReader, given the File to read, using the default charset.
`FileReader(FileDescriptor fd)`	Creates a new FileReader, given the FileDescriptor to read, using the default charset.
`FileReader(File file, Charset charset)`	Creates a new FileReader, given the File to read and the charset.
`FileReader(String fileName)`	Creates a new FileReader, given the name of the file to read, using the default charset.
`FileReader(String fileName, Charset charset)`	Creates a new FileReader, given the name of the file to read and the charset.

```
public class FileReaderExample
{
    public static void main(String[] args){
        String fileName, path;
        path = "./DataFiles/";
        System.out.print("\fEnter your file name: ");
        fileName = new java.util.Scanner(System.in).nextLine();
        try(FileReader fr = new FileReader(path+fileName)){
            for(int ch; (ch = fr.read()) != -1; System.out.print((char)ch));
        }
        catch(IOException ioe){
            System.err.print("\nNetwork error!");
        }
    }
}
```

File:
If

If you can keep your head when all about you
Are losing theirs and blaming it on you,
If you can trust yourself when all men doubt you,
But make allowance for their doubting too;
If you can wait and not be tired by waiting,
Or being lied about, don't deal in lies,
Or being hated, don't give way to hating,
And yet don't look too good, nor talk too wise:

If you can dream—and not make dreams your master;
If you can think—and not make thoughts your aim;
If you can meet with Triumph and Disaster
And treat those two impostors just the same;
If you can bear to hear the truth you've spoken
Twisted by knaves to make a trap for fools,
Or watch the things you gave your life to, broken,
And stoop and build 'em up with worn-out tools:

If you can make one heap of all your winnings
And risk it on one turn of pitch-and-toss,
And lose, and start again at your beginnings
And never breathe a word about your loss;
If you can force your heart and nerve and sinew
To serve your turn long after they are gone,
And so hold on when there is nothing in you
Except the Will which says to them: 'Hold on!'

If you can talk with crowds and keep your virtue,
Or walk with Kings—nor lose the common touch,
If neither foes nor loving friends can hurt you,
If all men count with you, but none too much;
If you can fill the unforgiving minute
With sixty seconds' worth of distance run,
Yours is the Earth and everything that's in it,
And—which is more—you'll be a Man, my son!

By Rudyard Kipling

Home work:
WAP to create a list of poems, user will select the list and read poem, and user can select another poem to read also. When user will select exit. Then your code will exit by "Keep reading".

1. Poem1(Write name of poem along with poet name)
2. Poem2(Write name of poem along with poet name)
3. Poem3(Write name of poem along with poet name)
4. …
5. Press x or X to exit.

Multiple text files of poem in your directory and read those files depending upon users choice.

FileWriter Class
The FileWriter class is used for writing character data to a file.

- Writes character-by-character or strings.

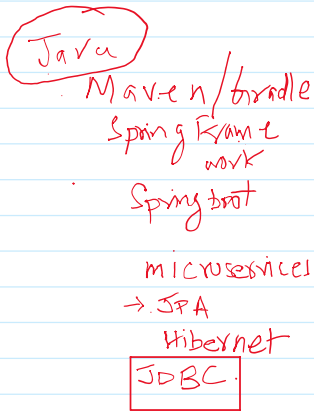
- If the file does not exist -> it will be created automatically.
- If the file exists:
  - Default behavior: Overwrites the file content.
  - Append mode: to add content without overwriting, use the constructor with a second argument as true.

Constructors

Constructor	Description
`FileWriter(File file)`	Constructs a FileWriter given the File to write, using the default charset
`FileWriter(FileDescriptor fd)`	Constructs a FileWriter given a file descriptor, using the default charset.
`FileWriter(File file, boolean append)`	Constructs a FileWriter given the File to write and a boolean indicating whether to append the data written, using the default charset.
`FileWriter(File file, Charset charset)`	Constructs a FileWriter given the File to write and charset.
`FileWriter(File file, Charset charset, boolean append)`	Constructs a FileWriter given the File to write, charset and a boolean indicating whether to append the data written.
`FileWriter(String fileName)`	Constructs a FileWriter given a file name, using the default charset
`FileWriter(String fileName, boolean append)`	Constructs a FileWriter given a file name and a boolean indicating whether to append the data written, using the default charset.
`FileWriter(String fileName, Charset charset)`	Constructs a FileWriter given a file name and charset.
`FileWriter(String fileName, Charset charset, boolean append)`	Constructs a FileWriter given a file name, charset and a boolean indicating whether to append the data written.

True means append mode is on.

UTF-8



`FileWriter` class uses its parent class methods.

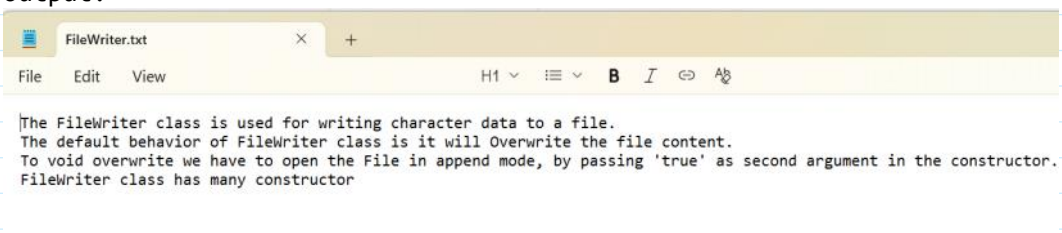
```
package FileHandling;
import java.util.*;
import java.io.*;

/**
 * Write a description of class FileWriterExample here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class FileWriterExample
{
   public static void main(String args[]){
      Scanner sc = new Scanner(System.in);
      String path="./DataFiles/";
      System.out.print("\fEnter filename: ");
      String fileName = sc.nextLine();

      try(FileWriter fw = new FileWriter(path+fileName, true)){
         while(true){
            System.out.print("\nEnter a line or type 'stop': ");
            String line = sc.nextLine();
            if(line.equalsIgnoreCase("stop")){
               break;
            }
            fw.write(line+"\n");
         }
      }
      catch(IOException ioe){
         System.err.print("\nNetwork error!");
```

```
    }
  }
}
```

Output:



If I need to copy the content of FileWriter.txt to CopyFileWriter.txt file. Write down Java code do this operation.

BufferedWriter class: Java BufferedWriter class is used to provide buffering for Writer instances. It inherits Writer class. The buffering characters are used for providing the efficient writing of a single arrays, characters and strings.

Constructors

Constructor	Description
`BufferedWriter(Writer out)`	Creates a buffered character-output stream that uses a default-sized output buffer.
`BufferedWriter(Writer out, int sz)`	Creates a new buffered character-output stream that uses an output buffer of the given size.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`close()`	Closes the stream, flushing it first.
void	`flush()`	Flushes the stream.
void	`newLine()`	Writes a line separator.
void	`write(char[] cbuf, int off, int len)`	Writes a portion of an array of characters.
void	`write(int c)`	Writes a single character.
void	`write(String s, int off, int len)`	Writes a portion of a String.

```
package FileHandling;
import java.util.*;
import java.io.*;


/**
 * Write a description of class BufferedWriterExample here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class BufferedWriterExample
{
   public static void main(String args[]){
      Scanner sc = new Scanner(System.in);
      String path="./DataFiles/";
      System.out.print("\fEnter filename: ");
      String fileName = sc.nextLine();

      try(FileWriter fw = new FileWriter(path+fileName, true);
```

```
      BufferedWriter bw = new BufferedWriter(fw)){
        while(true){
          System.out.print("\nEnter a line or type 'stop': ");
          String line = sc.nextLine();
          if(line.equalsIgnoreCase("stop")){
            break;
          }
          fw.write(line+"\n");
        }
      }
      catch(IOException ioe){
        System.err.print("\nNetwork error!");
      }
    }
}
```

BufferedReader class

Written down here among the many tools available within Java's extensive standard library. The BufferedReader class stands out as a versatile and efficient means of reading character input from various sources. Java Buffer reader class is used to read the text from the character based input stream. It can be used to read data line by line by readLine() method. It makes the performance fast. It inherits the Reader class.

Constructors

Constructor	Description
`BufferedReader(Reader in)`	Creates a buffering character-input stream that uses a default-sized input buffer.
`BufferedReader(Reader in, int sz)`	Creates a buffering character-input stream that uses an input buffer of the specified size.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
void	`close()`	Closes the stream and releases any system resources associated with it.
`Stream<String>`	`lines()`	Returns a `Stream`, the elements of which are lines read from this `BufferedReader`.
void	`mark(int readAheadLimit)`	Marks the present position in the stream.
boolean	`markSupported()`	Tells whether this stream supports the mark() operation, which it does.
int	`read()`	Reads a single character.
int	`read(char[] cbuf, int off, int len)`	Reads characters into a portion of an array.
String	`readLine()`	Reads a line of text.
boolean	`ready()`	Tells whether this stream is ready to be read.
void	`reset()`	Resets the stream to the most recent mark.

```
package FileHandling;
import java.io.*;


/**
 * Write a description of class BufferedReaderExample here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class BufferedReaderExample
```

```
{
   public static void main(String[] args){
      BufferedReader br = new BufferedReader(
      new InputStreamReader(System.in));
      String path = "./DataFiles/";
      try
      {
         System.out.print("\fEnter file name: ");
      → String fileName = br.readLine();
         try(FileReader fr = new FileReader(path+fileName);
         BufferedReader buffer = new BufferedReader(fr)){
            for(int i = 0; (i = buffer.read())!=-1; System.out.print((char)i));
         }
      }
      catch (IOException ioe)
      {
         System.err.print("\nNetwork Error!");
      }
   }
}
```

Console input it is always in the form of String.

Must kept under try .. catch block. because it throws → IOException object.