CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

BufferedWriter

Overview

1. Introduction
2. BufferedWriter class declaration
3. BufferedWriter constructors
4. BufferedWriter methods
5. BufferedWriter examples

Introduction

- **BufferedWriter in Java** is a `Writer` that buffers the stream of characters before writing them to an underlying output stream.

- It adds the buffering capability to the underlying output character stream, so that there is no need to access the underlying file system for each read and write operation.

- When a program writes to a `BufferedWriter`, the text is stored in the buffer.

- When the buffer is filled up or explicitly flushed, the text is moved to the underlying output stream that makes the writes much faster and improves the performance.

Bufferedwriter class declaration

`BufferedWriter` class is a subclass of `Writer` *class* that extends `Object` class. It implements `Closeable, Flushable, Appendable`, and `AutoCloseable` interfaces.

The general syntax to declare `BufferedWriter` class in Java is as follows:

```
public class BufferedWriter
       extends Writer
       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.

Bufferedwriter class declaration

The inheritance hierarchy for `BufferedWriter` class is as follows:

```
java.lang.Object
   java.io.Writer
       java.io.BufferedWriter
```

`BufferedWriter` class was added in Java 1.1 version. It is defined in the `java.io.BufferedWriter` package that is imported into the program before using it.

BufferedWriter Constructors

A `BufferedWriter` class has **two** constructors in Java that allow passing a `Writer` object.

They are as follows:

BufferedWriter Constructors

1. BufferedWriter(Writer out):
This constructor creates a buffered writer object that buffers the character output stream specified by out. It uses a default buffer size.

The general syntax to create `BufferedWriter` object with default size is as follows:

```
BufferedWriter bw = new BufferedWriter(Writer out);
```

For example, we can wrap the FileWriter with BufferedWriter by using the following lines of code:

```
FileWriter fw = new FileWriter("myfile.txt");
BufferedWriter bw = new BufferedWriter(fw);
```

Thus, Java `BufferedWriter` warps another `Writer` and adds a buffer that will write the text much faster and improve performance by buffering output.

BufferedWriter Constructors

2. BufferedWriter(Writer out, int size):
This constructor creates a buffered writer object that buffers character output stream out with specified buffer size.

Simple Steps for Buffering a Character-based File Stream

There are the following steps for buffering the character-based file stream:

1. Construct the underlying output stream. It will be an object of `FileWriter.`
2. Wrap the file stream in the appropriate bufferedwriter using Java `BufferedWriter` class.
3. Perform all I/O operations through the buffered writer.
4. At last, close the buffered output stream.

BufferedWriter Methods

In addition to methods inherited from the `Writer` class, `BufferedWriter` class also provides some useful methods that are as follows:

Note:

All the `BufferedWriter` class methods can throw an exception named `IOException` if any error occurs.

BufferedWriter Methods

Method	Description
void flush()	This method flushes the stream.
void newLine()	This method writes a newline character as defined by the line separator. The newLine() method is useful when we want a line separator in the output. There is no need to include a '\n' character in the text.
void write(char[ ] cbuf, int offset, int length)	This method writes a segment of an array of characters, starting at the position offset.
void write(int c)	This method is used to write a single character c.
void write(String s, int offset, int length)	This method writes a segment of a String, starting at the position offset.

Example 1: Write String to File

1. Let’s take a simple example program where we will write lines of text into the file using `BufferedWriter` class.

Example 1: Write String to File

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

public class BufferedWriterTester1 {
   public static void main(String[] args) throws IOException {
       // Create an object of FileWriter class.
       FileWriter fw = new FileWriter("./src/myfile.txt");

       // Create an object of BufferedWriter class and reference variable fw to its constructor.
       BufferedWriter bw = new BufferedWriter(fw);

       bw.write("This is an apple");
       bw.newLine(); // For line separator.
       bw.write("This is an orange");

       bw.close(); // Closing the stream.
       System.out.println("File written successfully.");
   }
}
```

Example 1: Write String to File

Output:

```
Successfully written...
```

myfile.txt:

```
This is an apple
This is an orange
```

Explanation:

In this program, we have created a `BufferedWriter` object wrapped around a `FileWriter` object, that specified the name of the 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

```
import java.io.BufferedWriter;
import java.io.FileWriter;
import java.io.IOException;
public class BufferedWriterTester2 {
   public static void main(String[] args) throws IOException {
       String[] strs = {
               "New Delhi is the capital of India.",
               "Washington, D.C. is the capital of US.",
               "Canberra is the capital of Australia."
       };

       // Create an object of FileWriter class.
       FileWriter fw = new FileWriter("./src/myfile.txt");

       // Create an object of BufferedWriter class by passing fw to its constructor.
       BufferedWriter bw = new BufferedWriter(fw);

       for(int i = 0; i < strs.length; i++) {
           bw.write(strs[i]);
           bw.newLine();
       }
       bw.close(); // Closing the stream.
       System.out.println("File written successfully.");
   }
}
```

Example 2: Write String[ ] Array to File

Output:

```
Successfully written...
```

myfile.txt:

```
New Delhi is the capital of India.
Washington D.C. is the capital of US.
Canberra is the capital of Australia.
```

Explanation:

In this program, we have created a `BufferedWriter` object wrapped around a `FileWriter` object, that specified the name of the 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.

END