CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

BufferedReader

Overview

1. Introduction
2. BufferedReader class declaration
3. BufferedReader constructors
4. Create BufferedReader object
5. Simple steps for buffering console-based user input
6. BufferedReader methods
7. BufferedReader Examples

Introduction

- **BufferedReader in Java** is a buffering input character stream that reads text from the buffer rather than directly underlying input stream or other text sources.

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

Introduction

Java `BufferedReader` is good for two important things:

1. It wraps another `Reader` and adds a buffer that will read the text much faster and improves performance by buffering input.

   For example, `InputStreamReader` can be wrapped by `BufferedReader` to buffer input data taken from the underlying file.

2. It provides a `readLine()` method to read a string of characters, arrays, and text lines.

Introduction

Note:

a. A buffered stream stores a large number of characters from the stream so that more than one character at a time can be read.

b. When a buffered stream is empty, it is filled again with as much text as possible, even if not all of it is immediately required. Thus, it will make future reads much faster and improve performance.

BufferedReader class declaration

`BufferedReader` is a subclass of the `Reader` class that extends `Object` class. It implements `Closeable`, `AutoCloseable`, and `Readable` interfaces. It is also a superclass of `LineNumberReader` class.

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

```
public class BufferedReader
       extends Reader
       implements Closeable, AutoCloseable, Readable
```

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

BufferedReader constructors

BufferedReader class provides two constructors for creating bufferedreader objects:

---

1. BufferedReader(Reader inputStream):
This constructor creates a bufferedreader object that buffers the input stream specified by `inputStream`.

It uses a default-sized input buffer. The default buffer size of BufferedReader is `8192` characters which is sufficient for most purposes.

---

2. BufferedReader(Reader inputStream, int bufSize):
This constructor creates a bufferedreader object that uses an input buffer of the specified size which must be greater than zero.

Create a BufferedReader object

To create a buffered input stream reader, create `BufferedReader` object.

The general syntax to create `BufferedReader` object in java program is as follows:

```
BufferedReader br = new BufferedReader(Reader inputStream);
```

For example:

```
InputStreamReader isr = new InputStreamReader(System.in);
BufferedReader br = new BufferedReader(isr);
```

After both statements execute, `br` is a character-based stream that is connected to the console through `System.in.`

Simple steps for buffering Console-based user input

Java `BufferedReader` can be used to buffer (store) the data input received from an `InputStreamReader` object. There are the following simple steps for buffering console-based user input.

---

1. Create an `InputStreamReader` object using a standard input stream.

```
InputStreamReader isr = new InputStreamReader(System.in); // Line 1
```

Simple steps for buffering Console-based user input

Java `BufferedReader` can be used to buffer (store) the data input received from an `InputStreamReader` object. There are the following simple steps for buffering console-based user input.

---

2. Create `BufferedReader` object and pass the reference variable `isr` to its constructor.

```
// Wrapping inputstreamreader into bufferedreader. // Line 2.
BufferedReader br = new BufferedReader(isr);
```

Simple steps for buffering Console-based user input

Java `BufferedReader` can be used to buffer (store) the data input received from an `InputStreamReader` object. There are the following simple steps for buffering console-based user input.

---

3. Perform all input operations through the buffered reader.

Simple steps for buffering Console-based user input

Java `BufferedReader` can be used to buffer (store) the data input received from an `InputStreamReader` object. There are the following simple steps for buffering console-based user input.

---

4. At last, close the buffered stream. Closing the buffered stream automatically causes the underlying file stream to be closed.

Simple steps for buffering Console-based user input

Java `BufferedReader` can be used to buffer (store) the data input received from an `InputStreamReader` object. There are the following simple steps for buffering console-based user input.

Note:
We can also combine step 1 and step 2 to make a single `BufferedReader` object for keyword input.

```
BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
```

Similarly, we can also wrap `FileReader` into `BufferedReader` for buffering the character-based file stream. For example, suppose a file called `myfile.txt`. To create a `BufferedReader` linked to that file.

```
BufferedReader br = new BufferedReader(new FileReader("myfile.txt"));
```

BufferedReader Methods

In addition to methods inherited from the `Reader` class, `BufferedReader` class also provides some useful methods for buffering character-based streams. They are as follows:

Note:
All the methods except lines(), and markSupported() will throw an exception named IOException if any I/O error is encountered.

BufferedReader Methods

Method	Description
void close()	This method closes the buffered input stream and releases any system resources associated with it.
Stream<String> lines()	This method returns a Stream that contains the text of lines read from this BufferedReader.
void mark(int readAheadLimit)	This method marks the present position in the buffered stream.
boolean markSupported()	This method tests whether this stream supports the mark() operation or not.
int read()	This method is used to read a single character from the buffered stream and returns it as an integer value. It returns -1 if the end of stream is encountered.
int read(char[ ] c, int off, int len)	This method reads characters into a portion of an array.

BufferedReader Methods

Method	Description
String readLine()	The readLine() method reads an entire line of text from what the user has typed. It is generally used to read a string from the keyboard. It returns a string that contains the characters read, and returns null if an attempt is made to read at the end of the stream.
boolean ready()	This method is used to verify whether this stream is ready to be read.
void reset()	The reset() method is used to reset the stream to the most recent mark.
long skip(long numChars)	The skip() method is used to skip over numChars characters of input. It returns the number of characters actually skipped.

Example 1: Read data from File

1. Let’s create a program where we will read a text of line from an existing file and display it on the console.

Example 1: Read data from File

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
public class BufferedReaderTester1 {
   public static void main(String[] args) throws IOException {
       String filepath = "./src/myfile.txt";

       // Create FileReader using filepath in its constructor.
       FileReader fr = new FileReader(filepath);

       // Create BufferedReader and pass FileReader fr to its constructor.
       BufferedReader br = new BufferedReader(fr);

       String lineOfText;
       // Read a line of text.
       while((lineOfText = br.readLine()) != null) {
           System.out.println(lineOfText);
       }
   }
}
```

Example 1: Read data from File

Output:

```
Welcome to Java Programming.
```

myfile.txt:

```
Welcome to Java Programming.
```

Example 2: Read characters from keyboard

2. Let’s create a Java program where we will read string data from the keyboard and display them on the console.

Example 2: Read characters from keyboard

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class BufferedReaderTester2 {
   public static void main(String [] args) throws IOException{
       // Create an object of BufferedReader using System.in.
       BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

       String str;
       System.out.println("Enter lines of text:");
       System.out.println("Enter 'end' to quit.");

       do{
           // After reading lines of text, it gets stored in variable str.
           str = br.readLine();
           System.out.println(str);
       } while(!str.equals("end"));
   }
}
```

Example 2: Read characters from keyboard

Output:

```
Enter lines of text:
Enter 'end' to quit.
Hello World
Hello World
end
end
```

In this program, an object of `BufferedReader` class takes an input as an argument that is an object of `InputStreamReader` with `System.in` as a parameter. It indicates that the default input device is keyboard.

Example 3: Character to ASCII

3. Let’s take an example program where we will take an input a character and display its ASCII value.

Example 3: Character to ASCII

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
public class BufferedReaderTester3 {
   public static void main(String [] args) throws IOException {

       BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
       char ch;
       System.out.println("Enter a character:");

       int data = br.read();
       ch = (char)data;
       System.out.println("ASCII value is " +(int)ch);
   }
}
```

Example 3: Character to ASCII

Output:

```
Enter a character:
Q
ASCII value is 81
```

END