CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

Reader

Overview

1. Introduction
2. Reader class declaration
3. Reader subclasses
4. Reader fields
5. Reader constructors
6. Reader methods
7. Reader Examples
8. Difference between Reader & InputStream

Introduction

- **Reader in Java** is an abstract class that reads characters from the files. In simple words, the reader class reads character streams.

- It is the superclass of all reader subclasses. The concrete class that extends `Reader` class operates on Unicode character streams.

Reader class declaration

`Reader` stream class extends `Object` class and implements `Closeable`, `AutoCloseable`, and `Readable` interfaces.

The general syntax to declare `Reader` stream class in java is as follows:

```
public abstract class Reader
       extends Object
       implements Readable, Closeable
```

Reader subclasses

The **most important** concrete subclass of the `Reader` class is `InputStreamReader`.

It contains an underlying input stream from which it reads input in bytes and then converts these bytes into Unicode characters according to the specified encoding.

Reader subclasses

In addition to `InputStreamReader` class, `java.io.` package also contains several reader classes that read characters without directly requiring an underlying input stream. They are as follows:

- FileReader
- BufferedReader
- CharArrayReader
- StringReader
- PipedReader
- FilterReader
- PushbackReader
- URLReader
- LineNumberReader

These concrete subclasses of `Reader` class have very similar functionality as `InputStream` subclasses, except input streams use bytes as their fundamental unit of information, while reader streams use characters. That is, where an inputstream reads a byte, reader stream reads a character.

Reader Fields

`Reader` class in Java defines the following field that is protected access modifier.

Field	Description
`protected Object lock`	It is used to synchronize operations on this stream.

Reader Constructors

`Reader` class in Java defines two protected constructors. It does not define any public constructor.

---

1. protected Reader():
This constructor constructs a new character-stream reader whose critical sections will synchronize on the reader itself.

---

2. protected Reader(Object lock):
This constructor constructs a new character-stream reader whose critical sections will synchronize on the given object.

Reader Methods

`Reader` class defines the following methods in java, all of which are public. All these methods are identical to methods available in the `InputStream` class. The most important reader class methods are as follows:

Reader Methods

Method	Description
abstract void close()	This method closes the stream and releases any system resources associated with it.
void mark(int readAheadLimit)	This method marks the present position in the stream.
boolean markSupported()	This method tests whether this stream supports the mark() operation. It returns true if the reader supports mark and reset operations.
static Reader nullReader()	This method returns a new Reader that reads no characters.
int read()	This method reads a single character.
int read(char[ ] characterArray)	This method reads an array of characters with data. It returns an int value that represents the number of characters that were read. If the reader stream is reached at the end, the value of -1 is returned and the array is not modified.

Reader Methods

Method	Description
abstract int read(char[ ] c, int n, int m)	This method reads a specified number of characters from the underlying input stream into an array starting from nth character. It returns an int value that represents the actual number of characters that were read. -1 returns if the reader stream reached the end. In simple words, this method reads characters into a part of an array.
int read(CharBuffer target)	This read() method was added in Java 5 version. It directly reads characters into the specified character buffer starting at buffer's current position. It returns the number of characters read, or -1 on the end of stream.
boolean ready()	The read() method returns true if the reader stream is ready to be read, or false if it isn't. It is not similar to InputStream's available() method.

Reader Methods

Method	Description
void reset()	The reset() method resets the reader's stream by moving back to an earlier position.
long skip(long numChars)	The skip() method reads and skips the specified number of characters of input. It returns the number of characters actually skipped, or -1 if the end of stream is reached.
long transferTo(Writer out)	The transferTo() method reads all characters from this reader and writes the characters to the specified writer in the same order that they were read. All the methods defined in Java Reader class (except for markSupported( )) will throw an IOException on error conditions.

Example 1: Read data from File

1. Let’s take an example program where we will read data from a file and display it on the console. Look at the following source code.

Example 1: Read data from File

```
import java.io.FileReader;
import java.io.Reader;
import java.io.IOException;
public class ReaderTester1 {
   public static void main(String[] args) throws IOException{
           // Create an object of Reader class and pass path of filename.
           Reader reader = new FileReader("./src/myfile.txt");

           int data = reader.read();
           while (data != -1) {
               System.out.print((char) data);
               data = reader.read();
           }
           System.out.println("\n");
           System.out.println("Does myfile.txt support mark operation: " +reader.markSupported());

           reader.close(); // Closing reader stream.
   }
}
```

Example 1: Read data from File

Output:

```
Welcome to Java Programming.

Does myfile.txt support mark operation: false
```

myfile.txt:

```
Welcome to Java Programming.
```

Difference between Reader and InputStream

These are the following differences between `Reader` and `InputStream` in Java:

1. `Reader` reads sequences of characters, whereas `InputStream` reads sequences of bytes.

2. `Reader` class defines `read(char[])` and `read(char[], int, int)` methods instead of `read(byte[])` and `read(byte[], int , int)` methods.

3. `Reader` class does not define an `available()` method. It defines a boolean `ready()` method.

4. `Reader` defines an int `read(CharBuffer target)` method for reading characters from a character buffer.

END