CSCI 2120:
Software Design & Development II

UNIT3: I/O management

io api

InputStreamReader

Overview

1. Introduction
2. InputStreamReader class declaration
3. InputStreamReader constructors
4. InputStreamReader methods
5. InputStreamReader Examples

Introduction

- An **InputStreamReader in Java** is a character input stream that uses the stream of bytes as its data source.
- It acts as a bridge between an incoming stream of bytes and an outgoing sequence of characters and converts a byte stream into a character stream.
- Java `InputStreamReader` reads bytes from a specified `InputStream` and converts (translates) into Unicode characters according to the default or specified character encoding.
- In other words, data read from the source input stream are decoded from bytes using the specified charset.

InputStreamReader class declaration

An `InputStreamReader` is a concrete subclass of `Reader` class that extends `Object` class. It is also a superclass of `FileReader` class. It implements `Closeable`, `AutoCloseable`, and `Readable` interfaces.

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

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

InputStreamReader Constructors

1. InputStreamReader(InputStream in):
This constructor creates an `InputStreamReader` object that uses the default character encoding to convert bytes into characters.

The general syntax to create an object of `InputStream` class is given below:

```
InputStreamReader inStream = new InputStreamReader(InputStream in);
```

Here, the parameter to the constructor of `InputStreamReader` is of type `InputStream`, so we can pass an object of any class derived from `InputStream` to it.

For example:

```
InputStreamReader inStream = new InputStreamReader(System.in);
```

InputStreamReader Constructors

2. InputStreamReader(InputStream in, String charsetName):
This constructor creates an `InputStreamReader` object that uses the named character encoding.

`charsetName` specifies the character encoding that is used to convert bytes into characters. This constructor throws an exception named `UnsupportedEncodingException` when named character encoding is not supported.

---

3. InputStreamReader(InputStream in, Charset cs):
This character creates an `InputStreamReader` object that uses the specified `charset` to decode bytes into characters.

---

4. InputStreamReader(InputStream in, CharsetDecoder dec):
This constructor creates an `InputStreamReader` object that uses the specified `charset` decoder.

InputStreamReader Methods

In addition to methods inherited from the `Reader` class, `InputStreamReader` class in Java also defines some useful methods. They are as follows:

InputStreamReader Methods

Method	Description
String getEncoding()	This method returns the name of the character encoding being used by this stream.
int read()	This method reads a single character.
int read(char[ ] c, int n, int m)	This method reads characters into a portion of an array.
boolean ready()	This method checks whether this stream is ready to be read. It returns true if the stream is ready to be read.

Example 1: Read data from File

1. Let’s take a simple example program where we will read data from a file and display it on the console using the input stream reader class.

Example 1: Read data from File

```
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.io.IOException;
public class InputStreamReaderTester1 {
   public static void main(String[] args) throws IOException {
       // Create an object of FileInputStream class and pass path of filename.
       FileInputStream fis = new FileInputStream("./src/myfile.txt");

       // Create InputStreamReader and pass fis to its constructor.
       InputStreamReader inStream = new InputStreamReader(fis);

       int data = inStream.read(); // Calling to read() method.
       while (data != -1) {
           System.out.print((char) data);
           data = inStream.read();
       }
       inStream.close();
   }
}
```

Example 1: Read data from File

Output:

```
Welcome to Java Programming.
```

myfile.txt:

```
Welcome to Java Programming.
```

Example 2: ready() and getEncoding() method

2. Let’s take a simple example program where we will implement `ready()` and `getEncoding()` method

Example 2: ready() and getEncoding() method

```
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.io.IOException;
public class InputStreamReaderTester2 {
   public static void main(String[] args) throws IOException {
       // Create FileInputStream and pass path of filename.
       FileInputStream fis = new FileInputStream("./src/myfile.txt");

       // Create InputStreamReader and pass fis to its constructor.
       InputStreamReader inStream = new InputStreamReader(fis);

       // Calling getEncoding() method to get the character encoding present in the stream.
       String encoding = inStream.getEncoding();
       System.out.println("Name of encoding used : " + encoding);

       System.out.println("Ready? : " + inStream.ready());
       while (inStream.ready()) {
           int byteVal = inStream.read();
           char ch = (char) byteVal;
           System.out.print(ch);
       }
       System.out.println("\nReady? : " + inStream.ready());

   }
}
```

Example 2: ready() and getEncoding() method

Output:

```
Name of encoding used : UTF8
Ready? : true
Welcome to Java Programming.
Ready? : false
```

myfile.txt:

```
Welcome to Java Programming.
```

Example 3: Read characters from keyboard

3. Let’s create a program where we will take a character as input from the keyboard and display it on the console. Look at the following source code step by step.

Example 3: Read characters from keyboard

```
import java.io.IOException;
import java.io.InputStreamReader;
public class InputStreamReaderTester3 {
   public static void main(String[] args) throws IOException {

       // Create an InputStreamReader object using standard input stream.
       InputStreamReader isr = new InputStreamReader(System.in);

       System.out.println("Enter a character:");
       char ch = (char) isr.read();
       System.out.println("Input Character: " +ch);
   }
}
```

Example 3: Read characters from keyboard

Output:

```
Enter a character:
A
Input Character: A
```

Example 4: Recursive read method

4. Let’s take a simple example program where we will refactor Example 2 using a recursive method instead of a while-loop to read all the characters from the reader

Example 4: Recursive read method

```
import java.io.FileInputStream;
import java.io.InputStreamReader;
import java.io.IOException;
public class InputStreamReaderTester4 {
   public static void main(String[] args) throws IOException {
       FileInputStream fis = new FileInputStream("./src/myfile.txt");          // Create FileInputStream using filepath
       InputStreamReader inStream = new InputStreamReader(fis);                // Create InputStreamReader using fis.

       String encoding = inStream.getEncoding();
       System.out.println("Name of encoding used : " + encoding);

       System.out.println("Ready? : " + inStream.ready());
       recursiveRead(inStream);
       System.out.println("\nReady? : " + inStream.ready());

   }

   static boolean recursiveRead(InputStreamReader inStream) throws IOException{
       if (inStream.ready() == false){                                          //base case
           return false;
       }
       int byteVal = inStream.read();                                           //recursive case
       char ch = (char) byteVal;
       System.out.print(ch);
       return recursiveRead(inStream);
   }
}
```

Example 4: Recursive read method

Output:

```
Name of encoding used : UTF8
Ready? : true
Welcome to Java Programming.
Ready? : false
```

myfile.txt:

```
Welcome to Java Programming.
```

END