Difference between FileInputStream and FileReader in Java | InputStream vs Reader

Before going to explain specific difference between `FileInputStream` and `FileReader` in Java, I would like to state fundamental difference between an `InputStream` and a `Reader` in Java, and when to use `InputStream` and when to go for `Reader`. Actually, Both `InputStream` and `Reader` are abstractions to read data from source, which can be either file or socket, but main difference between them is, InputStream is used to read binary data, while `Reader` is used to read text data, precisely Unicode characters. So *what is difference between binary and text data?* well everything you read is essentially bytes, but to convert a byte to text, you need a character encoding scheme. `Reader` classes uses character encoding to decode bytes and return characters to caller.

`Reader` can either use default character encoding of platform on which your Java program is running or accept a `Charset` object or name of character encoding in String format e.g. `"UTF-8"`. Despite being one of the simplest concept, lots of Java developers make mistakes of not specifying character encoding, while reading text files or text data from socket.

Remember, if you don't specify correct encoding, or your program is not using character encoding already present in protocol e.g. encoding specified in `"Content-Type"` for HTML files and encoding presents in header of XML files, you may not read all data correctly. Some characters which are not present in default encoding, may come up as ? or little square.

Once you know this fundamental *difference between stream and reader*, understanding difference between `FileInputStream` and `FileReader` is quite easy. Both allows you to read data from File, but `FileInputStream` is used to read binary data, while `FileReader` is used to read character data.

FileReader vs FileInputStream Java

Since `FileReader` extends `InputStreamReader`, it uses character encoding provided to this class, or else default character encoding of platform. Remember, `InputStreamReader` caches the character encoding and setting character encoding after creating object will not have any affect. Let's see an example of *How to use FileInputStream and FileReader in Java*. You can provide either a File object or a String, containing location of file to start reading character data from File. This is similar to FileInputStream, which also provides similar constructors for reading from file source. Though its advised to use BufferedReader to read

Interview Questions
- core java interview question (161)
- data structure and algorithm (45)
- Coding Interview Question (32)
- SQL Interview Questions (24)
- thread interview questions (20)
- database interview questions (18)
- servlet interview questions (17)
- collections interview questions (15)
- spring interview questions (9)
- Programming interview question (4)
- hibernate interview questions (4)

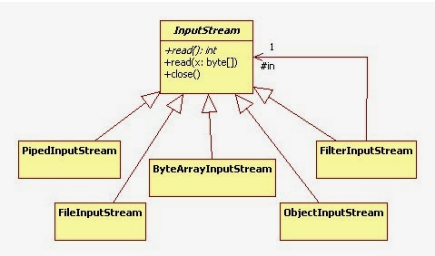
Translate this blog
Select Language
Powered by Google Translate

data from file.

```
import java.awt.Color;
import java.io.FileInputStream;
import java.io.FileReader;
import java.io.IOException;

/**
 * Java Program to read data from file as stream of bytes and stream of characters.
 * It also highlight key difference between FileInputStream and FileReader that
 * FileReader is meant for reading streams of characters.
 * For reading streams of raw bytes, consider using a FileInputStream.
 *
 * @author Javin Paul
 */
public class HowToReadFileInJava {
    public static void main(String args[]) {

        // Example 1 - Reading File's content using FileInputStream
        try (FileInputStream fis = new FileInputStream("data.txt")) {
            int data = fis.read();
            while (data != -1) {
                System.out.print(Integer.toHexString(data));
                data = fis.read();
            }
        } catch (IOException e) {
            System.out.println("Failed to read binary data from File");
            e.printStackTrace();
        }


        // Example 2 - Reading File data using FileReader in Java
        try (FileReader reader = new FileReader("data.txt")) {
            int character = reader.read();
            while (character != -1) {
                System.out.print((char) character);
                character = reader.read();
            }
        } catch (IOException io) {
            System.out.println("Failed to read character data from File");
            io.printStackTrace();
        }
    }
}
```

Output:

```
4157532d416d617a6f6e20576562205365727696365da474f4f472d476f6f676c65da4150504c2d417070
AWS-Amazon Web Service
GOOG-Google
APPL-Apple
GS-Goldman Sachs
```



Our first example is reading data from file byte by byte, so its bound to be very slow. `read()` method from `FileInputStream` is a blocking method, which reads a byte of data or blocks if no input is yet available. It either returns next byte of data, or `-1` if the end of the file is reached. This means we read one byte in each iteration of loop and prints it as Hexadecimal String. By the way, there is options to convert InputStream into byte array as well. On the other hand, in example 2 are reading data

Java Tutorials

- date and time tutorial (18)
- FIX protocol tutorial (16)
- java collection tutorial (53)
- java IO tutorial (25)
- Java JSON tutorial (6)
- Java multithreading Tutorials (33)
- Java Programming Tutorials (27)
- Java xml tutorial (9)

Search This Blog

Search

character by character. `read()` method from `InputStreamReader`, which is inherited by `FileReader` reads a single character and returns the character read, or -1 if the end of the stream has been reached. This is why you see exactly same text as written in file output from our example 2.

That's all on **difference between FileInputStream and FileReader in Java**. Bottom line is use `FileReader` or `BufferedReader` to read stream of characters or text data from File and always specify character encoding. Use FileInputStream to read raw streams of bytes from file or socket in Java.

You might like:

- ARM- Automatic resource management in Java7 an example tutorial