CS209A - LAB1

Key Content

- File
- I/O Stream
- Charsets
- Some pitfalls

1. Class `File` basic usage

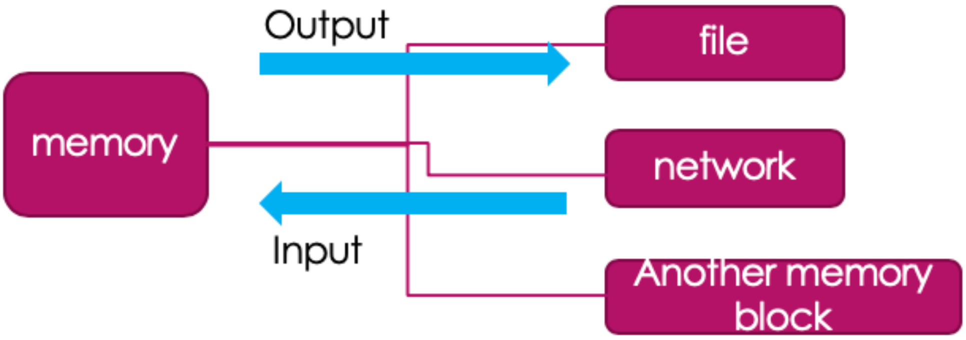
```java
public static boolean createFile(String destFileName) {
    File file = new File(destFileName);
    if (file.exists()) {
        System.out.println("Create single file " + destFileName + " fail,target file already exists! ");
        return false;
    }
    if (destFileName.endsWith(File.separator)) {
        System.out.println("Create single file " + destFileName + " fail, target file cannot be a directory! ");
        return false;
    }
    // Check if the directory where the target file is located exists
    if (!file.getParentFile().exists()) {
        // if the directory where the target file is located doesn't exist, create
        // its' parent directory.
        System.out.println("directory where target file is located doesn't exist, create its' parent directory! ");
        File parentFile = file.getParentFile();
        parentFile.mkdirs();
        if (!file.getParentFile().mkdirs()) {
            System.out.println("Create directory where target file is located fails! ");
            return false;
        }
    }
    // Create target file
    try {
        if (file.createNewFile()) {
            System.out.println("Create single file " + destFileName + " success! ");
            return true;
        } else {
            System.out.println("Create single file " + destFileName + " fail! ");
            return false;
        }
    } catch (IOException e) {
        e.printStackTrace();
        System.out.println("Create single file " + destFileName + " fail! " + e.getMessage());
        return false;
    }
```

Invoke above method and observe result.

2. JAVA I/O

2.1 I/O streams

A computer can be connected to many different types of input and output devices. If a programming language had to deal with each type of device as a special case, the complexity would be overwhelming. One of the major achievements in the history of programming has been to come up with good abstractions for representing I/O devices. In Java, the main I/O abstractions are called I/O streams.



Files are common sources and destination for an IO stream.

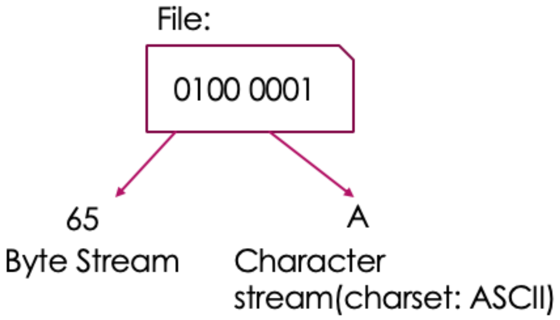
2.2 Byte and Character Streams

Byte stream:

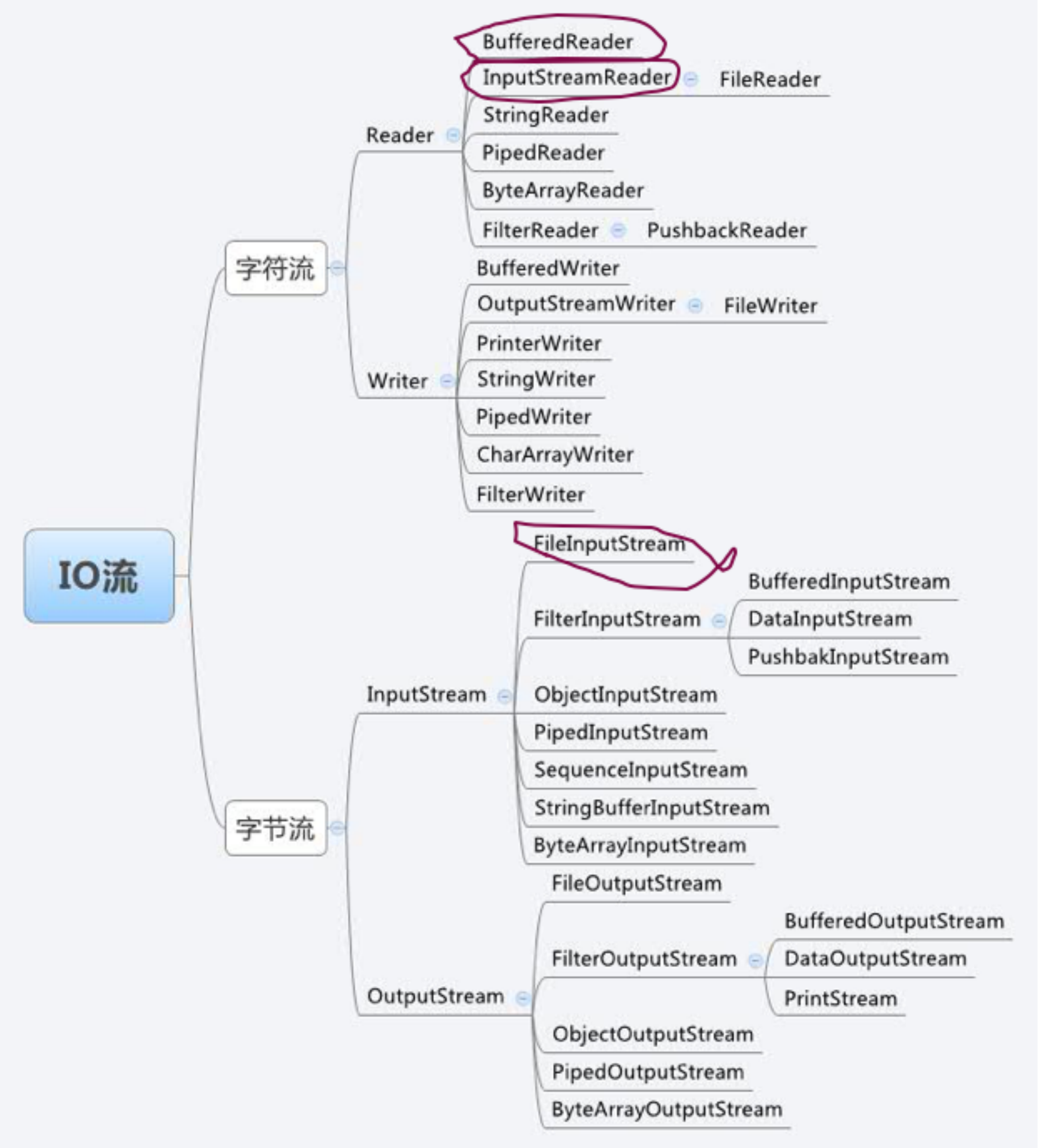
A byte stream is for machine-formatted data, is represented in binary form, the same way that data is represented inside the computer, that is, as strings of zeros and ones.

Character stream:

A character stream is for human-readable data – for instance text in English or Chinese. To work, the essence is to look up the specified charset (such as utf-8,utf-16) when reading based on the byte stream. Because when working with text data, the same code can represent characters with different ways (please see the prac intro video).



2.3 JAVA IO Stream Class Structure



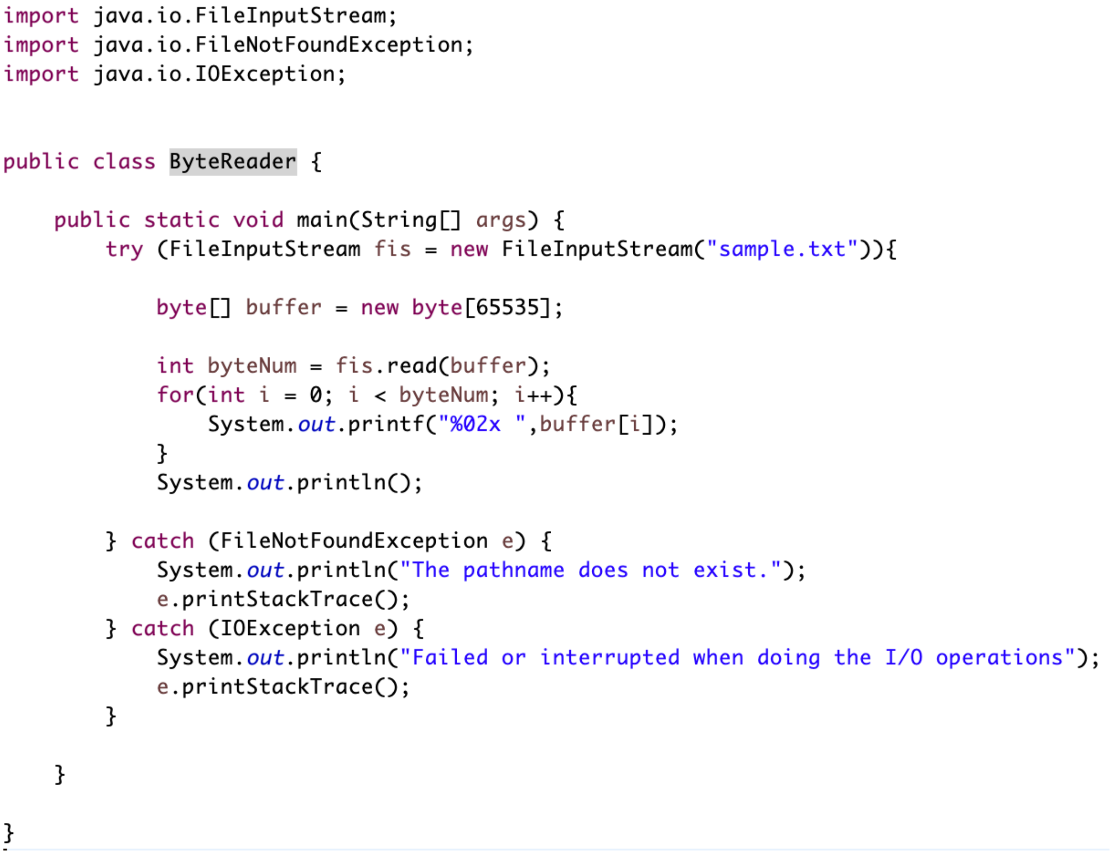
2.4 Sample Code

2.4.1 FileInputStream

FileInputStream obtains input bytes from a file in a file system.

Parent class： InputStream

Other related classes： ByteArrayInputStream, StringBufferInputStream, and FileInputStream are three basic media streams that read data from Byte arrays, stringbuffers, and local files, respectively. The PipedInputStream reads data from a pipe, often a pipe can be used to provide shared memory among several threads.



```
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;


public class ByteReader {

    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream("sample.txt")){

            byte[] buffer = new byte[65535];

            int byteNum = fis.read(buffer);
            for(int i = 0; i < byteNum; i++){
                System.out.printf("%02x ",buffer[i]);
            }
            System.out.println();

        } catch (FileNotFoundException e) {
            System.out.println("The pathname does not exist.");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Failed or interrupted when doing the I/O operations");
            e.printStackTrace();
        }

    }

}
```

Observe the result.

2.4.2 InputStreamReader

InputStreamReader is a bridge between a byte stream and a character stream that converts a byte stream into a character stream.

```java
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;

public class StreamReader {

    public static void main(String[] args) {

        try (InputStreamReader isr = new InputStreamReader(new FileInputStream("sample.txt"), "gb18030")) {

            char[] cbuf = new char[65535];
            int file_len = isr.read(cbuf);

            System.out.println(file_len);
            System.out.println(cbuf);

        } catch (FileNotFoundException e) {
            System.out.println("The pathname does not exist.");
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            System.out.println("The Character Encoding is not supported.");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Failed or interrupted when doing the I/O operations");
            e.printStackTrace();
        }
    }
}
```

Observe the result.

2.4.3 BufferedReader

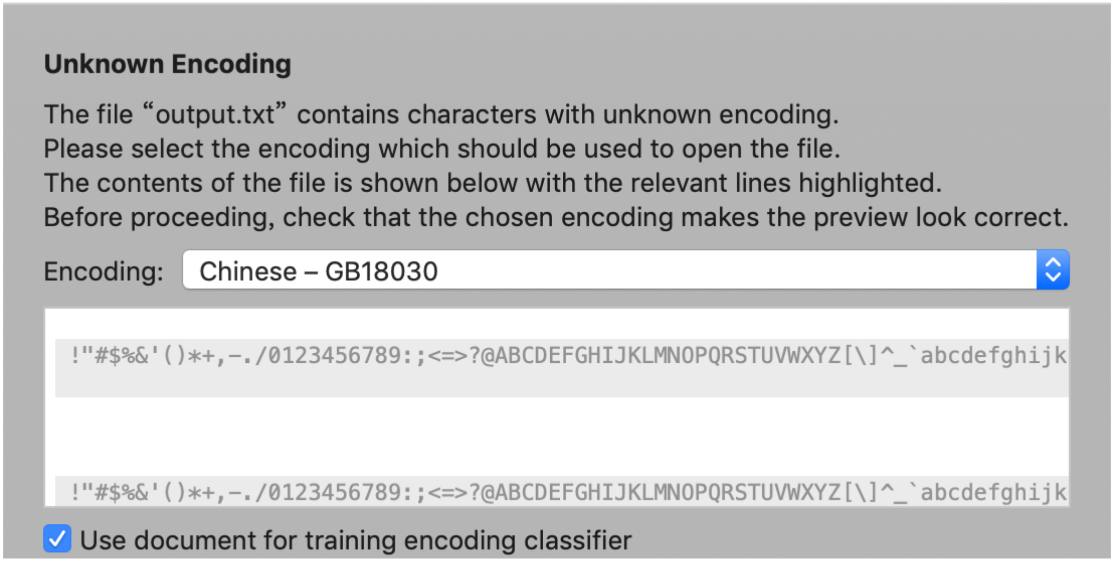
```
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;

public class BufferReader {

    public static void main(String[] args) {
        try (FileInputStream fis = new FileInputStream(new File("sample.txt"));
                InputStreamReader isr = new InputStreamReader(fis, "gb18030");
                BufferedReader bReader = new BufferedReader(isr);){


            char[] cbuf = new char[65535];
            int file_len = bReader.read(cbuf);

            System.out.println(file_len);
            System.out.println(cbuf);

        } catch (FileNotFoundException e) {
            System.out.println("The pathname does not exist.");
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            System.out.println("The Character Encoding is not supported.");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Failed or interrupted when doing the I/O operations");
            e.printStackTrace();
        }
    }
}
```

Observe the result.

2.4.4 FileOutputStream

```java
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;

public class ByteWriter {

    public static void main(String[] args) {
        try (FileOutputStream fos = new FileOutputStream("output.txt")){

            byte[] buffer = new byte[65535];
            for(int i = 0; i < buffer.length; i++){
                buffer[i] = (byte) i;
            }

            fos.write(buffer);
            fos.flush();//fos.close();
        } catch (FileNotFoundException e) {
            System.out.println("The pathname does not exist.");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Failed or interrupted when doing the I/O operations");
            e.printStackTrace();
        }

    }

}
```

When you try to open the output.txt, it is possible that you will encounter a problem like this:



Unknown Encoding

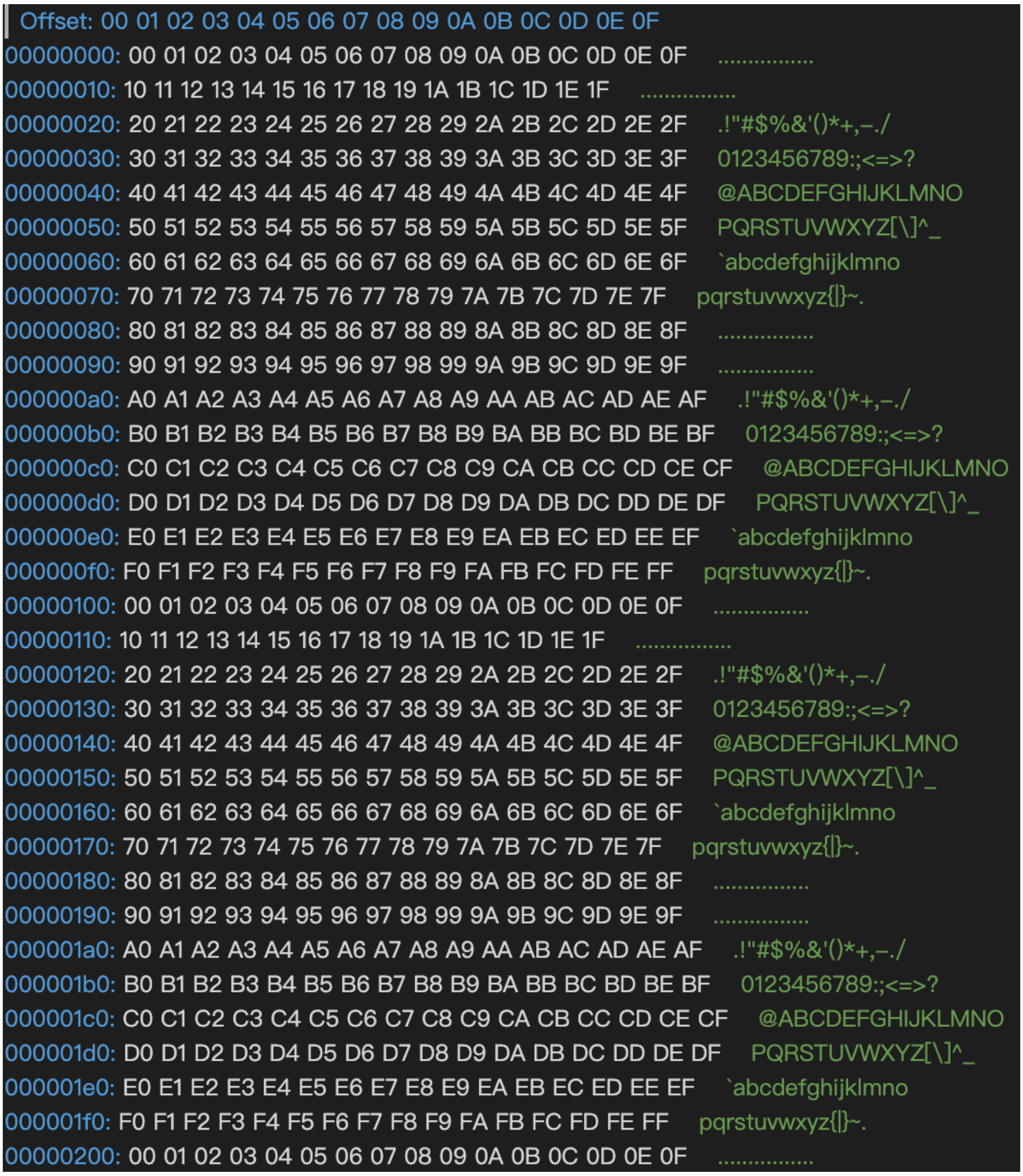
The file “output.txt” contains characters with unknown encoding.
Please select the encoding which should be used to open the file.
The contents of the file is shown below with the relevant lines highlighted.
Before proceeding, check that the chosen encoding makes the preview look correct.

Encoding: Chinese – GB18030

```
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk

 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijk
```

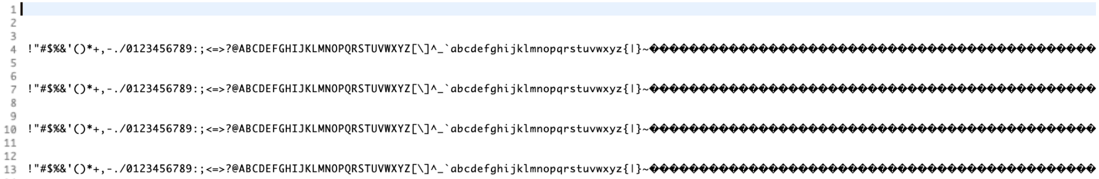
☑ Use document for training encoding classifier

TO solve this problem should open a binary document with Notepad++(install HexEditor) / VS Code(install extension:Hexdump)/Sublime Text(install plugin: HexViewer) /UltraEdit and so on.



```
 Offset: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   ................
00000010: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F   ................
00000020: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   .!"#$%&'()*+,-./
00000030: 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F   0123456789:;<=>?
00000040: 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F   @ABCDEFGHIJKLMNO
00000050: 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F   PQRSTUVWXYZ[\]^_
00000060: 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F   `abcdefghijklmno
00000070: 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F   pqrstuvwxyz{|}~.
00000080: 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F   ................
00000090: 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F   ................
000000a0: A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF   .!"#$%&'()*+,-./
000000b0: B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF   0123456789:;<=>?
000000c0: C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF   @ABCDEFGHIJKLMNO
000000d0: D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF   PQRSTUVWXYZ[\]^_
000000e0: E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF   `abcdefghijklmno
000000f0: F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF   pqrstuvwxyz{|}~.
00000100: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   ................
00000110: 10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F   ................
00000120: 20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F   .!"#$%&'()*+,-./
00000130: 30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F   0123456789:;<=>?
00000140: 40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F   @ABCDEFGHIJKLMNO
00000150: 50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F   PQRSTUVWXYZ[\]^_
00000160: 60 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F   `abcdefghijklmno
00000170: 70 71 72 73 74 75 76 77 78 79 7A 7B 7C 7D 7E 7F   pqrstuvwxyz{|}~.
00000180: 80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F   ................
00000190: 90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F   ................
000001a0: A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF   .!"#$%&'()*+,-./
000001b0: B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF   0123456789:;<=>?
000001c0: C0 C1 C2 C3 C4 C5 C6 C7 C8 C9 CA CB CC CD CE CF   @ABCDEFGHIJKLMNO
000001d0: D0 D1 D2 D3 D4 D5 D6 D7 D8 D9 DA DB DC DD DE DF   PQRSTUVWXYZ[\]^_
000001e0: E0 E1 E2 E3 E4 E5 E6 E7 E8 E9 EA EB EC ED EE EF   `abcdefghijklmno
000001f0: F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 FA FB FC FD FE FF   pqrstuvwxyz{|}~.
00000200: 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F   ................
```

If you open it with UTF-8:



```



 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~������������������������������������������������


 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~������������������������������������������������


 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~������������������������������������������������


 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~������������������������������������������������
```

If you open it with UTF-16 BE:

âS⅌□□ฏ∞ሐᐅ·ທ□□ḟ‡|⫽ℛ∵≒□ʿ〈㈳仟垅帉摰歐犺晾箹滕褉軛龎魺乏佩㔟呕噗墧婃扉幟愮扣搨晧桩横沩湯炈牳瑤罕磁穻粁繿庳芃菜蚾袉誋貍躏邑銓钕隗顋骛鲞麟

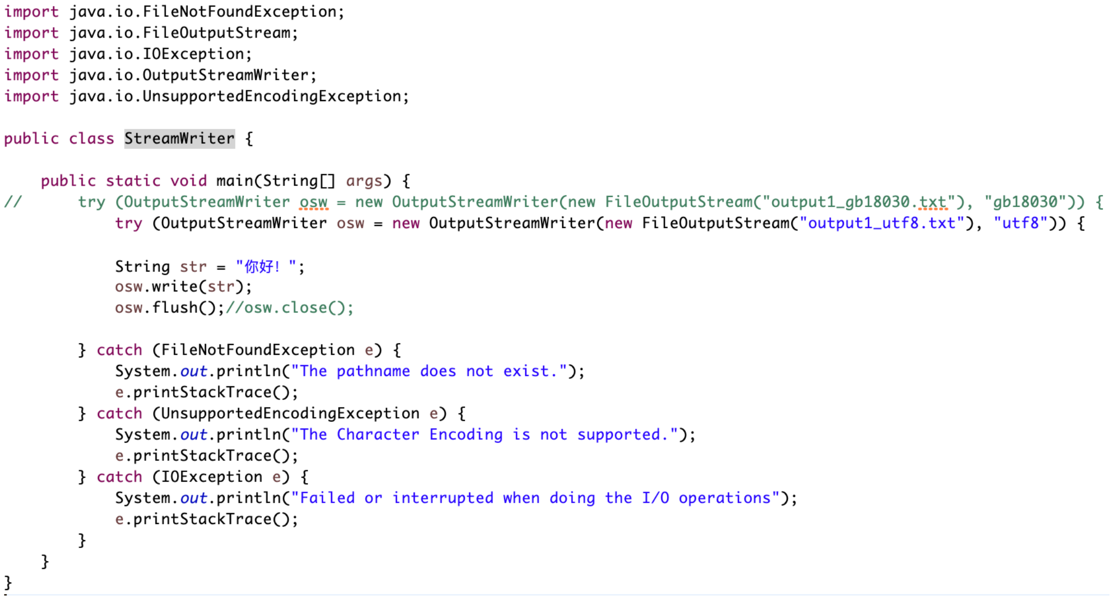
ᦀᄐU□□□걙꺖낱늴됩뚱뢹못벽뺍상쉿쉞웃좠좛첼캔탑퉁퓔훗\xD8\xD9\xDE\xDF 〉 □

شىâS⅌□□ฏ∞ሐᐅ·ທ□□ḟ‡|⫽ℛ∵≒□ʿ〈㈳仟垅帉摰歐犺晾箹滕褉軛龎魺乏佩㔟呕噗墧婃扉幟愮扣搨晧桩横沩湯炈牳瑤罕磁穻粁繿庳芃菜蚾袉誋貍躏邑銓钕隗顋骛鲞麟

ঌ⾣∪□□□겼꺪낱늽뒵뚱뢹못벽뺍상싃쒪웃즱쬻첼캡탑퉁퓖혿\xD8\xD9□xDE\xDF 〉 □

(Additional content is omitted)

2.4.5 OutputStreamWriter



```
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;

public class StreamWriter {

    public static void main(String[] args) {
//      try (OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("output1_gb18030.txt"), "gb18030")) {
            try (OutputStreamWriter osw = new OutputStreamWriter(new FileOutputStream("output1_utf8.txt"), "utf8")) {

            String str = "你好！";
            osw.write(str);
            osw.flush();//osw.close();

        } catch (FileNotFoundException e) {
            System.out.println("The pathname does not exist.");
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            System.out.println("The Character Encoding is not supported.");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Failed or interrupted when doing the I/O operations");
            e.printStackTrace();
        }
    }
}
```

(1) Run above program, write “你好！” to output1_utf8.txt, charset is “utf8”;

(2) Modify the program, write “你好！” to output1_gb18030.txt, charset is “gb18030”;

(3) Open the output1_utf8.txt and output1_gb18030.txt in your notepad;

(4) Open the output1_utf8.txt and output1_gb18030.txt with a Hex Editor.

2.4.6 BufferWriter

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.UnsupportedEncodingException;

public class BufferWriter {

    public static void main(String[] args) {
        try (FileOutputStream fos = new FileOutputStream(new File("output2_gb18030.txt"));
                OutputStreamWriter osw = new OutputStreamWriter(fos, "gb18030");
                BufferedWriter bWriter = new BufferedWriter(osw);){
            bWriter.write("你好！\n");
//          bWriter.write(100);
            bWriter.write("100");
            bWriter.write(" 分 \n");
            bWriter.write("送给你！\n");
            bWriter.flush();//bWriter.close();

        } catch (FileNotFoundException e) {
            System.out.println("The pathname does not exist.");
            e.printStackTrace();
        } catch (UnsupportedEncodingException e) {
            System.out.println("The Character Encoding is not supported.");
            e.printStackTrace();
        } catch (IOException e) {
            System.out.println("Failed or interrupted when doing the I/O operations");
            e.printStackTrace();
        }
    }
}
```

(1) Run above program, open output1_gb18030.txt;

(2) Modify “100” to 100, open output1_gb18030.txt and see what happened;

(3) Modify above program, try to produce massive data and write to a file;

(4) Using OutputStreamWriter to write massive data to a file, compare the run time.

3 Charsets and Character Encoding

There are various ways for characters to be encoded as binary data. A particular encoding is known as a charset or character set . The encoding for charsets are specified by international standards organizations and have names such as “UTF-16”, “UTF-8,” and “ISO-8859-1”.

In UTF-16, characters are encoded as 16-bit UNICODE values; this is the character set that is **used internally by Java**. UTF-8 is another way of encoding UNICODE characters using 8 bits for common ASCII characters and longer codes for other characters. Both UTF-16 and UTF-8 use variable length encodings, UTF-16 uses either 2 or 4 bytes (instead of 1, 2, 3, or 4 bytes in UTF-8).

ISO-8859-1, is a widely used standard for Roman letters (ie English type letters and European variations), also known as “Latin-1,” is an 8-bit encoding that includes ASCII characters as well as certain accented characters that are used in several European languages.

3.1 Char vs binary value

Run the following code:

```
char c = '赵';
int value = c;
System.out.printf("%s\n",c);
System.out.printf("%X\n",value);
```

`Observe the result.`

3.2 Transform from different charset

Run the following code:

```
    String str = "赵耀"; // UTF-16
    try
    {
        byte[] bytes1 = str.getBytes("GBK"); // or GBK
        for (byte b : bytes1) {
            System.out.printf("%2X ", b);
        }
        System.out.println();
        byte[] bytes2 = str.getBytes("UTF-16");
        for (byte b : bytes2) {
            System.out.printf("%02X ", b);
        }
        System.out.println();

        byte[] bytes3 = str.getBytes("UTF-16BE");
        for (byte b : bytes3) {
            System.out.printf("%02X ", b);
        }
        System.out.println();

        byte[] bytes4 = str.getBytes("UTF-16LE");
        for (byte b : bytes4) {
            System.out.printf("%02X ", b);
        }
        System.out.println();
    }catch(UnsupportedEncodingException e){

        e.printStackTrace();
    }
```

Observe the result.

PS: UTF-16:赵-8D75 耀-8000,   GB:赵-D5D4 耀-D2AB

4 Some pitfalls

4.1 Sample 1

In StreamReader(2.4.2 InputStreamReader), try to change the following line:

```
InputStreamReader isr = new InputStreamReader(fis, "gb18030");
```

To

```
InputStreamReader isr = new InputStreamReader(fis, "utf8");
```

Observe the result.

4.2 Sample 2

Try to run the following code:

```
public class SurrogatePairsTest {

    public static void main(String[] args) {
        String s=String.valueOf(Character.toChars(0x10437));
        System.out.println(s);
        System.out.println(s.charAt(0));

        char[]chars=s.toCharArray();
        for(char c:chars){
            System.out.format("%x",(short)c);
        }
    }
}
```

Observe the result and explain why the output of `s` is not the same as `s.charAt(0)` ?
Why `0x10437` could be converted to `0xd801dc37` ?

Answer:

UTF-16 is used internally by Java, and Java primitive type `char` is 16 bits wide. When a Unicode character is with code above 0xFFFF, is encoded in UTF-16 by pairs of 16-bit code units called **surrogate pairs**.

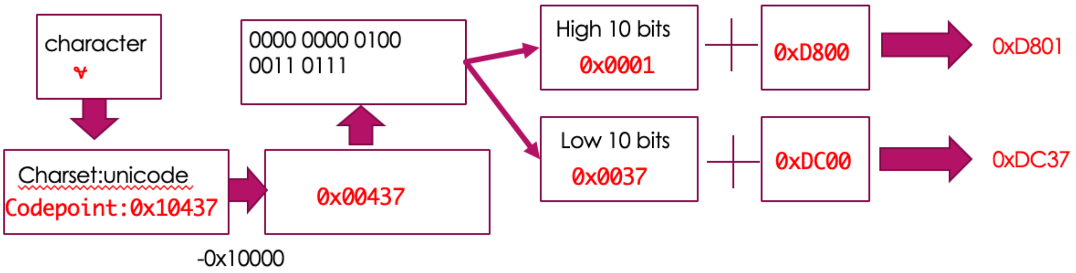
`0x10437 to 0xd801dc37`

Step1: 0x10437 minus 0x10000 gives 0x00437, binary 0000 0000 0100 0011 0111(0x00437)

Step2: Partition its upper and lower 10 bit values (binary) :0000000001 and 0000110111

Step3: Add 0xD800 to the upper value to form the higher part: 0xD800 + 0x0001 = 0xD801

Step4: Add 0xDC00 to the lower value to form a lower part: 0xDC00 + 0x0037 = 0xDC37.



Hint: Don't use notepad please use Notepad++ / VS Code/Sublime Text and other software that can handle multiple encodings easily!

Reference

https://zh.wikipedia.org/wiki/Unicode%E5%AD%97%E7%AC%A6%E5%B9%B3%E9%9D%A2%E6%98%A0%E5%B0%84

https://unicode-table.com/cn/blocks/cjk-unified-ideographs/

https://www.qqxiuzi.cn/bianma/zifuji.php

https://www.jianshu.com/p/ad4bff4d9fa3

https://docs.oracle.com/javase/8/docs/technotes/guides/intl/overview.html

http://blog.51cto.com/cnn237111/1080628