

CS209A - File IO

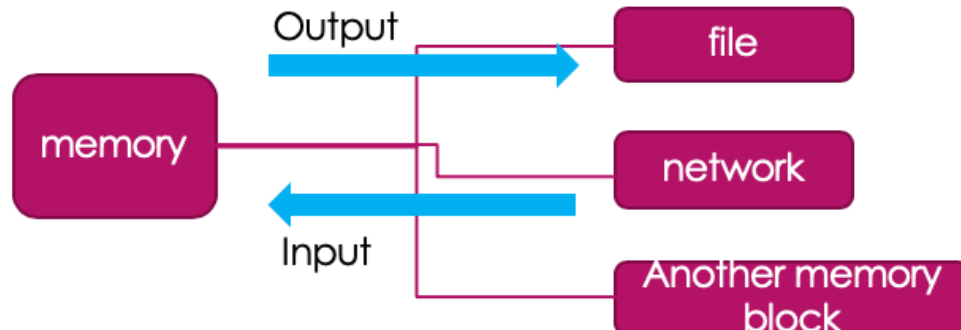
Key Content

- I/O Stream
- Charsets
- Some pitfalls

1. I/O streams

1.1 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.

1.2 Byte and Character Streams

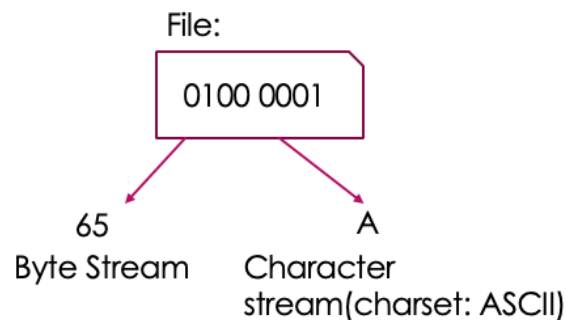
Byte streams:

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 streams:

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 in different ways.



1.3 JAVA IO Stream Class Structure

Character streams are often "wrappers" for byte streams. The character stream uses the byte stream to perform the physical I/O, while the character stream handles translation between characters and bytes. [FileReader](#), for example, uses [FileInputStream](#), while [FileWriter](#) uses [FileOutputStream](#).

There are two general-purpose byte-to-character "bridge" streams: [InputStreamReader](#) and [OutputStreamWriter](#). Use them to create character streams when there are no prepackaged character stream classes that meet your needs.

1.4 Sample Code

1.4.1 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.

1.4.2InputStreamReader

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

```
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.

1.4.3BufferedReader

If have no buffer, each read or write request is handled directly by the underlying OS. This can make a program much less efficient, since each such request often triggers disk access, network activity, or some other operation that is relatively expensive.

To reduce this kind of overhead, the Java platform implements buffered I/O streams. Buffered input streams read data from a memory area known as a buffer; the native input API is called only when the buffer is empty. Similarly, buffered output streams write data to a buffer, and the native output API is called only when the buffer is full.

There are four buffered stream classes used to wrap unbuffered streams: **BufferedInputStream** and **BufferedOutputStream** create buffered byte streams, while **BufferedReader** and **BufferedWriter** create buffered character streams.

```
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.

1.4.4FileOutputStream

```

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

public class ByteWriter {

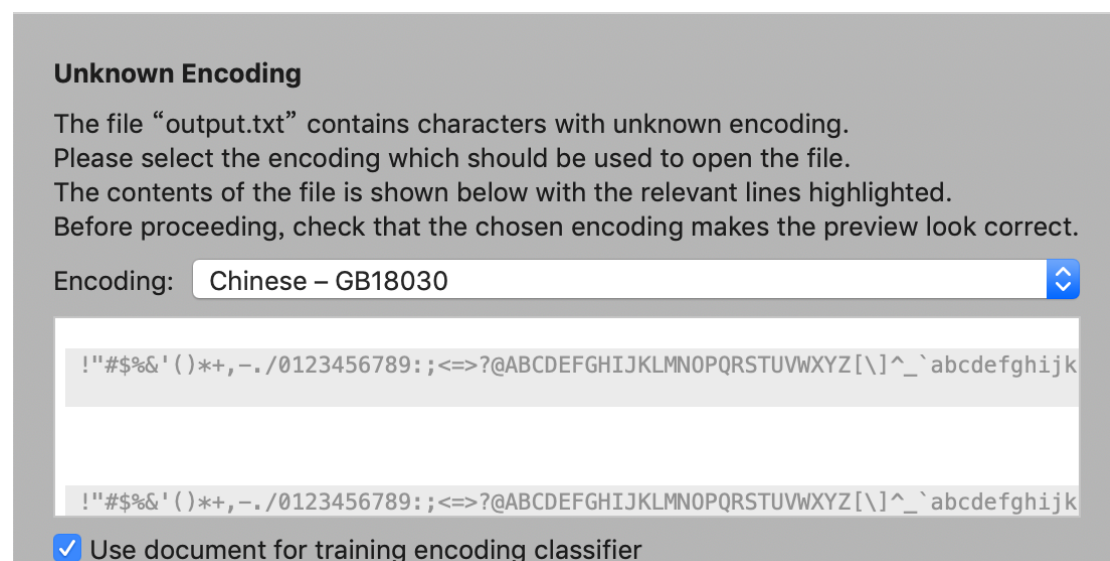
    public static void main(String[] args) {
        try (FileOutputStream fos = new FileOutputStream("output.txt")){

            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:



To solve this problem should open a binary document with Notepad++(install HexEditor) / VS Code(install extension: HexEditor)/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 `abcdefghijklmnopqrstuvwxyz
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 `abcdefghijklmnopqrstuvwxyz
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 `abcdefghijklmnopqrstuvwxyz
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 `abcdefghijklmnopqrstuvwxyz
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:

```
1 |
2 |
3 |
4 | !"#%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNopQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
5 |
6 | !"#%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNopQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
7 |
8 |
9 |
10 | !"#%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNopQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
11 |
12 |
13 | !"#%&'()*+,-./0123456789;<=>?@ABCDEFGHIJKLMNopQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

If you open it with UTF-16 BE:

[illegible]

𐄂𐄃𐄄𐄅𐄆𐄇𐄈𐄉𐄊𐄋𐄌𐄍𐄎𐄏𐄐𐄑𐄒𐄓𐄔𐄕𐄖𐄗𐄘𐄙𐄚𐄛𐄜𐄝𐄞𐄟𐄠𐄡𐄢𐄣𐄤𐄥𐄦𐄧𐄨𐄩𐄪𐄫𐄬𐄭𐄮𐄯𐄰𐄱𐄲𐄳𐄴𐄵𐄶𐄷𐄸𐄹𐄺𐄻𐄼𐄽𐄾𐄿𐅀𐅁𐅂𐅃𐅄𐅅𐅆𐅇𐅈𐅉𐅊𐅋𐅌𐅍𐅎𐅏𐅐𐅑𐅒𐅓𐅔𐅕𐅖𐅗𐅘𐅙𐅚𐅛𐅜𐅝𐅞𐅟𐅠𐅡𐅢𐅣𐅤𐅥𐅦𐅧𐅨𐅩𐅪𐅫𐅬𐅭𐅮𐅯𐅰𐅱𐅲𐅳𐅴𐅵𐅶𐅷𐅸𐅹𐅺𐅻𐅼𐅽𐅾𐅿𐆀𐆁𐆂𐆃𐆄𐆅𐆆𐆇𐆈𐆉𐆊𐆋𐆌𐆍𐆎𐆏𐆐𐆑𐆒𐆓𐆔𐆕𐆖𐆗𐆘𐆙𐆚𐆛𐆜𐆝𐆞𐆟𐆠𐆡𐆢𐆣𐆤𐆥𐆦𐆧𐆨𐆩𐆪𐆫𐆬𐆭𐆮𐆯𐆰𐆱𐆲𐆳𐆴𐆵𐆶𐆷𐆸𐆹𐆺𐆻𐆼𐆽𐆾𐆿𐇀𐇁𐇂𐇃𐇄𐇅𐇆𐇇𐇈𐇉𐇊𐇋𐇌𐇍𐇎𐇏𐇐𐇑𐇒𐇓𐇔𐇕𐇖𐇗𐇘𐇙𐇚𐇛𐇜𐇝𐇞𐇟𐇠𐇡𐇢𐇣𐇤𐇥𐇦𐇧𐇨𐇩𐇪𐇫𐇬𐇭𐇮𐇯𐇰𐇱𐇲𐇳𐇴𐇵𐇶𐇷𐇸𐇹𐇺𐇻𐇼𐇽𐇾𐇿𐈀𐈁𐈂𐈃𐈄𐈅𐈆𐈇𐈈𐈉𐈊𐈋𐈌𐈍𐈎𐈏𐈐𐈑𐈒𐈓𐈔𐈕𐈖𐈗𐈘𐈙𐈚𐈛𐈜𐈝𐈞𐈟𐈠𐈡𐈢𐈣𐈤𐈥𐈦𐈧𐈨𐈩𐈪𐈫𐈬𐈭𐈮𐈯𐈰𐈱𐈲𐈳𐈴𐈵𐈶𐈷𐈸𐈹𐈺𐈻𐈼𐈽𐈾𐈿𐉀𐉁𐉂𐉃𐉄𐉅𐉆𐉇𐉈𐉉𐉊𐉋𐉌𐉍𐉎𐉏𐉐𐉑𐉒𐉓𐉔𐉕𐉖𐉗𐉘𐉙𐉚𐉛𐉜𐉝𐉞𐉟𐉠𐉡𐉢𐉣𐉤𐉥𐉦𐉧𐉨𐉩𐉪𐉫𐉬𐉭𐉮𐉯𐉰𐉱𐉲𐉳𐉴𐉵𐉶𐉷𐉸𐉹𐉺𐉻𐉼𐉽𐉾𐉿𐊀𐊁𐊂𐊃𐊄𐊅𐊆𐊇𐊈𐊉𐊊𐊋𐊌𐊍𐊎𐊏𐊐𐊑𐊒𐊓𐊔𐊕𐊖𐊗𐊘𐊙𐊚𐊛𐊜𐊝𐊞𐊟𐊠𐊡𐊢𐊣𐊤𐊥𐊦𐊧𐊨𐊩𐊪𐊫𐊬𐊭𐊮𐊯𐊰𐊱𐊲𐊳𐊴𐊵𐊶𐊷𐊸𐊹𐊺𐊻𐊼𐊽𐊾𐊿𐋀𐋁𐋂𐋃𐋄𐋅𐋆𐋇𐋈𐋉𐋊𐋋𐋌𐋍𐋎𐋏𐋐𐋑𐋒𐋓𐋔𐋕𐋖𐋗𐋘𐋙𐋚𐋛𐋜𐋝𐋞𐋟𐋠𐋡𐋢𐋣𐋤𐋥𐋦𐋧𐋨𐋩𐋪𐋫𐋬𐋭𐋮𐋯𐋰𐋱𐋲𐋳𐋴𐋵𐋶𐋷𐋸𐋹𐋺𐋻𐋼𐋽𐋾𐋿𐌀𐌁𐌂𐌃𐌄𐌅𐌆𐌇𐌈𐌉𐌊𐌋𐌌𐌍𐌎𐌏𐌐𐌑𐌒𐌓𐌔𐌕𐌖𐌗𐌘𐌙𐌚𐌛𐌜𐌝𐌞𐌟𐌠𐌡𐌢𐌣𐌤𐌥𐌦𐌧𐌨𐌩𐌪𐌫𐌬𐌭𐌮𐌯𐌰𐌱𐌲𐌳𐌴𐌵𐌶𐌷𐌸𐌹𐌺𐌻𐌼𐌽𐌾𐌿𐍀𐍁𐍂𐍃𐍄𐍅𐍆𐍇𐍈𐍉𐍊𐍋𐍌𐍍𐍎𐍏𐍐𐍑𐍒𐍓𐍔𐍕𐍖𐍗𐍘𐍙𐍚𐍛𐍜𐍝𐍞𐍟𐍠𐍡𐍢𐍣𐍤𐍥𐍦𐍧𐍨𐍩𐍪𐍫𐍬𐍭𐍮𐍯𐍰𐍱𐍲𐍳𐍴𐍵𐍶𐍷𐍸𐍹𐍺𐍻𐍼𐍽𐍾𐍿𐎀𐎁𐎂𐎃𐎄𐎅𐎆𐎇𐎈𐎉𐎊𐎋𐎌𐎍𐎎𐎏𐎐𐎑𐎒𐎓𐎔𐎕𐎖𐎗𐎘𐎙𐎚𐎛𐎜𐎝𐎞𐎟𐎠𐎡𐎢𐎣𐎤𐎥𐎦𐎧𐎨𐎩𐎪𐎫𐎬𐎭𐎮𐎯𐎰𐎱𐎲𐎳𐎴𐎵𐎶𐎷𐎸𐎹𐎺𐎻𐎼𐎽𐎾𐎿𐏀𐏁𐏂𐏃𐏄𐏅𐏆𐏇𐏈𐏉𐏊𐏋𐏌𐏍𐏎𐏏𐏐𐏑𐏒𐏓𐏔𐏕𐏖𐏗𐏘𐏙𐏚𐏛𐏜𐏝𐏞𐏟𐏠𐏡𐏢𐏣𐏤𐏥𐏦𐏧𐏨𐏩𐏪𐏫𐏬𐏭𐏮𐏯𐏰𐏱𐏲𐏳𐏴𐏵𐏶𐏷𐏸𐏹𐏺𐏻𐏼𐏽𐏾𐏿𐐀𐐁𐐂𐐃𐐄𐐅𐐆𐐇𐐈𐐉𐐊𐐋𐐌𐐍𐐎𐐏𐐐𐐑𐐒𐐓𐐔𐐕𐐖𐐗𐐘𐐙𐐚𐐛𐐜𐐝𐐞𐐟𐐠𐐡𐐢𐐣𐐤𐐥𐐦𐐧𐐨𐐩𐐪𐐫𐐬𐐭𐐮𐐯𐐰𐐱𐐲𐐳𐐴𐐵𐐶𐐷𐐸𐐹𐐺𐐻𐐼𐐽𐐾𐐿𐑀𐑁𐑂𐑃𐑄𐑅𐑆𐑇𐑈𐑉𐑊𐑋𐑌𐑍𐑎𐑏𐑐𐑑𐑒𐑓𐑔𐑕𐑖𐑗𐑘𐑙𐑚𐑛𐑜𐑝𐑞𐑟𐑠𐑡𐑢𐑣𐑤𐑥𐑦𐑧𐑨𐑩𐑪𐑫𐑬𐑭𐑮𐑯𐑰𐑱𐑲𐑳𐑴𐑵𐑶𐑷𐑸𐑹𐑺𐑻𐑼𐑽𐑾𐑿𐒀𐒁𐒂𐒃𐒄𐒅𐒆𐒇𐒈𐒉𐒊𐒋𐒌𐒍𐒎𐒏𐒐𐒑𐒒𐒓𐒔𐒕𐒖𐒗𐒘𐒙𐒚𐒛𐒜𐒝𐒞𐒟𐒠𐒡𐒢𐒣𐒤𐒥𐒦𐒧𐒨𐒩𐒪𐒫𐒬𐒭𐒮𐒯𐒰𐒱𐒲𐒳𐒴𐒵𐒶𐒷𐒸𐒹𐒺𐒻𐒼𐒽𐒾𐒿𐓀𐓁𐓂𐓃𐓄𐓅𐓆𐓇𐓈𐓉𐓊𐓋𐓌𐓍𐓎𐓏𐓐𐓑𐓒𐓓𐓔𐓕𐓖𐓗𐓘𐓙𐓚𐓛𐓜𐓝𐓞𐓟𐓠𐓡𐓢𐓣𐓤𐓥𐓦𐓧𐓨𐓩𐓪𐓫𐓬𐓭𐓮𐓯𐓰𐓱𐓲𐓳𐓴𐓵𐓶𐓷𐓸𐓹𐓺𐓻𐓼𐓽𐓾𐓿𐔀𐔁𐔂𐔃𐔄𐔅𐔆𐔇𐔈𐔉𐔊𐔋𐔌𐔍𐔎𐔏𐔐𐔑𐔒𐔓𐔔𐔕𐔖𐔗𐔘𐔙𐔚𐔛𐔜𐔝𐔞𐔟𐔠𐔡𐔢𐔣𐔤𐔥𐔦𐔧𐔨𐔩𐔪𐔫𐔬𐔭𐔮𐔯𐔰𐔱𐔲𐔳𐔴𐔵𐔶𐔷𐔸𐔹𐔺𐔻𐔼𐔽𐔾𐔿𐕀𐕁𐕂𐕃𐕄𐕅𐕆𐕇𐕈𐕉𐕊𐕋𐕌𐕍𐕎𐕏𐕐𐕑𐕒𐕓𐕔𐕕𐕖𐕗𐕘𐕙𐕚𐕛𐕜𐕝𐕞𐕟𐕠𐕡𐕢𐕣𐕤𐕥𐕦𐕧𐕨𐕩𐕪𐕫𐕬𐕭𐕮𐕯𐕰𐕱𐕲𐕳𐕴𐕵𐕶𐕷𐕸𐕹𐕺𐕻𐕼𐕽𐕾𐕿𐖀𐖁𐖂𐖃𐖄𐖅𐖆𐖇𐖈𐖉𐖊𐖋𐖌𐖍𐖎𐖏𐖐𐖑𐖒𐖓𐖔𐖕𐖖𐖗𐖘𐖙𐖚𐖛𐖜𐖝𐖞𐖟𐖠𐖡𐖢𐖣𐖤𐖥𐖦𐖧𐖨𐖩𐖪𐖫𐖬𐖭𐖮𐖯𐖰𐖱𐖲𐖳𐖴𐖵𐖶𐖷𐖸𐖹𐖺𐖻𐖼𐖽𐖾𐖿𐗀𐗁𐗂𐗃𐗄𐗅𐗆𐗇𐗈𐗉𐗊𐗋𐗌𐗍𐗎𐗏𐗐𐗑𐗒𐗓𐗔𐗕𐗖𐗗𐗘𐗙𐗚𐗛𐗜𐗝𐗞𐗟𐗠𐗡𐗢𐗣𐗤𐗥𐗦𐗧𐗨𐗩𐗪𐗫𐗬𐗭𐗮𐗯𐗰𐗱𐗲𐗳𐗴𐗵𐗶𐗷𐗸𐗹𐗺𐗻𐗼𐗽𐗾𐗿𐘀𐘁𐘂𐘃𐘄𐘅𐘆𐘇𐘈𐘉𐘊𐘋𐘌𐘍𐘎𐘏𐘐𐘑𐘒𐘓𐘔𐘕𐘖𐘗𐘘𐘙𐘚𐘛𐘜𐘝𐘞𐘟𐘠𐘡𐘢𐘣𐘤𐘥𐘦𐘧𐘨𐘩𐘪𐘫𐘬𐘭𐘮𐘯𐘰𐘱𐘲𐘳𐘴𐘵𐘶𐘷𐘸𐘹𐘺𐘻𐘼𐘽𐘾𐘿𐙀𐙁𐙂𐙃𐙄𐙅𐙆𐙇𐙈𐙉𐙊𐙋𐙌𐙍𐙎𐙏𐙐𐙑𐙒𐙓𐙔𐙕𐙖𐙗𐙘𐙙𐙚𐙛𐙜𐙝𐙞𐙟𐙠𐙡𐙢𐙣𐙤𐙥𐙦𐙧𐙨𐙩𐙪𐙫𐙬𐙭𐙮𐙯𐙰𐙱𐙲𐙳𐙴𐙵𐙶𐙷𐙸𐙹𐙺𐙻𐙼𐙽𐙾𐙿𐚀𐚁𐚂𐚃𐚄𐚅𐚆𐚇𐚈𐚉𐚊𐚋𐚌𐚍𐚎𐚏𐚐𐚑𐚒𐚓𐚔𐚕𐚖𐚗𐚘𐚙𐚚𐚛𐚜𐚝𐚞𐚟𐚠𐚡𐚢𐚣𐚤𐚥𐚦𐚧𐚨𐚩𐚪𐚫𐚬𐚭𐚮𐚯𐚰𐚱𐚲𐚳𐚴𐚵𐚶𐚷𐚸𐚹𐚺𐚻𐚼𐚽𐚾𐚿𐛀𐛁𐛂𐛃𐛄𐛅𐛆𐛇𐛈𐛉𐛊𐛋𐛌𐛍𐛎𐛏𐛐𐛑𐛒𐛓𐛔𐛕𐛖𐛗𐛘𐛙𐛚𐛛𐛜𐛝𐛞𐛟𐛠𐛡𐛢𐛣𐛤𐛥𐛦𐛧𐛨𐛩𐛪𐛫𐛬𐛭𐛮𐛯𐛰𐛱𐛲𐛳𐛴𐛵𐛶𐛷𐛸𐛹𐛺𐛻𐛼𐛽𐛾𐛿𐜀𐜁𐜂𐜃𐜄𐜅𐜆𐜇𐜈𐜉𐜊𐜋𐜌𐜍𐜎𐜏𐜐𐜑𐜒𐜓𐜔𐜕𐜖𐜗𐜘𐜙𐜚𐜛𐜜𐜝𐜞𐜟𐜠𐜡𐜢𐜣𐜤𐜥𐜦𐜧𐜨𐜩𐜪𐜫𐜬𐜭𐜮𐜯𐜰𐜱𐜲𐜳𐜴𐜵𐜶𐜷𐜸𐜹𐜺𐜻𐜼𐜽𐜾𐜿𐝀𐝁𐝂𐝃𐝄𐝅𐝆𐝇𐝈𐝉𐝊𐝋𐝌𐝍𐝎𐝏𐝐𐝑𐝒𐝓𐝔𐝕𐝖𐝗𐝘𐝙𐝚𐝛𐝜𐝝𐝞𐝟𐝠𐝡𐝢𐝣𐝤𐝥𐝦𐝧𐝨𐝩𐝪𐝫𐝬𐝭𐝮𐝯𐝰𐝱𐝲𐝳𐝴𐝵𐝶𐝷𐝸𐝹𐝺𐝻𐝼𐝽𐝾𐝿𐞀𐞁𐞂𐞃𐞄𐞅𐞆𐞇𐞈𐞉𐞊𐞋𐞌𐞍𐞎𐞏𐞐𐞑𐞒𐞓𐞔𐞕𐞖𐞗𐞘𐞙𐞚𐞛𐞜𐞝𐞞𐞟𐞠𐞡𐞢𐞣𐞤𐞥𐞦𐞧𐞨𐞩𐞪𐞫𐞬𐞭𐞮𐞯𐞰𐞱𐞲𐞳𐞴𐞵𐞶𐞷𐞸𐞹𐞺𐞻𐞼𐞽𐞾𐞿𐟀𐟁𐟂𐟃𐟄𐟅𐟆𐟇𐟈𐟉𐟊𐟋𐟌𐟍𐟎𐟏𐟐𐟑𐟒𐟓𐟔𐟕𐟖𐟗𐟘𐟙𐟚𐟛𐟜𐟝𐟞𐟟𐟠𐟡𐟢𐟣𐟤𐟥𐟦𐟧𐟨𐟩𐟪𐟫𐟬𐟭𐟮𐟯𐟰𐟱𐟲𐟳𐟴𐟵𐟶𐟷𐟸𐟹𐟺𐟻𐟼𐟽𐟾𐟿𐠀𐠁𐠂𐠃𐠄𐠅𐠆𐠇𐠈𐠉𐠊𐠋𐠌𐠍𐠎𐠏𐠐𐠑𐠒𐠓𐠔𐠕𐠖𐠗𐠘𐠙𐠚𐠛𐠜𐠝𐠞𐠟𐠠𐠡𐠢𐠣𐠤𐠥𐠦𐠧𐠨𐠩𐠪𐠫𐠬𐠭𐠮𐠯𐠰𐠱𐠲𐠳𐠴𐠵𐠶𐠷𐠸𐠹𐠺𐠻𐠼𐠽𐠾𐠿𐡀𐡁𐡂𐡃𐡄𐡅𐡆𐡇𐡈𐡉𐡊𐡋𐡌𐡍𐡎𐡏𐡐𐡑𐡒𐡓𐡔𐡕𐡖𐡗𐡘𐡙𐡚𐡛𐡜𐡝𐡞𐡟𐡠𐡡𐡢𐡣𐡤𐡥𐡦𐡧𐡨𐡩𐡪𐡫𐡬𐡭𐡮𐡯𐡰𐡱𐡲𐡳𐡴𐡵𐡶𐡷𐡸𐡹𐡺𐡻𐡼𐡽𐡾𐡿𐢀𐢁𐢂𐢃𐢄𐢅𐢆𐢇𐢈𐢉𐢊𐢋𐢌𐢍𐢎𐢏𐢐𐢑𐢒𐢓𐢔𐢕𐢖𐢗𐢘𐢙𐢚𐢛𐢜𐢝𐢞𐢟𐢠𐢡𐢢𐢣𐢤𐢥𐢦𐢧𐢨𐢩𐢪𐢫𐢬𐢭𐢮𐢯𐢰𐢱𐢲𐢳𐢴𐢵𐢶𐢷𐢸𐢹𐢺𐢻𐢼𐢽𐢾𐢿𐣀𐣁𐣂𐣃𐣄𐣅𐣆𐣇𐣈𐣉𐣊𐣋𐣌𐣍𐣎𐣏𐣐𐣑𐣒𐣓𐣔𐣕𐣖𐣗𐣘𐣙𐣚𐣛𐣜𐣝𐣞𐣟𐣠𐣡𐣢𐣣𐣤𐣥𐣦𐣧𐣨𐣩𐣪𐣫𐣬𐣭𐣮𐣯𐣰𐣱𐣲𐣳𐣴𐣵𐣶𐣷𐣸𐣹𐣺𐣻𐣼𐣽𐣾𐣿𐤀𐤁𐤂𐤃𐤄𐤅𐤆𐤇𐤈𐤉𐤊𐤋𐤌𐤍𐤎𐤏𐤐𐤑𐤒𐤓𐤔𐤕𐤖𐤗𐤘𐤙𐤚𐤛𐤜𐤝𐤞𐤟𐤠𐤡𐤢𐤣𐤤𐤥𐤦𐤧𐤨𐤩𐤪𐤫𐤬𐤭𐤮𐤯𐤰𐤱𐤲𐤳𐤴𐤵𐤶𐤷𐤸𐤹𐤺𐤻𐤼𐤽𐤾𐤿𐥀𐥁𐥂𐥃𐥄𐥅𐥆𐥇𐥈𐥉𐥊𐥋𐥌𐥍𐥎𐥏𐥐𐥑𐥒𐥓𐥔𐥕𐥖𐥗𐥘𐥙𐥚𐥛𐥜𐥝𐥞𐥟𐥠𐥡𐥢𐥣𐥤𐥥𐥦𐥧𐥨𐥩𐥪𐥫𐥬𐥭𐥮𐥯𐥰𐥱𐥲𐥳𐥴𐥵𐥶𐥷𐥸𐥹𐥺𐥻𐥼𐥽𐥾𐥿𐦀𐦁𐦂𐦃𐦄𐦅𐦆𐦇𐦈𐦉𐦊𐦋𐦌𐦍𐦎𐦏𐦐𐦑𐦒𐦓𐦔𐦕𐦖𐦗𐦘𐦙𐦚𐦛𐦜𐦝𐦞𐦟𐦠𐦡𐦢𐦣𐦤𐦥𐦦𐦧𐦨𐦩𐦪𐦫𐦬𐦭𐦮𐦯𐦰𐦱𐦲𐦳𐦴𐦵𐦶𐦷𐦸𐦹𐦺𐦻𐦼𐦽𐦾𐦿𐧀𐧁𐧂𐧃𐧄𐧅𐧆𐧇𐧈𐧉𐧊𐧋𐧌𐧍𐧎𐧏𐧐𐧑𐧒𐧓𐧔𐧕𐧖𐧗𐧘𐧙𐧚𐧛𐧜𐧝𐧞𐧟𐧠𐧡𐧢𐧣𐧤𐧥𐧦𐧧𐧨𐧩𐧪𐧫𐧬𐧭𐧮𐧯𐧰𐧱𐧲𐧳𐧴𐧵𐧶𐧷𐧸𐧹𐧺𐧻𐧼𐧽𐧾𐧿𐨀𐨁𐨂𐨃𐨄𐨅𐨆𐨇𐨈𐨉𐨊𐨋𐨌𐨍𐨎𐨏𐨐𐨑𐨒𐨓𐨔𐨕𐨖𐨗𐨘𐨙𐨚𐨛𐨜𐨝𐨞𐨟𐨠𐨡𐨢𐨣𐨤𐨥𐨦𐨧𐨨𐨩𐨪𐨫𐨬𐨭𐨮𐨯𐨰𐨱𐨲𐨳𐨴𐨵𐨶𐨷𐨹𐨺𐨸𐨻𐨼𐨽𐨾𐨿𐩀𐩁𐩂𐩃𐩄𐩅𐩆𐩇𐩈𐩉𐩊𐩋𐩌𐩍𐩎𐩏𐩐𐩑𐩒𐩓𐩔𐩕𐩖𐩗𐩘𐩙𐩚𐩛𐩜𐩝𐩞𐩟𐩠𐩡𐩢𐩣𐩤𐩥𐩦𐩧𐩨𐩩𐩪𐩫𐩬𐩭𐩮𐩯𐩰𐩱𐩲𐩳𐩴𐩵𐩶𐩷𐩸𐩹𐩺𐩻𐩼𐩽𐩾𐩿𐪀𐪁𐪂𐪃𐪄𐪅𐪆𐪇𐪈𐪉𐪊𐪋𐪌𐪍𐪎𐪏𐪐𐪑𐪒𐪓𐪔𐪕𐪖𐪗𐪘𐪙𐪚𐪛𐪜𐪝𐪞𐪟𐪠𐪡𐪢𐪣𐪤𐪥𐪦𐪧𐪨𐪩𐪪𐪫𐪬𐪭𐪮𐪯𐪰𐪱𐪲𐪳𐪴𐪵𐪶𐪷𐪸𐪹𐪺𐪻𐪼𐪽𐪾𐪿𐫀𐫁𐫂𐫃𐫄𐫅𐫆𐫇𐫈𐫉𐫊𐫋𐫌𐫍𐫎𐫏𐫐𐫑𐫒𐫓𐫔𐫕𐫖𐫗𐫘𐫙𐫚𐫛𐫜𐫝𐫞𐫟𐫠𐫡𐫢𐫣𐫤𐫦𐫥𐫧𐫨𐫩𐫪𐫫𐫬𐫭𐫮𐫯𐫰𐫱𐫲𐫳𐫴𐫵𐫶𐫷𐫸𐫹𐫺𐫻𐫼𐫽𐫾𐫿𐬀𐬁𐬂𐬃𐬄𐬅𐬆𐬇𐬈𐬉𐬊𐬋𐬌𐬍𐬎𐬏𐬐𐬑𐬒𐬓𐬔𐬕𐬖𐬗𐬘𐬙𐬚𐬛𐬜𐬝𐬞𐬟𐬠𐬡𐬢𐬣𐬤𐬥𐬦𐬧𐬨𐬩𐬪𐬫𐬬𐬭𐬮𐬯𐬰𐬱𐬲𐬳𐬴𐬵𐬶𐬷𐬸𐬹𐬺𐬻𐬼𐬽𐬾𐬿𐭀𐭁𐭂𐭃𐭄𐭅𐭆𐭇𐭈𐭉𐭊𐭋𐭌𐭍𐭎𐭏𐭐𐭑𐭒𐭓𐭔𐭕𐭖𐭗𐭘𐭙𐭚𐭛𐭜𐭝𐭞𐭟𐭠𐭡𐭢𐭣𐭤𐭥𐭦𐭧𐭨𐭩𐭪𐭫𐭬𐭭𐭮𐭯𐭰𐭱𐭲𐭳𐭴𐭵𐭶𐭷𐭸𐭹𐭺𐭻𐭼𐭽𐭾𐭿𐮀𐮁𐮂𐮃𐮄𐮅𐮆𐮇𐮈𐮉𐮊𐮋𐮌𐮍𐮎𐮏𐮐𐮑𐮒𐮓𐮔𐮕𐮖𐮗𐮘𐮙𐮚𐮛𐮜𐮝𐮞𐮟𐮠𐮡𐮢𐮣𐮤𐮥𐮦𐮧𐮨𐮩𐮪𐮫𐮬𐮭𐮮𐮯𐮰𐮱𐮲𐮳𐮴𐮵𐮶𐮷𐮸𐮹𐮺𐮻𐮼𐮽𐮾𐮿𐯀𐯁𐯂𐯃𐯄𐯅𐯆𐯇𐯈𐯉𐯊𐯋𐯌𐯍𐯎𐯏𐯐𐯑𐯒𐯓𐯔𐯕𐯖𐯗𐯘𐯙𐯚𐯛𐯜𐯝𐯞𐯟𐯠𐯡𐯢𐯣𐯤𐯥𐯦𐯧𐯨𐯩𐯪𐯫𐯬𐯭𐯮𐯯𐯰𐯱𐯲𐯳𐯴𐯵𐯶𐯷𐯸𐯹𐯺𐯻𐯼𐯽𐯾𐯿𐰀𐰁𐰂𐰃𐰄𐰅𐰆𐰇𐰈𐰉𐰊𐰋𐰌𐰍𐰎𐰏𐰐𐰑𐰒𐰓𐰔𐰕𐰖𐰗𐰘𐰙𐰚𐰛𐰜𐰝𐰞𐰟𐰠𐰡𐰢𐰣𐰤𐰥𐰦𐰧𐰨𐰩𐰪𐰫𐰬𐰭𐰮𐰯𐰰𐰱𐰲𐰳𐰴𐰵𐰶𐰷𐰸𐰹𐰺𐰻𐰼𐰽𐰾𐰿𐱀𐱁𐱂𐱃𐱄𐱅𐱆𐱇𐱈𐱉𐱊𐱋𐱌𐱍𐱎𐱏𐱐𐱑𐱒𐱓𐱔𐱕𐱖𐱗𐱘𐱙𐱚𐱛𐱜𐱝𐱞𐱟𐱠𐱡𐱢𐱣𐱤𐱥𐱦𐱧𐱨𐱩𐱪𐱫𐱬𐱭𐱮𐱯𐱰𐱱𐱲𐱳𐱴𐱵𐱶𐱷𐱸𐱹𐱺𐱻𐱼𐱽𐱾𐱿𐲀𐲁𐲂𐲃𐲄𐲅𐲆𐲇𐲈𐲉𐲊𐲋𐲌𐲍𐲎𐲏𐲐𐲑𐲒𐲓𐲔𐲕𐲖𐲗𐲘𐲙𐲚𐲛𐲜𐲝𐲞𐲟𐲠𐲡𐲢𐲣𐲤𐲥𐲦𐲧𐲨𐲩𐲪𐲫𐲬𐲭𐲮𐲯𐲰𐲱𐲲𐲳𐲴𐲵𐲶𐲷𐲸𐲹𐲺𐲻𐲼𐲽𐲾𐲿𐳀𐳁𐳂𐳃𐳄𐳅𐳆𐳇𐳈𐳉𐳊𐳋𐳌𐳍𐳎𐳏𐳐𐳑𐳒𐳓𐳔𐳕𐳖𐳗𐳘𐳙𐳚𐳛𐳜𐳝𐳞𐳟𐳠𐳡𐳢𐳣𐳤𐳥𐳦𐳧𐳨𐳩𐳪𐳫𐳬𐳭𐳮𐳯𐳰𐳱𐳲𐳳𐳴𐳵𐳶𐳷𐳸𐳹𐳺𐳻𐳼𐳽𐳾𐳿𐴀𐴁

1.4.6 BufferedWriter

```
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 BufferedWriter {

    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.

1.4.7 Scanning and Formatting

Programming I/O often involves translating to and from the neatly formatted data humans like to work with. To assist you with these chores, the Java platform provides two APIs. The scanner API breaks input into individual tokens associated with bits of data. The formatting API assembles data into nicely formatted, human-readable form.

Before, we usually use scanner to read data from console like this:

```
Scanner s = new Scanner( System.in );
s.nextDoulbe();
s.next();
```

now we can also use it to read data from a file.

```
Scanner s = new Scanner(new BufferedReader(new FileReader("1.txt")));
```

Formatter example:

```
Formatter formatter = new Formatter(new File("1.txt"));  
formatter.format ("%s %f", "Pi is", 3.0/7);  
formatter.flush();
```

2 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.

2.1 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.

2.2 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("%02X ", 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

3 Some pitfalls

3.1 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.

3.2 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**.

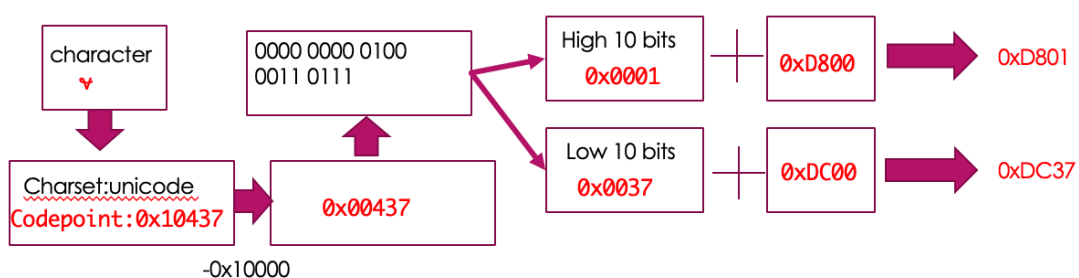
`0x10437` to `0xd801dc37`

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%A7%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.5lcto.com/cnn237111/1080628>

<https://docs.oracle.com/javase/tutorial/essential/io/index.html>

<https://docs.oracle.com/javase/7/docs/api/java/io/package-summary.html>