day10_缓冲流、转换流、序列化流、打印流

昨天学习了一些基本的流(4个 FileXxx 文件流)作为 IO 流的入门，今天我们要见识一些更强大的流。比如能够高效读写的缓冲流，能够转换编码的转换流，能够持久化存储对象的序列化流等等。这些功能更为强大的流，都是在基本的流对象基础之上创建而来的，相当于是对基本流对象的一种增强。

一.缓冲流

缓冲流，是一种能够高效读写的 IO 流，是对基本流对象的一种增强。缓冲流的基本原理，是在创建缓冲流对象时，会创建一个内置的默认大小的缓冲区数组，通过该缓冲区数组来读写数据，数据不直接读写到目的地，而是暂时存放在缓冲区数组中，当该缓冲区数组满了，才会将数据读写到目的地。这样就减少了系统 IO 次数，从而提高读写的效率。

- 字节缓冲流：BufferedInputStream、BufferedOutputStream
- 字符缓冲流：BufferedReader、BufferedWriter

1.字节缓冲流

2.字符缓冲流

3.练习：文本排序

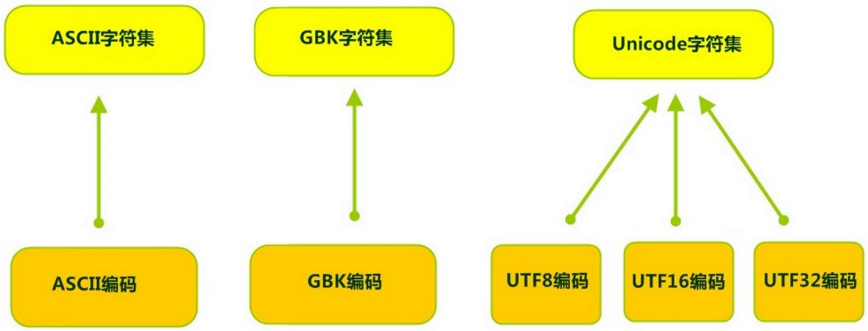
二.转换流

1.字符编码和字符集

计算机中储存的信息都是用二进制数表示的，而我们在屏幕上看到的数字、英文、标点符号、汉字等字符是二进制数转换之后的结果。按照某种规则，将字符转换为二进制数存储到计算机中，称为编码。反之，将存储在计算机中的二进制数按照某种规则解析显示出来，称为解码。比如，按照 A 规则存储，同样按照 A 规则解析，那么就能显示正确的文本符号。反之，如果按照 A 规则存储，再按照 B 规则解析，就会出现乱码。

- 字符编码(Character Encoding)：就是一套自然语言的字符与二进制数之间的对应规则
- 字符集(Charset)：也叫编码表，是使用字符编码规则形成的一个字符与数字的对照表

所以计算机要准确的存储和识别各种符号，就需要对各种符号进行字符编码，使用不同的字符编码规则所形成的编码表也不尽相同。常见的字符编码和字符集如下：



- ASCII 字符集：美国信息交换标准代码，是基于拉丁字母的一套编码表，用于显示现代英语。主要包括控制字符(回车键、退格、换行键等)和可显示字符(英文大小写字符、阿拉伯数字和西文符号等)

- GBK 字符集：最常用的中文编码表，使用两个字节的数字来表示一个汉字，共收录了21003个汉字，同时支持繁体汉字以及日韩汉字等

- Unicode 字符集：Unicode 编码表为表达任意语言的任意字符而设计，是业界的一种标准，也称为统一码、标准万国码。它共有三种编码方案，UTF-8、UTF-16、UTF-32，其中最常用的是 UTF-8 编码。

  UTF-8 编码可以用来表示 Unicode 编码表中的任何字符，它是电子邮件、网页及其他存储或传送文字的应用中，优先采用的编码方式。互联网工程工作小组（IETF）要求所有互联网协议都必须支持 UTF-8 编码格式，所以开发 Web 应用也要使用 UTF-8 编码。它使用一至四个字节的数字为每个字符进行编码，编码规则如下：
  (1).128个 ASCII 字符集中的字符，只需一个字节编码
  (2).拉丁文等字符，需要二个字节编码
  (3).大部分常用字(含中文)，使用三个字节编码
  (4).其他极少使用的 Unicode 辅助字符，使用四字节编码

2.转换流的原理与使用

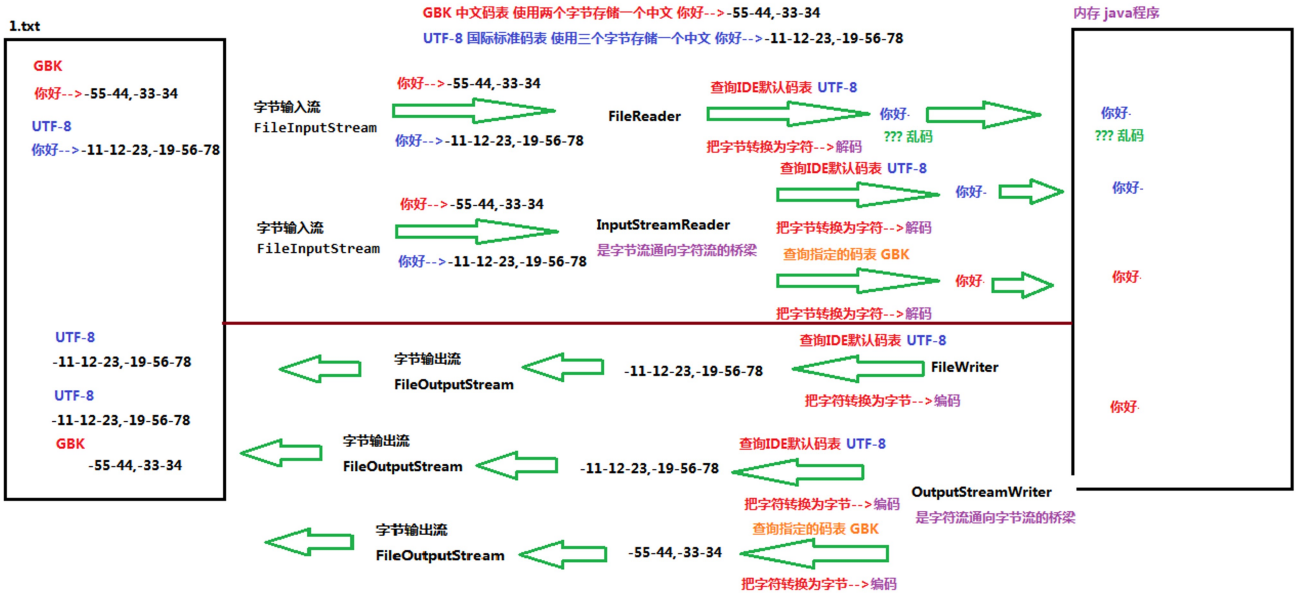
在 IDEA 中，使用 FileReader 读取项目中的文本文件，由于 IDEA 默认的是 UTF-8 编码，所以没有任何问题。但是，当读取Windows 系统中创建的文本文件时，由于 Windows 系统默认是 GBK 编码，就会出现乱码。

```
public class DemoMain {
    public static void main(String[] args) throws IOException {

        FileReader fr = new FileReader( fileName: "day10_code\\File_GBK.txt");

        int length;
        while ((length = fr.read()) != -1) {
            System.out.print((char) length); // 输出乱码���
        }

        fr.close();
    }
}
```

FileReader 类的原理、转换流的原理：

使用字符输入流 FileReader 读取数据时，会 new 一个字节输入流 FileInputStream 对象用来读取数据的字节，然后会将读取到的字节数据传递给字符输入流 FileReader，FileReader 会使用 IDE 默认的编码方式将字节数据转换为字符数据进行传输。

转换流 InputStreamReader 的读取原理与 FileReader 类似，底层也是使用一个字节输入流 FileInputStream 对象来读取字节数据，然后将字节数据传递给转换流 InputStreamReader。但是它可以使用指定的编码方式将字节数据转换为字符数据进行传输，而不像 FileReader 那样只能使用 IDE 默认的编码方式，这样就可以避免乱码现象的出现。所以转换流本质上还是一种字符流，只不过是对普通字符流做了改进。



(1).字符输出转换流 OutputStreamWriter

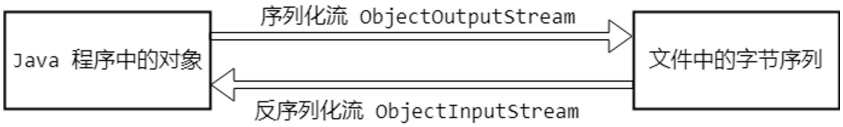
(2).字符输入转换流 InputStreamReader

(3).练习：转换文件编码

三.序列化流

Java 提供了一种对象序列化的机制，即用一个字节序列可以表示一个对象，该字节序列包含该对象的数据、对象的类型、对象中存储的属性等信息。通过序列化流 ObjectOutputStream 可以将对象转化为相应的字节序列，并将该字节序列写出到文件，此时相当于在文件中持久保存了一个对象的信息。这称为对象的序列化。

反之，还可以通过反序列化流 ObjectInputStream 将该字节序列从文件中读取回来，并根据读取到的字节序列来重构对象，此时相当于使用该字节序列在程序中创建了一个对象。这称为对象的反序列化。



1.序列化流 ObjectOutputStream

2.反序列化流 ObjectInputStream

3.练习：序列化集合

四.打印流