Introduction to Java

Special Care Set for Java Begineers/ Ex 13

・Notice!
The motivation of this page is to lead regular exercises.
We prepare many useful information in this page as hints of exercises, but this page has no answer.
If you have question, please don't hesitate to ask and discuss your teacher and teaching assistants.
このページは、通常の演習問題への導きとして作られています。
多くの有用な情報をヒントとして準備しておりますが、ここに回答が隠されているとかはありません。
もし、演習に対して質問等ございましたら、遠慮なく教員・TAに質問・議論をしてください。

Example Problem: ファイル入出力の使い方とその性質
Usage example and characteristic of file I/O methods.

今週はファイル入出力周りについて色々と見ていきます。
ファイル入出力は、『計算機』としてプログラムを作成するときに非常に重要な位置を占めます。
外部からのファイル入出力が無いと、毎回毎回ソースコード中にデータを記述する必要が出てきますので、大規模に使うことが出来ません。
Javaはそのための機構をしっかりと持っています。今回はそのファイル入出力について、いくつかの例を見ていきましょう。

1.1バイト毎にデータを書き込み/読み込みをするストリーム
"FileInputStream/FileOutputStream"
FileInputStream / FileOutputStream: I/O Stream for each byte.

先ずは、一番原始的な『１バイト毎』の入出力の方法を紹介します。 FileInputStream / FileOutputStreamは、１バイト毎のデータの入出力を行うことが出来ます。
ゲーム等のデータ格納では、こういった手法を良く使いますが、テキスト読み込み等ではあまり使い勝手が良くないかもしれません。

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

public class ByteStreamTest {
        public static void main(String[] args) {
                try {
            FileInputStream fin = new FileInputStream("in.txt");
            FileOutputStream fout = new FileOutputStream("out.txt");
            int ch;
```

```
            while ((ch = fin.read()) != -1) {
                System.out.print(Integer.toHexString(ch) + " ");
                fout.write(ch);
            }
            fin.close();
            fout.close();
        } catch (IOException e) {
            System.out.println(e);
        }
        }
}
```

2. 文字コードを扱うことの出来るストリーム "InputStreamReader/OutputStreamWriter"
InputStreamReader / OutputStreamWriterr: I/O Stream for each character with character codes.

次に、InputStreamReader, OutputStreamWriterを紹介します。これらは、先に紹介したFileInputStream, FileOutputStreamを拡張して、文字コードに対応できるようにしたクラスです。サンプルコードでは、読み込みにShift-JIS, 書き込みにUTF-8を指定しています。このようにすると、文字コード変換を行うことが出来ます。
使い方は、FileInputStream, FileOutputStreamを先に定義して、それを受け渡す形でInputStreamReader, OutputStreamWriterを使うことが出来るようにします。

```
import java.io.*;
public class CharacterTest {

    public static void main(String[] args) {
        try {
            FileInputStream is = new FileInputStream("in.txt");
            FileOutputStream os = new FileOutputStream("out.txt");
            InputStreamReader isr = new InputStreamReader(is, "SJIS");
            OutputStreamWriter osw = new OutputStreamWriter(os,"UTF-8");
            int ch;
            while ((ch = isr.read()) != -1) {
                System.out.print(Integer.toHexString(ch) + " ");
                osw.write(ch);
            }
            isr.close();
            osw.close();
        } catch (IOException e) {
            System.out.println(e);
        }
    }
}
```

3. 行ごとに読み込むためのバッファを持つ BufferedReader / BufferedWriter と, テキストファイル入出力用クラス FileReader / FileWriter
BufferedReader / BufferedWriter; has buffered one line for I/O, FileReader / FileWriter: for text file I/O

最後に、テキストファイルの書き込み/読み込みを行単位で出来る方法を紹介します。

FileReader/FileWriterくらすは、先ほどのInputStreamReader等の煩雑なやり方を一まとめにして簡単にしたものです。
これと組み合わせて、行バッファを確保するBufferedReader/BufferedWriterを使用すると、Javaの標準入出力で行っている r eadLine等の行単位I/Oが可能となります。
なお、BufferedWriterには、writeLineはありません。改行が必要な場合は、改行文字 "\n"を入力するか、newLine()メソッドを使用してください。

```
import java.io.*;

public class BufferedExample {

    public static void main(String[] args) {
        try {
            FileReader in = new FileReader("in.txt");
            BufferedReader br = new BufferedReader(in);
            //Write in one line
            BufferedWriter bw = new BufferedWriter(new FileWriter("out.txt"));
            String line;
            while ((line = br.readLine()) != null) {
                System.out.println(line);
                bw.write(line);
                bw.newLine(); // in BufferedWriter, it has no writeLine.
            }
            br.close();
            in.close();
            bw.close();
        } catch (IOException e) {
            System.out.println(e);
        }
    }
}
```