2018 年 6 月 28 日（木）実施

ファイル操作とディレクトリ操作

今回の授業では，Java 言語でのファイル操作とディレクトリ操作とについて学習する。

ファイル操作

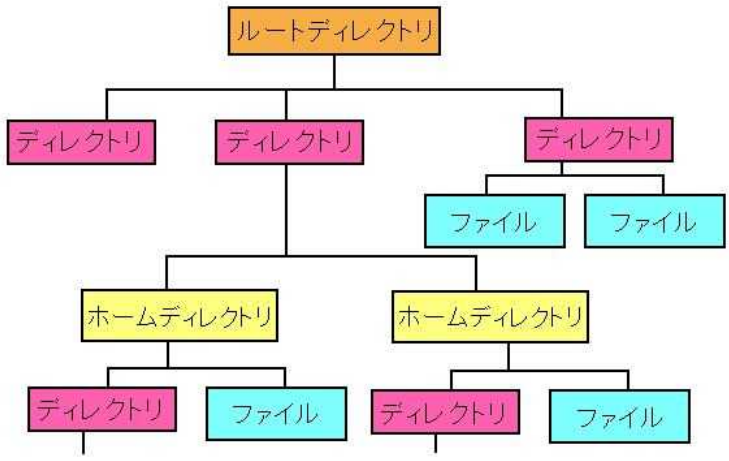
ファイル(File)とは，データの集合体のことで，JIS(日本工業規格)では，ファイルはレコードの集合体，レコードはデータの集合体と定義されている。ファイル操作は，次の順序で行う。なお，**ストリーム**とは，入力元または出力先を持つ，順序付けられたデータ列である。

1) ストリームを開く
2) ストリームからの入力，ストリームへの出力
3) ストリームを閉じる

Java 言語では，ファイル操作に関係するクラスが複数用意されている。ここでは，テキストファイルから文字を読み込むための簡易クラス FileReader，テキストファイルに文字を書き込むための簡易クラス FileWriter の使い方を中心に学ぶ。

ディレクトリ操作

ディレクトリ(Directory)とはファイルの管理簿で，階層構造を持つ。Java 言語でディレクトリ操作を行うには，ファイルおよびディレクトリの**パス名**の抽象表現を表すクラス File を用いる。パス名には絶対パス名(ルートディレクトリ[階層の先頭]から辿ったパス名)または相対パス名(カレント［Current；現在作業中の］ディレクトリから辿ったパス名)を用いる。



例題 1（ファイルからの読み込み）

次のプログラムは，ファイルから 1 行ずつ，カンマ区切りの文字列を読み込み，それを分解して各項目を表示するものである。これを入力，ビルドして，実行せよ。なお，これを実行する前に、ダウンロードしたファイル"in.csv"を workspace の Prog1 のフォルダにコピーしておく必要がある。ここで，クラス名は Sample11_1，ソースファイル名は Sample11_1.java とする。

```
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;


public class Sample11_1 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        String fname = "in.csv";

        try {
            BufferedReader br = new BufferedReader(new FileReader(fname));
            String iline = null;

            while ((iline = br.readLine()) != null) {
                String[] str =iline.split(",");
                System.out.println(str[0]+":"+"¥t データ 1="+str[1]+
                "¥t データ 2="+str[2]);
            }

            br.close();
        } catch (FileNotFoundException fe) {
            System.out.println(fname +"というファイルが開けません");
        } catch (IOException err) {
            System.out.println("IOException をキャッチ"+err);
        }
    }

}
```

【解説】

1. FileReader クラスのインスタンスが生成されるとストリームが開かれる。
2. BufferedReader クラスは，文字，配列，行をバッファリングすることによって，文字型入力ストリームからテキストを効率良く読み込むためのものである。クラスの階層は次の通りである。
   java.lang.Object → java.io.Reader → java.io.BufferedReader（→は上位を拡張）
3. readLine はテキスト行を読み込むメソッドである。
4. while 文の継続条件を満たさなくなるのはファイルの終わりに達した時である。
5. split(",") メソッドは文字列を","に一致する位置で分割する。
6. close メソッドはストリームを閉じる際に用いる。
7. 例外クラス FileNotFoundException は，指定されたパス名で示されるファイルが開けなかった例外を通知する。
8. 例外クラス IOException は，入出力処理の失敗，または割り込みの発生によって生成される例外に対応する。

例題 2（**ファイルへの書き込み**）

次のプログラムは，ファイルへ 1 行ずつ，カンマ区切りの文字列を書き込むものである。これを入力，ビルドして，実行せよ。ここで，クラス名は Sample11_2，ソースファイル名は Sample11_2.java とする。

```
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Scanner;


public class Sample11_2 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM = 5;
        int po;
        String na;
        Scanner sc = new Scanner(System.in);
        String fname = "out.csv";

        try {
            PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(fname)));

            for (int i=0; i<NUM; i++) {
                System.out.println((i+1)+"人目の");
                System.out.print("  氏名を入力してください：");
                na = sc.next();
                System.out.print("  点数を入力してください：");
                po = sc.nextInt();

                pw.printf("%s,%d%n", na, po);
            }

            pw.close();
        } catch (FileNotFoundException fe) {
            System.out.println(fname +"というファイルが開けません");
        } catch (IOException err) {
            System.out.println("IOException をキャッチ"+err);
        }
    }

}
```

【解説】

1. BufferedWriter クラスは，文字を**バッファリング**することによって，文字，配列または文字列を効率良く文字型出力ストリームに書き込むために用いる。
2. PrintWriter クラスは，オブジェクトの書式付き表現をテキスト出力ストリームに出力する際に用いる。そのインスタンスに対して，printf メソッドは書式付きの書き込みを行う。

例題 3（ディレクトリの表示）

次のプログラムは，ディレクトリのパス名を与えると，その内容を表示するものである。これを入力し，ビルドして，実行せよ。ここで，クラス名は Sample11_3，ソースファイル名は Sample11_3.java とする。なお，実行は次の様に 2 回行う。

[1 回目の実行]　そのまま実行する。

[2 回目の実行]　Eclipse では，『実行』→『実行構成』→『引数』→『プログラムの引数』と辿り，テキストエリア内に jimbo（jimbo の箇所は自分の名前に置き換える）を入力して実行する。

```
import java.io.File;

public class Sample11_3 {

    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("利用法：java Sample11_3 パス名");
            System.exit(1);
        }

        String dname = args[0];

        File directory = new File(dname);
        String[] dlist = directory.list();

        for (int i=0; i<dlist.length; i++) {
            System.out.println(dlist[i]);
        }
    }

}
```

【解説】

1. main メソッドの引数 args は String 型の配列を参照するが，その配列の要素にはプログラム実行時に与えられたコマンドライン引数（プログラム名 引数 1 引数 2 ・・・）が順に格納される。なお，args[0]には，最初のコマンドライン引数（引数 1）が格納される。
2. File クラスのインスタンスに対して，list メソッドは抽象パス名が示すディレクトリにあるファイル及びディレクトリを示す文字列の配列を返す。抽象パス名がディレクトリを示さない場合，このメソッドは null を返す。ディレクトリを示す場合は，文字列の配列が返される。なお，結果の配列の名前文字列は特定の順序にはならない。
3. length 属性は配列の要素数を返す。

演習

次のプログラムリストの空欄 1)　　～3)　　に適切な語句を埋めたプログラムを作成し，ビルドして実行せよ。ここで，プログラムのクラス名は Ex11，ソースプログラム名は Ex11.java とする。なお，このプログラムは，**10 個の整数データ**が各行に記述された入力用のファイル input.txt を開き，そこから読み込んだデータをソートした結果を，出力用のファイル output.txt を開いて

書き込むものである。なお，実行は次の様に 2 回行う。

[1 回目の実行]　そのまま実行する。

[2 回目の実行]　Eclipse では，workspace の直下の Prog1（プロジェクト名のディレクトリ）にダウンロードした input.txt を配置して実行する。

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;


public class Ex11 {

    public static void main(String[] args) {
        // TODO 自動生成されたメソッド・スタブ
        final int NUM = 10;
        int[] x = {0, 0, 0, 0, 0, 0, 0, 0, 0, 0};
        String fin = "input.txt";
        String fout = "output.txt";

        try {
            BufferedReader br = new BufferedReader(new FileReader(fin));
            PrintWriter pw = new PrintWriter(new BufferedWriter(new FileWriter(fout)));
            String iline = null;

            for (int i=0; i<NUM; i++) {
                iline = br.1)        ();
                x[i] = Integer.parseInt( 2)        );
            }

            sort(x);

            for (int i=0; i<NUM; i++) {
                3)     .println(x[i]);
            }

            pw.close();
            br.close();
        } catch (FileNotFoundException fe) {
            System.out.println("input.txt というファイルが開けません");
        } catch (IOException err) {
            System.out.println("IOException をキャッチ"+err);
        } catch (NumberFormatException exnf) {
            System.out.println("整数でないものが含まれています");
        }

    }

    private static void sort(int[] a){
        for(int i=0; i<a.length-1; i++)
            for(int j=i+1; j<a.length; j++)
```

```
                if(a[j]>a[i]){
                    int temp = a[i];
                    a[i] = a[j];
                    a[j]=temp;
                }
    }

}
```

提出物：

1) 例題 1 及び例題 3 の出力結果をコピーして貼り付けたテキストファイル res11.txt をメールに添付する。
2) 例題 2 で出力されたファイル out.csv をメールに添付する。
3) 演習で空欄を埋めたソースプログラムのファイル Ex11.java をメールに添付する。
4) 演習で出力されたファイル output.txt をメールに添付する。
5) 第 10 回の授業の復習の内容を埋めたファイル Review_10th.txt をメールに添付する。

＊ メールの件名は『プログラミング 1　第 11 回課題』（鍵括弧は要らない）とする。