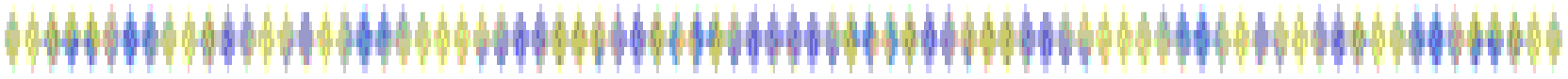
第11回　Javaゼミ



11.1　ファイルとディレクトリ

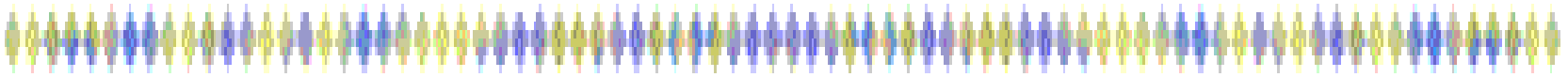
11.2　文字ストリーム

11.3　バッファ付き文字ストリーム

発表日：2007/6/22

発表者：阿部 竜之介

Fileクラス(1)



Fileクラス

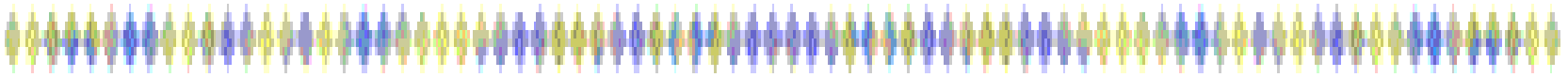
■ファイルまたはディレクトリのプロパティに関する情報をカプセル化する。新しいディレクトリを作成したり、既存のファイルとディレクトリを削除したりすることができる

Fileクラスの主なコンストラクタ

Fileコンストラクタ

File(String path)
File(String directoryPath, String filename)
File(File directory, String filename)

Fileクラス(2)



■Fileでは2つのchar定数が定義されている

- separatorChar

  ファイル名の中でディレクトリとファイルを区切る文字

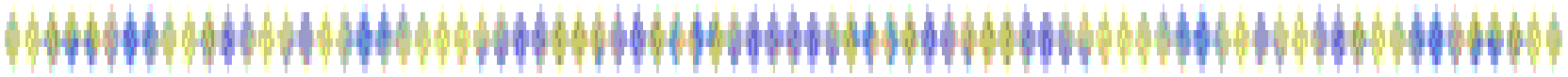
- pathSeparatorChar

  パスリスト内で構成要素を区切る文字

■どちらもプラットフォームにより異なる
（pathSeparatorCharはWindowsでは￥、UNIXでは / を取得する）

◆Fileクラスに定義されている主なインスタンスメソッドについてはP.322･323を参照

文字ストリーム



ストリーム

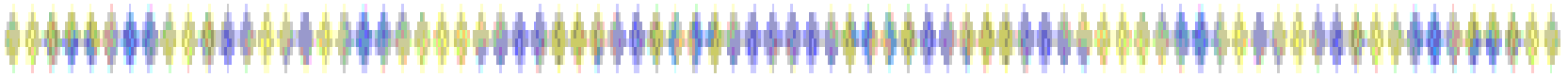
■データのソース（源流）またはデスティネーション（宛先）の抽象的な概念

■各種の物理デバイス（キーボード、ファイル、メモリバッファなど）と接続することが可能になる

文字ストリーム

■文字および文字列の読み取りおよび書き込みを行うことができる

Writerクラス



Writerクラス

- すべての文字出力ストリームで利用できる機能が定義される

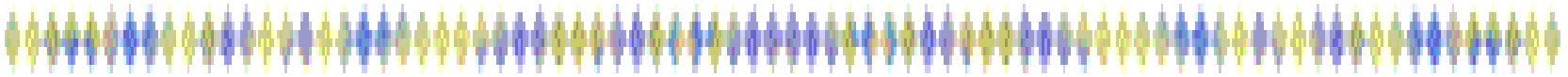
OutputStreamWriterクラス

- Writerを拡張したクラス
- 文字のストリームをバイトのストリームに変換する

FileWriterクラス

- OutputStreamWriterを拡張したクラス
- ファイルに文字を出力する

Readerクラス



Readerクラス

- すべての文字入力ストリームで利用できる機能が定義される

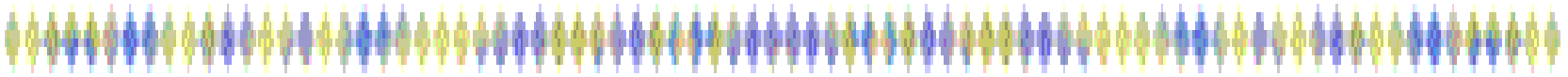
InputStreamReaderクラス

- Readerを拡張したクラス
- バイトのストリームを文字のストリームに変換する

FileReaderクラス

- InputStreamReaderを拡張したクラス
- ファイルから文字を入力する

バッファ付き文字ストリーム



バッファ付き文字ストリーム

- 文字ストリームの入力および出力をバッファに入れる
- バッファリングの利点は、物理デバイスへの読み取りおよび書き込みの回数を減らせる点

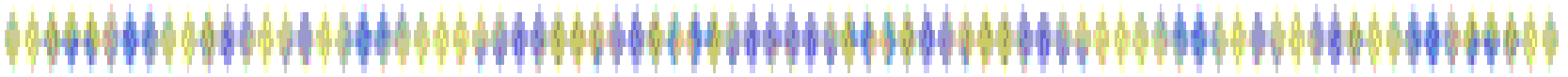
BufferedWriterクラス

- Writerを拡張したクラス
- 文字ストリームへの出力をバッファに入れる

BufferedReaderクラス

- Readerを拡張したクラス
- 文字ストリームからの入力をバッファに入れる

BufferedWriterクラス　例



```
import java.io.*;

class BufferedWriterDemo {
        public static void main ( String args[] ) {
                try {
                        FileWriter fw = new FileWriter( args[0] );
                        BufferedWriter bw = new BufferedWriter( fw );
                        for( int i = 0; i < 5; i++ ) {
                                bw.write( "Line " + i + "¥n" );
                        }
                        bw.close();
                }
                catch( Exception e ) {
                        System.out.println( "Exception: " + e );
                }
        }
}
```

BufferedWriterクラス　例-実行結果

■ファイル名を output.txt としてコマンドライン引数に取ると、output.txt ファイルが作成（上書き）される

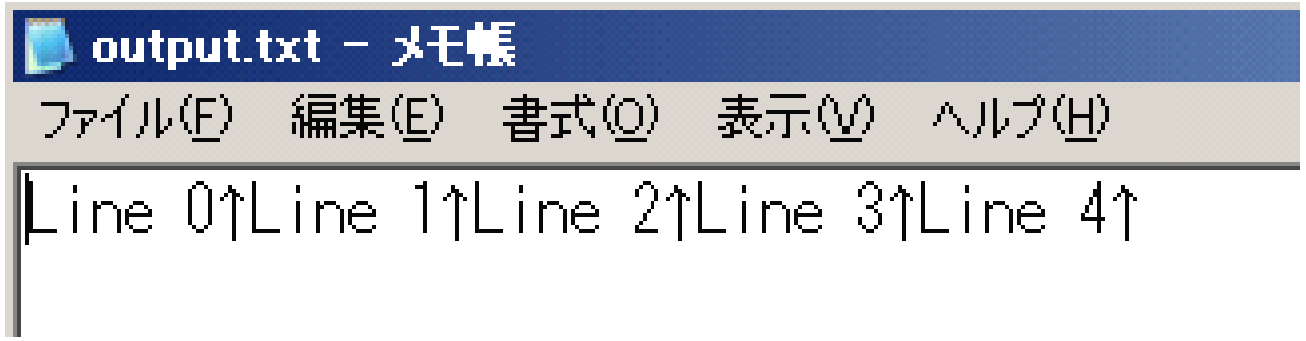
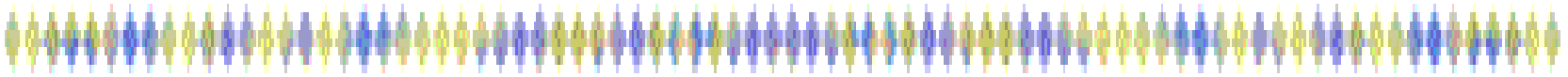


図1.ファイルの中身

BufferedReaderクラス　例



```
import java.io.*;

class BufferedReaderDemo {
        public static void main ( String args[] ) {
                try {
                        FileReader fr = new FileReader( args[0] );
                        BufferedReader br = new BufferedReader( fr );
                        String s;
                        while( ( s = br.readLine() ) != null ) {
                                System.out.println( s );
                        }
                        fr.close();
                }
                catch( Exception e ) {
                        System.out.println( "Exception: " + e );
                }
        }
}
```

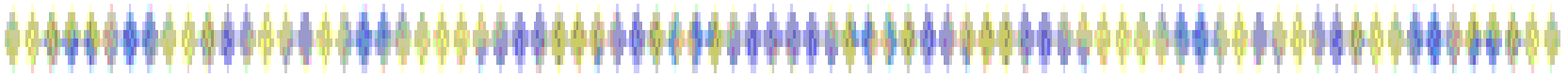
BufferedReaderクラス　例-実行結果

■ファイル名を output.txt としてコマンドライン引数に取り、先ほどの例で用いた output.txt ファイルを用いると以下のように表示される

実行結果

```
Line 0
Line 1
Line 2
Line 3
Line 4
```

練習問題



txtファイルを読み取り、ファイルに含まれる数値の和を求めて表示するプログラムを作成せよ。txtファイルの中身は以下の通りとし、数値は「,」で区切られているものとする（「1,2,3」なら 1+2+3=6 となる）。

txtファイルの中身

12,26,11,5
37,10,49,6
81,23,55,18
22,91,55,54