

● 授業のポイント

- ① InputStream と OutputStream (バイナリの入出力) について学習します。
- ② 例外発生時のクローズ処理について学習します。

● 学習項目

- ・ InputStream、OutputStream (教科書記載なし)
- ・ 例外発生時のクローズ処理 (教科書記載なし)

● J2Kad18D 「ファイルへの書き出し (OutputStream)」

1. バイナリと文字列

ファイルやネットワークなどからデータを読み込むには入力ストリーム (**InputStream**)、書き出すには出力ストリーム (**OutputStream**) を使います。ストリームはバイトデータ (バイナリデータ) の流れですが、実際には文字列として処理することが多いため、文字列処理用に **Reader** と **Writer** が準備されています (前回)。今回は InputStream と OutputStream に関する処理を作成します。

2. 入出力のデコレータ

InputStream や OutputStream、Reader や Writer は必要最小限の機能しかありません。そこでサブクラスとしてファイルを扱うためのクラスやバッファリングを行うクラスなどが準備されており、これらを連結して使うことができます。もとのクラスに機能を付加するので (装飾するので) これらのクラスを **デコレータ** と呼びます。

データの種類	入出力	クラス	ファイルを扱うクラス	バッファリングを提供するクラス
バイナリ	入力	InputStream	FileInputStream	BufferedInputStream
	出力	OutputStream	FileOutputStream	BufferedOutputStream
文字列	入力	Reader	FileReader (前回)	BufferedReader (前回)
	出力	Writer	FileWriter (前回)	BufferedWriter (前回)

3. FileOutputStream

FileOutputStream でファイルにデータを書き出すには以下のようにします。

```
OutputStream out = new FileOutputStream(ファイル名);
out.write(バイト配列);                                // 配列の全データを書き込む
out.close();
```

● J2Kad18C 「ファイルからの読み込み (InputStream)」

1. FileInputStream

`FileInputStream` でデータを読み込むには以下のようにします。

```
InputStream in = new FileInputStream(ファイル名);
int len;
byte[] b = new byte[1024];
while ((len = in.read(b)) != -1) {
    // 読み込んだときの処理
}
in.close();
```

`read` メソッドでバイト型配列にデータを読み込みます。戻り値は読み込んだバイト数です。データがないときは-1を返します。

● J2Kad18B 「ファイルコピー」

1. ファイルコピー

ファイル入力はJ2Kad18C、ファイル出力はJ2Kad18Dの復習です。ただしJ2Kad18Dではバイト型配列の全データを出力していましたが、ここではバイト型配列に読み込んだデータを出力するので（配列の最後までデータを読み込んでいるとは限らないので）、以下のメソッドを使います。


```
out.write(バイト配列, 先頭インデックス, バイト数);
```

2. クローズ処理

ファイルコピーに関するすべての処理（オープンからクローズまで）を `try` ブロックに入れてしまうと、コピー中に何らかの例外が発生するとクローズ処理が行われなくなります。ここでは `finally` を追加してクローズ処理を行います。なお `InputStream` と `OutputStream` には初期値 `null` を設定しておき、ファイルがオープンされているかどうかを判別するようにします。


Before

```
try {
    InputStream in = FileInputStream(...);
    OutputStream out = FileOutputStream(...);
    :
    in.close();
    out.close();
} catch(IOException e) {
    System.out.println(e);
}
```



After

```
InputStream in = null;
OutputStream out = null;
try {
    in = FileInputStream(...);
    out = FileOutputStream(...);
    :
} catch(IOException e) {
    System.out.println(e);
} finally {
    // クローズ処理
}
```



● J2Kad18A 「Web ページのコピー」

1. Web ページのコピー

Web ページから `InputStream` を取得します。あとは（対象がネット、かつ `test.bin` に比べて容量も大きいので）バッファリングを追加して `InputStream` から `OutputStream` へコピーします。コピーは J2Kad18B と同じです。

```
URL url = new URL(URL の文字列);
in = new BufferedInputStream(url.openStream());
out = new BufferedOutputStream(new FileOutputStream(ファイル名));
```

● J2Kad18S 「DataInputStream の連結」

1. DataInputStream

J2Kad18X のファイルダンプ作成のための準備です。ここでは `DataInputStream` を使ってバイト単位でデータを読み込んで表示します。ただし `DataInputStream` の `readByte` メソッドは読み込んだバイトデータが戻り値なので、データ終了かどうかを戻り値で判別することができません。そこでデータの最後まで到達すると `EOFException` をスローします。

● J2Kad18X1 「ファイルダンプ①」

1. ファイルダンプ

J2Kad18S を改造してファイルダンプを作ります。J2Kad18S にてデータの読み込みはできているので、ここでは表示の体裁を整える作業になります。方法はいろいろありますが、`System.out.printf` を使うのが最も手っ取り早いです。（検索して調べてみましょう）ちなみに `nJava` の `class` ファイルは先頭に「CA」「FE」「BA」「BE」（Cafe Babe）が入っています。

● J2Kad18X2 「ファイルダンプ②」

1. ASCII 文字列の表示

J2Kad18X1 のダンプ表示の右端に表示するデータを ASCII 文字列として表示する処理を追加します。ただしバイトデータを文字コードとしてそのまま `print` するとコントロールコードも入っているので（画面表示が崩れるので）、英数字に該当する箇所以外は「.」（ピリオド）として表示します。ファイルの最後はバイトデータが行の途中で終わる可能性が高いですが、このときも途中までの文字列を右端に表示させるようにしてください。

なお、ASCII 文字列を表示させると、いろいろな単語が含まれているのがわかります。これを読み解くことで `class` ファイルにどんな情報が含まれているのか、おぼろげに見えてきます。