● J2Kad18D「ファイルへの書き出し（OutputStream）」

OutputStream を使ってファイル「test.bin」へ以下のデータを書き出す処理を作成せよ。

・ファイル名：　　test.bin
・出力するデータ：　byte[] b = {69, 67, 67, 32, 67, 79, 77, 80};　　// ECC COMP

OutputStream による書き出し

```
OutputStream out = new FileOutputStream(ファイル名);
out.write(出力するデータ);                          // 配列名で指定
out.close();
```

課題完成時の画面

```
test.bin にデータを出力しました！
```

IntelliJ で test.bin を開くと（バイナリーエディターなどのプラグインが入っていない場合は）「ECC COMP」と表示される。

● J2Kad18C「ファイルからの読み込み（InputStream）」

InputStream を使って「test.bin」からデータを読み込んで表示する処理を作成せよ。

InputStream によるの読み込み

```
InputStream in = new FileInputStream(ファイル名);
int len;                                    // 読み込んだデータ数
byte[] b = new byte[1024];                  // データ読み込み用配列
while ((len = in.read(b)) != -1) {          // データ読み込み（データがないときは-1）
    // 読み込んだデータに対する処理          // ここでｈ配列ｂの各要素を表示する
}
in.close();
```

課題完成時の画面

```
69 67 67 32 67 79 77 80
```

● J2Kad18B「ファイルコピー」

以下の仕様でプログラムを作成せよ。

① ファイル「test.bin」（J2Kad18D で作成）をファイル「test2.bin」へコピーする処理を作成せよ。
② ①の処理実行中に例外が発生してもファイルがオープンされている場合は必ずクローズするように対応せよ。

①のヒント：配列 b のインデックス 0 から len 個のデータを書き込み場合は以下のようにする。

```
out.write(b, 0, len);     // 配列 b のインデックス 0 から len 個のデータを書き込む
```

②のヒント：①の処理実行中に例外が発生してもクローズするためには以下のようにすれば可能（もちろん他の方法もあり）。

```
try {
    // ファイルコピーの処理
} catch(IOException e) {
    System.out.println(e);
} finally {
    // クローズ処理
}
```

課題完成時の画面

```
ファイルコピー完了しました！
```

test2.bin を開いて test.bin と同じ内容が書き込まれているか確認すること。

● J2Kad18A「Web ページのコピー」

ECC コンピュータ専門学校のホームページ（https://comp.ecc.ac.jp/）よりソースコード（HTML）を取得し、ファイル「ecc.html」へ保存せよ。なお、ファイル処理を効率よく行うために入力ストリームには BufferedInputStream、出力ストリームには BufferedOutputStream を接続すること。

URL からのデータを入力ストリームとして設定

```
URL url = new URL("https://comp.ecc.ac.jp/");
InputStream in = url.openStream();
```

URL からのデータを（バッファ経由で）入力ストリームとして設定

```
URL url = new URL("https://comp.ecc.ac.jp/");
InputStream in = new BufferedInputStream(url.openStream());
```

課題完成時の画面

```
HTML を取得しました！
```

取得したコード（ファイル「ecc.html」）

```
<!DOCTYPE html>
<html lang="ja">

<head prefix="og: http://ogp.me/ns# fb: http://ogp.me/ns/fb# website: http://ogp.me/ns/website#">
  <!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||[];w[l].push({'gtm.start':
  new Date().getTime(),event:'gtm.js'});var f=d.getElementsByTagName(s)[0],
  j=d.createElement(s),dl=l!='dataLayer'?'&l='+l:'';j.asyn
  'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.pare
  })(window,document,'script','dataLayer','GTM-PJR44SZ');</
  <!-- End Google Tag Manager -->
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,user-sc
  <title>ゲーム専門学校 大阪｜ECC コンピュータ専門学校_IT,W
  <link rel="canonical" href="https://comp.ecc.ac.jp/" />
    ⋮
```

ブラウザで開くと HTML 部分のみ表示（ローカルなので css や script は読み込まれない）

ECCコンピュータ専門学校
ECC COLLEGE OF COMPUTER AND MULTIMEDIA

- 学校紹介
  - 学校紹介TOP
  - 理念
  - 学校概要・沿革
  - 情報の公表
  - 第三者評価
  - 施設・設備紹介
  - 交通アクセス
  - HAND BOOK 2022
  - 関連リンク
  - 職業実践専門課程設置校とは？

● J2Kad18S「DataInputStream の連結」

DataInputStream (データ用の入力ストリーム) を使ってファイルからデータを読み込んで表示する処理を作成せよ。

・ファイル名： ./out/production/J2Kad18/J2Kad18D.class ※他の課題の class ファイルでも OK
・データの流れ： ファイル → FileInputStream → BufferedInputStream → DataInputStream → 表示

DataInputStream の仕様について

・readByte メソッドで 1 バイトずつデータを取得
・データの最後に到達すると例外「EOFException」が発生する。

課題完成時の画面

```
-54 -2 -70 -66 0 0 0 62 0 66 10 0 2 0 3 7
0 4 12 0 5 0 6 1 0 16 106 97 118 97 47 108
97 110 103 47 79 98 106 101 99 116 1 0 6 60 105 110
    ⋮
```

DataInputStream の readByte メソッドで 1 バイトずつ読み込んで表示する。16 バイト分表示したら改行する。

● J2Kad18X1「ファイルダンプ①」※J2Kad18S の main メソッドをコピーして改造

課題完成時の画面を参考にファイルダンプを行う処理を作成せよ（J2Kad18S の表示を 16 進数表示にする）。仕様は以下の通り。

・1 行あたり 16 バイト分のデータを表示する。表示形式は 16 進数 2 桁。
・各データの間は半角スペースを入れる。ただし前半 8 バイトと後半 8 バイトの間は「-」を入れる。
・各行の先頭には表示したバイト数を 16 進数 8 桁で表示する。

課題完成時の画面

```
00000000：CA FE BA BE 00 00 00 3E-00 42 0A 00 02 00 03 07
00000010：00 04 0C 00 05 00 06 01-00 10 6A 61 76 61 2F 6C
00000020：61 6E 67 2F 4F 62 6A 65-63 74 01 00 06 3C 69 6E
00000030：69 74 3E 01 00 03 28 29-56 08 00 08 01 00 08 74
00000040：65 73 74 2E 62 69 6E 07-00 0A 01 00 18 6A 61 76
00000050：61 2F 69 6F 2F 46 69 6C-65 4F 75 74 70 75 74 53
00000060：74 72 65 61 6D 0A 00 09-00 0C 0C 00 05 00 0D 01
00000070：00 15 28 4C 6A 61 76 61-2F 6C 61 6E 67 2F 53 74
00000080：72 69 6E 67 3B 29 56 0A-00 0F 00 10 07 00 11 0C
      ⋮
000003F0：36 00 00 00 03 00 56 00-37 00 38 00 01 00 32 00
00000400：27 00 39 00 3A 00 02 00-3B 00 00 00 16 00 02 FF
00000410：00 50 00 03 07 00 3C 07-00 3D 07 00 3F 00 01 07
00000420：00 24 07 00 01 00 40 00-00 00 02 00 41
```

バイト数（16 進数 8 桁）　　8 バイト表示した後は「-」を入れる

ヒント：
- 1 バイトデータ（byte）を 16 進数 2 桁の文字列に変換するメソッドを自作する。
- 4 バイトデータ（int）を 16 進数 8 桁の文字列に反感するメソッドを自作する。
- もしくは System.out.printf を検索して使う。

Java の class ファイルでは先頭の 4 バイトに「CA」「FE」「BA」「BE」（Cafe Babe）が入っている。

● J2Kad18X2「ファイルダンプ②」※J2Kad18X1 の main メソッドをコピーして改造

J2Kad18X1 のダンプ表示の右端に表示したバイトデータを文字として表示する処理を追加せよ。バイトデータの変換の仕様は以下の通り。

・バイトデータの値が 0x20～0x7E のとき（英数字のとき）、そのままの値を文字として表示。
・上記以外のとき、「.」（ピリオド）として表示

課題完成時の画面

```
00000000：CA FE BA BE 00 00 00 3E-00 42 0A 00 02 00 03 07  .......>.B......
00000010：00 04 0C 00 05 00 06 01-00 10 6A 61 76 61 2F 6C  ..........java/l
00000020：61 6E 67 2F 4F 62 6A 65-63 74 01 00 06 3C 69 6E  ang/Object...<in
00000030：69 74 3E 01 00 03 28 29-56 08 00 08 01 00 08 74  it>...()V......t
00000040：65 73 74 2E 62 69 6E 07-00 0A 01 00 18 6A 61 76  est.bin......jav
00000050：61 2F 69 6F 2F 46 69 6C-65 4F 75 74 70 75 74 53  a/io/FileOutputS
00000060：74 72 65 61 6D 0A 00 09-00 0C 0C 00 05 00 0D 01  tream...........
00000070：00 15 28 4C 6A 61 76 61-2F 6C 61 6E 67 2F 53 74  ..(Ljava/lang/St
00000080：72 69 6E 67 3B 29 56 0A-00 0F 00 10 07 00 11 0C  ring;)V.........
      ⋮
000003F0：36 00 00 00 03 00 56 00-37 00 38 00 01 00 32 00  6.....V.7.8...2.
00000400：27 00 39 00 3A 00 02 00-3B 00 00 00 16 00 02 FF  '.9.:...;.......
00000410：00 50 00 03 07 00 3C 07-00 3D 07 00 3F 00 01 07  .P....<..=..?...
00000420：00 24 07 00 01 00 40 00-00 00 02 00 41           .$....@.....A
```

バイトデータ表示のあとに各数値を文字として表示する処理を追加。
なお、最終行のようにバイトデータの表示が途中で終わっている場合も文字表示を行うこと。