# **J2Kad17D「出力ストリーム（OutputStream）」**

OutputStreamを使ってファイル「test.bin」へ以下のデータを出力する処理を作成せよ。

・出力するデータ： byte[] b = {69, 67, 67, 32, 67, 79, 77, 80}; // ECC COMP

**OutputStreamによる出力**

OutputStream out = new FileOutputStream(ファイル名);

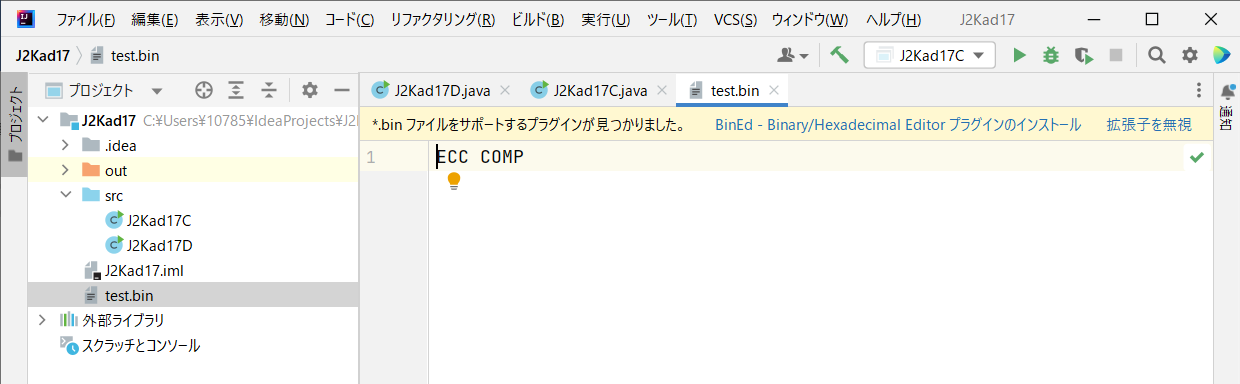
out.write(出力するデータ); // 配列名で指定

out.close();

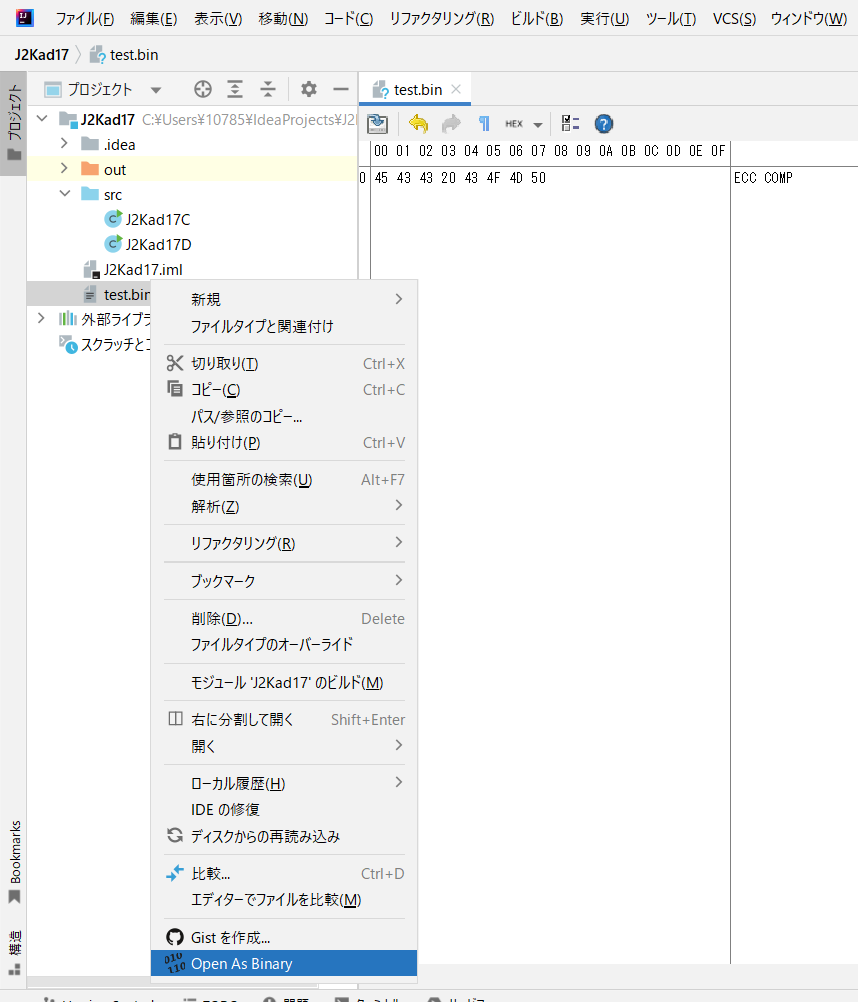
**課題完成時の画面**

test.binにデータを出力しました！

**InteliJで「test.bin」を開くと「ECC COMP」と文字列で表示される**

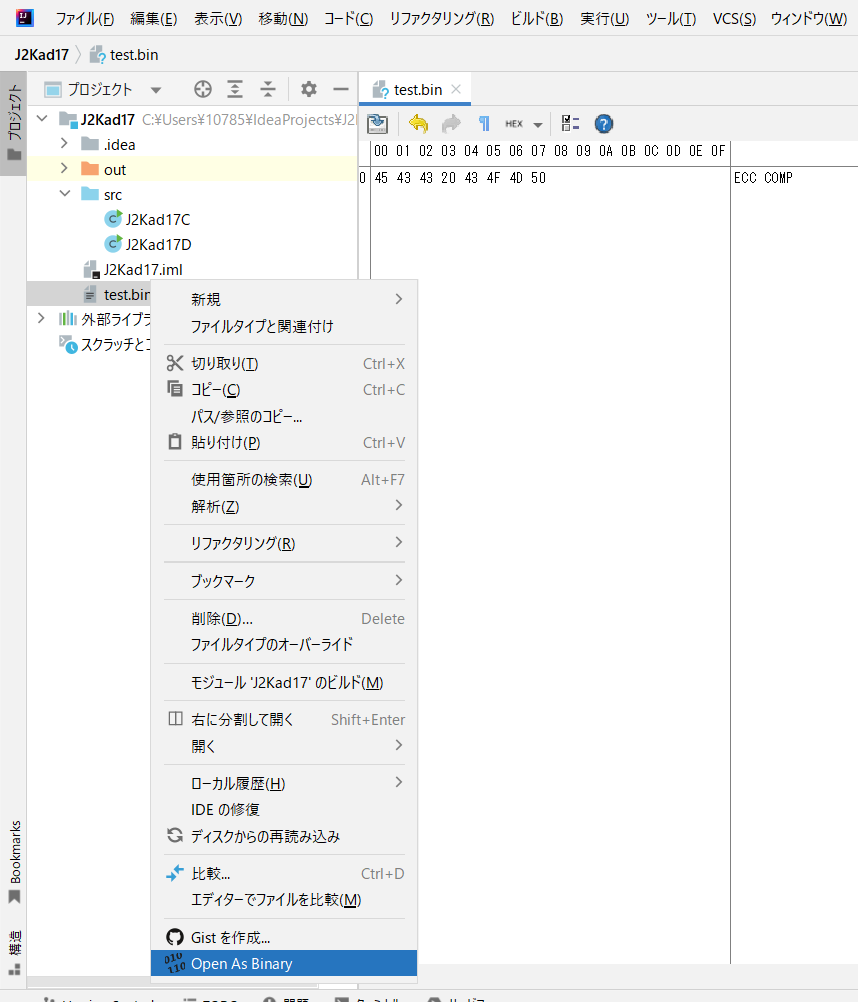
****

****

****

バイナリーエディターのプラグインをインストールして

「test.bin」を右クリック

****

バイナリーデータとして表示される（表示は16進数）

# **J2Kad17C「入力ストリーム（InputStream）」**

InputStreamを使って「test.bin」から入力したデータを表示する処理を作成せよ。

**InputStreamによる入力**

InputStream in = new FileInputStream(ファイル名);

int len; // 入力したデータ数

byte[] b = new byte[1024]; // データ入力用配列

while ((len = in.read(b)) != -1) { // データ入力（データがないときは-1）

// 入力した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

# **J2Kad17B「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」（J2Kad17D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個のデータを出力した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個のデータを出力

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**

test2.binを開いてtest.binと同じ内容が出力されているか確認すること。

ファイルコピー完了しました！

# **J2Kad17A「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」）※更新されていたら内容が異なることがあります。**

<!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.async=true;j.src=

'https://www.googletagmanager.com/gtm.js?id='+i+dl;f.parentNode.insertBefore(j,f);

})(window,document,'script','dataLayer','GTM-PJR44SZ');</script>

<!-- End Google Tag Manager -->

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width,user-scalable=1,maximum-scale=1" />

<title>ゲーム専門学校 大阪｜ECCコンピュータ専門学校\_IT,WEBデザイン,プログラミング</title>

<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）

# **J2Kad17S「DataInputStreamの連結」※J2Kad17Xの準備**

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/J2Kad187J2Kad17D.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**

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。

-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

…

# **J2Kad17X1「ファイルダンプ①」※J2Kad17Sをコピーして作成**

**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（J2Kad17S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

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桁の文字列に変換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。

8バイト表示した後は「-」を入れる

バイト数（16進数8桁）

# **J2Kad17X2「ファイルダンプ②」※J2Kad17X1をコピーして作成**

J2Kad17X1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。