# **J2Kad16D「ファイル出力のきほん（FileWriter）」 （実践編P.171「Fileクラス」、P.153「ファイルへの出力」）**

FileWriterを使ってファイル「test.txt」へ文字列「ECCCOMP」を出力する処理を作成せよ。なお、「test.txt」が存在する場合は削除してから（手作業でOK）実行すること。

**ファイルの新規作成**

File file = new File(ファイル名);

if (file.createNewFile()) {

// 新規作成できたときの処理

}

**ファイルへの出力**

ファイル

FileWriter

プログラム

FileWriter fw = new FileWriter(file):

fw.write(出力する文字列);

fw.close();

**課題完成時の画面**

実際に文字列が出力されているかどうかはtest.txtを開いて

確認すること（IntelliJのプロジェクトウインドウから開く

ことも可能）。

test.txtを作成しました！

test.txtにECCCOMPを出力しました！

# **J2Kad16C「ファイル入力のきほん（FileReader）」（実践編P.160「ファイルからの入力」）**

FileReaderを使って「test.txt」へ出力した文字列を入力して表示する処理を作成せよ。

**ファイルからの入力**

ファイル

FileReader

プログラム

FileReader　fr = new FileReader(file);

int data;

while((data = fr.read()) != -1) System.out.println(data); // 読み込んだ値（data）をそのまま表示

fr.close();

**課題完成時の画面**

test.txtから文字列を読み出します！

Eを読み出しました！

Cを読み出しました！

Cを読み出しました！

Cを読み出しました！

Oを読み出しました！

Mを読み出しました！

Pを読み出しました！

入力は1文字ずつ行われる。

なお、出力したデータをそのまま表示すると

文字ではなく数値（文字コード）が表示される。

# **J2Kad16B「ストリームの連結（BufferedWriter、BufferedReader）」 （実践編P.156「バッファ」、P.158「ストリームの連結」）**

|  |
| --- |
| **Monster** |
| name |
| Monster()  getName() : String  toString() : String |

課題フォルダにMonsterクラスが準備されている。以下の仕様でファイル入出力処理を作成せよ。なおFileWriterにはBufferedWriterとPrintWriterを連結し、FileReaderにはBufferedReaderを連結すること。

**ファイル出力**

ファイル

FileWriter

プログラム

BufferedWriter

PrintWriter

**ファイル入力**

ファイル

FileReader

プログラム

BufferedReader

**mainメソッドの仕様**

1. ファイル「test.txt」がすでに存在しているときは削除する。
2. ファイル「test.txt」を新規作成し、以下の処理を実行する。
3. Monsterクラスを5体生成し、各モンスターの名前をファイルへ書き込む。なお、名前の後は改行すること。
4. ファイルから文字列を読み出し表示する（モンスターの名前が5行分表示される）。

**課題完成時の画面**

扱うファイルはJ2Kad16Cと同じ「test.txt」。

J2Kad16B完成後に再度J2Kad16Cを実行してファイルの内容を表示してみるとFileReaderのみの読み込みとBufferedReaderを使った読み込みの違いがわかる。

test.txtを削除しました！

test.txtを作成しました！

モンスター一覧を作成します！

トランセルを追加した！

ムックルを追加した！

ベトベトンを追加した！

モクローを追加した！

イワンコを追加した！

モンスター一覧を表示します！

トランセルを読み出した！

ムックルを読み出した！

ベトベトンを読み出した！

モクローを読み出した！

イワンコを読み出した！

# **J2Kad16A「ドットパターンの表示」（フォルダの操作は実践編P.173「フォルダの操作」）**

課題フォルダの中にあるdataフォルダにA～Zおよび0～9の英数字のドットパターンが入ったファイルが準備されている（8×8ドット、ファイル名はA.txt、B.txt、…）。dataフォルダ内にあるファイル一覧を表示し、指定した英数字（A～Z、0～9）のドットパターンを画面に表示する処理を作成せよ。

**課題完成時の画面**

0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt 9.txt

A.txt B.txt C.txt D.txt E.txt F.txt G.txt H.txt I.txt J.txt

K.txt L.txt M.txt N.txt O.txt P.txt Q.txt R.txt S.txt T.txt

U.txt V.txt W.txt X.txt Y.txt Z.txt

どのドットパターンを見ますか？（拡張子不要、-1：終了）＞**0**

---\*\*\*--

--\*---\*-

--\*---\*-

--\*---\*-

--\*---\*-

--\*---\*-

---\*\*\*--

--------

0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt 9.txt

A.txt B.txt C.txt D.txt E.txt F.txt G.txt H.txt I.txt J.txt

K.txt L.txt M.txt N.txt O.txt P.txt Q.txt R.txt S.txt T.txt

U.txt V.txt W.txt X.txt Y.txt Z.txt

どのドットパターンを見ますか？（拡張子不要、-1：終了）＞**a**

---\*\*---

--\*--\*--

-\*----\*-

-\*----\*-

-\*\*\*\*\*\*-

-\*----\*-

-\*----\*-

--------

0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt 9.txt

A.txt B.txt C.txt D.txt E.txt F.txt G.txt H.txt I.txt J.txt

K.txt L.txt M.txt N.txt O.txt P.txt Q.txt R.txt S.txt T.txt

U.txt V.txt W.txt X.txt Y.txt Z.txt

どのドットパターンを見ますか？（拡張子不要、-1：終了）＞**ecc**

そのデータは存在しません！

0.txt 1.txt 2.txt 3.txt 4.txt 5.txt 6.txt 7.txt 8.txt 9.txt

A.txt B.txt C.txt D.txt E.txt F.txt G.txt H.txt I.txt J.txt

K.txt L.txt M.txt N.txt O.txt P.txt Q.txt R.txt S.txt T.txt

U.txt V.txt W.txt X.txt Y.txt Z.txt

どのドットパターンを見ますか？（拡張子不要、-1：終了）＞**-1**

ファイル名の間にはタブを入れている。

ファイル一覧は10ファイル単位で改行。

アルファベットは大文字

でも小文字でもOK

ファイルが存在しないときは「そのデータは存在しません！」と表示

-1で終了

# **J2Kad16S1「文字列描画システム①（BigCharクラス）」**

8×8ドットの英数字（J2Kad16Aで確認したドットパターン）を表すBigCharクラスを定義し、指定された英数字（1文字）をCanvasクラスへ描画して表示する処理を作成せよ。

**BigCharクラスの仕様（ファイル「BigChar.javaに作成」）**

|  |  |
| --- | --- |
| メンバ | 仕様 |
| private boolean[][] fontdata | ドットデータ用2次元配列（8×8） |
| public BigChar(char charname) | 2次元配列fontdata（8×8）を生成し、charnameで指定された英数字のドットパターンを  読み込む（ドットデータが「\*」のときはtrue、それ以外のときはflaseを設定する）。  読み込んだのち「～を読み込みました！」と表示する（～は大文字、変換は以下の通り）。  **String letter = String.valueOf(charname).toUpperCase(); // char型→大文字** |
| public void draw(Canvas c) | Canvasクラスにfontdataを描画する。描画方法は以下の通り。  **c.drawFont(fontdata); // Canvasへfontdataを描画** |

**mainメソッドの仕様**

1. 表示する英数字を入力する（-1で終了）。
2. 入力された英数字に該当するBigCharを生成し、Canvasクラスへ描画して表示する。描画と表示は以下の通り（c：Canvasクラス、bc：BigCharクラス）。  
     
    c.clear(); // Canvasのクリア  
    bc.draw(c); // BigCharの描画  
    c.show(); // Canvasの表示
3. ①へ戻る。

**課題完成時の画面**

文字（英数字）を入力してください（-1：終了）＞**e**

Eを読み込みました！

・○○○○○○○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○○○○○○○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○○○○○○○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

文字（英数字）を入力してください（-1：終了）＞**-1**

# **J2Kad16S2「文字列描画システム②（BigStringクラス）」**

BigCharクラスで構成される文字列を表すBigStringクラスを定義し、指定された英単語（文字列）をCanvasクラスへ描画して表示する処理を作成せよ。

**BigStringクラスの仕様（ファイル「BigString.javaに作成」）**

|  |  |
| --- | --- |
| メンバ | 仕様 |
| BigChar[] bigChars | 文字列を構成するBigCharの配列。 |
| public BigString(String str) | 文字列strの文字数分の配列bigCharsを生成し、各文字に該当するBigCharを生成する。 |
| public void draw(Canvas c) | 配列bigCharsのBigCharを描画する。 |

**課題完成時の画面**

文字列（英数字）を入力してください＞**ECCCOMP**

Eを読み込みました！

Cを読み込みました！

Cを読み込みました！

Cを読み込みました！

Oを読み込みました！

Mを読み込みました！

Pを読み込みました！

・○○○○○○○・・・○○○○・・・・○○○○・・・・○○○○・・・○○○○○・

・○・・・・・・・・○・・・・○・・○・・・・○・・○・・・・○・○・・・・・○

・○・・・・・・・○・・・・・・・○・・・・・・・○・・・・・・・○・・・・・○

・○○○○○○○・○・・・・・・・○・・・・・・・○・・・・・・・○・・・・・○

・○・・・・・・・○・・・・・・・○・・・・・・・○・・・・・・・○・・・・・○

・○・・・・・・・・○・・・・○・・○・・・・○・・○・・・・○・○・・・・・○

・○○○○○○○・・・○○○○・・・・○○○○・・・・○○○○・・・○○○○○・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・○・○○○○○○・・・・・・・・・・・・・・・・・・・・・・・・・

・○○・・・○○・○・・・・・○・・・・・・・・・・・・・・・・・・・・・・・・

・○・○・○・○・○・・・・・○・・・・・・・・・・・・・・・・・・・・・・・・

・○・・○・・○・○○○○○○・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・○・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・○・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・○・・・・・○・○・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・・

文字列（英数字）を入力してください＞**-1**

# **J2Kad16X1「文字列描画システム③（Flyweightパターン）」※J2Kad16S2をコピーして作成**

J2Kad16S2では文字列の中に同じ文字があっても何度も読み込みを行うようになっている（例えば「ECCCOMP」では「C」を3回読み込んでいる）。Flyweightパターン（←検索）を使って同じ文字は読み込まないようにせよ。

ヒント：

・Before ：BigStringでBigCharを生成

・After ：BigCharをプールするBigCharFactoryを作る。BigStringはBigCharFactoryからBigCharの参照をもらう。

**BigCharManagerクラスの仕様（ファイル「BigCharManager.javaに作成」）**

|  |  |
| --- | --- |
| メンバ | 仕様 |
| private HashMap<String, BigChar> pool | BigCharをプールする連想配列（初期化しておくこと）。 |
| public BigChar createBigChar(char charname) | charname（大文字の文字列に変換）がpoolに登録されているときは  該当するBigCharの参照を返す。まだpoolに登録されていないときは  該当するBigCharを生成しpoolに登録、そのBigCharの参照を返す。 |

**課題完成時の画面**

「ECCCOMP」に「C」は3回出てくるが、

読み込みは1回のみ。

文字列（英数字）を入力してください＞**ECCCOMP**

Eを読み込みました！

Cを読み込みました！

Oを読み込みました！

Mを読み込みました！

Pを読み込みました！

# **J2Kad16X2「文字列描画システム④（Singletonパターン）」※J2Kad16X1をコピーして作成**

J2Kad16X2では「ECCCOMP」（1回目）と入力したあとに「ABCDE」と入力すると「A」「B」「D」だけでなく再び「C」「E」が読み込まれてしまう。これまでに読み込んだ文字がある場合も再読み込みしないように修正せよ。

ヒント：BigCharManagerをSingleton（←検索）にする。

**J2Kad16X1の画面（ECCCOMPのあとにABCDE） 課題完成時の画面（ECCCOMPのあとにABCDE）**

文字列（英数字）を入力してください＞**ECCCOMP**

…

文字列（英数字）を入力してください＞**ABCDE**

Aを読み込みました！

Bを読み込みました！

Dを読み込みました！

…

文字列（英数字）を入力してください＞**ECCCOMP**

…

文字列（英数字）を入力してください＞**ABCDE**

Aを読み込みました！

Bを読み込みました！

Cを読み込みました！

Dを読み込みました！

Eを読み込みました！

…

「C」「E」は再読み込みしない。

「A」「B」「D」だけでなく「C」「E」も読み込む。