The Java 使いこなす

／

Javaでゲームを作ろう○

－ 共通編 －

Java8 ／ Java11 ／ Java25 以降対応

2026/01/04 修正

The Java 使いこなす／ Javaでゲームを作ろう○ － 共通編 －

０ はじめに

この本を読んでいただき、ありがとうございます。

この本は、「The Java 使いこなす」 および 「Javaでゲームを作ろう」の共通的な部分をおさめた本となります。「Javaでゲームを作ろう」の第３弾から、本に準備作業を載せるのをやめ、サイトに載せます。本当は本に入れたいと思いましたが、このページだけで、なん10ページとかかり（値段にはね返る）、それも毎回同じことを載せることになるので、いっそのこと、本の中から省いてみました。

準備作業も初めての人にとっては、難しいかもしれません。
でも、ここであきらめずに、乗り越えていこう♪

乗り越えるところはまだまだある。こんなところであきらめてはいけません。
がんばろう！

１　初級編　動かせる環境を作ろう

１－２　動かせる環境を作っていこう

１－１「Java のインストールについて」に続いて、次に、ゲームが動いて、作っていける環境を準備していこう。

Java／ゲームが動いて作っていける環境…つまり、Java、それも JDK が動くような状態にしていく。

（通常、インストールしたままの状態では、JRE が動くような状態になっている。そのままでは、Java の実行はできても、開発はできません）

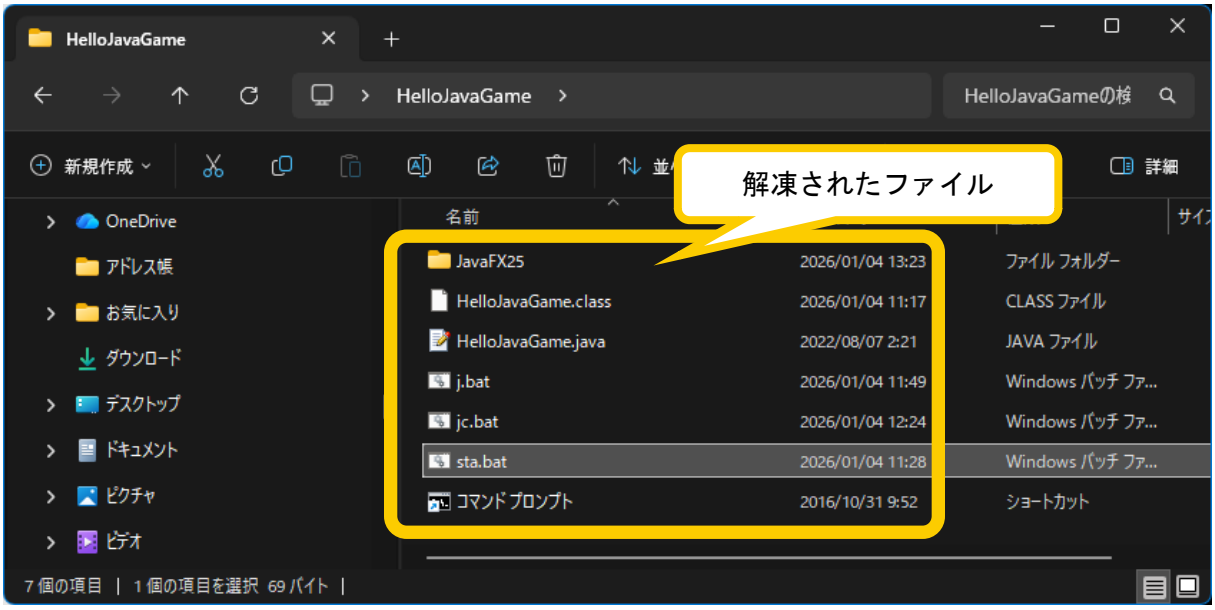
まずはサンプルプログラムをダウンロードしよう。

http://kinchannn.jp/javagame_common/　から HelloJavaGame.zip へのリンクを探してダウンロードしよう。

大文字／小文字を間違えないように入力してほしい。間違えると、ダウンロードできない。

ダウンロードしたら、zip ファイルを解凍しよう。
（最近の OS であれば、ファイルを右クリックすれば、解凍できる）

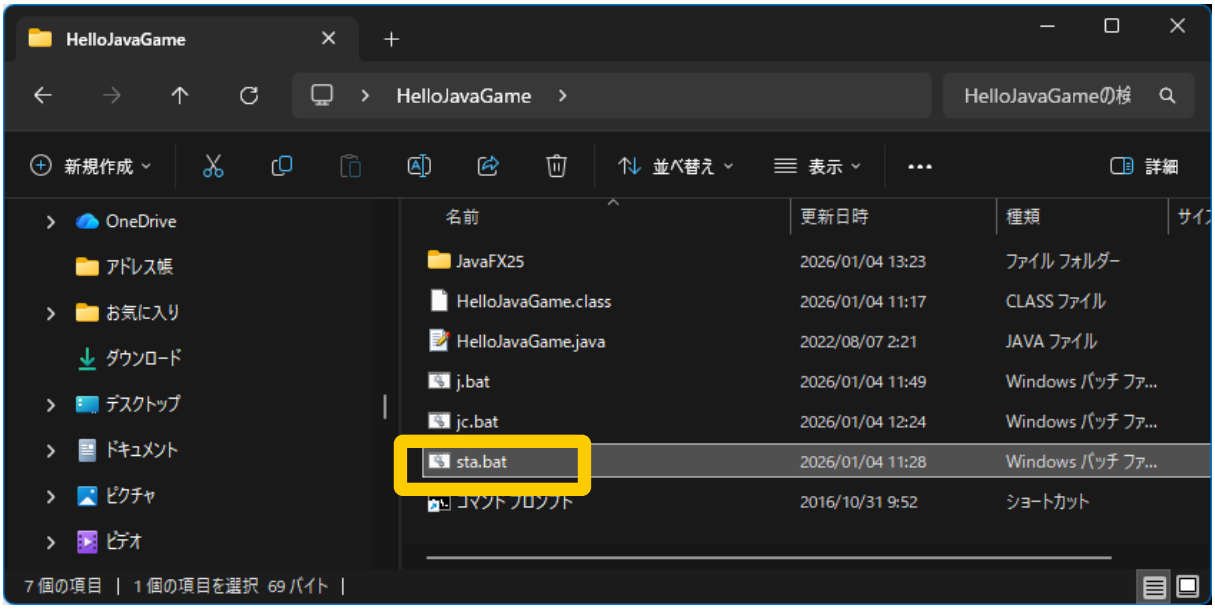
解凍すると下のような感じで解凍されるはずだ。



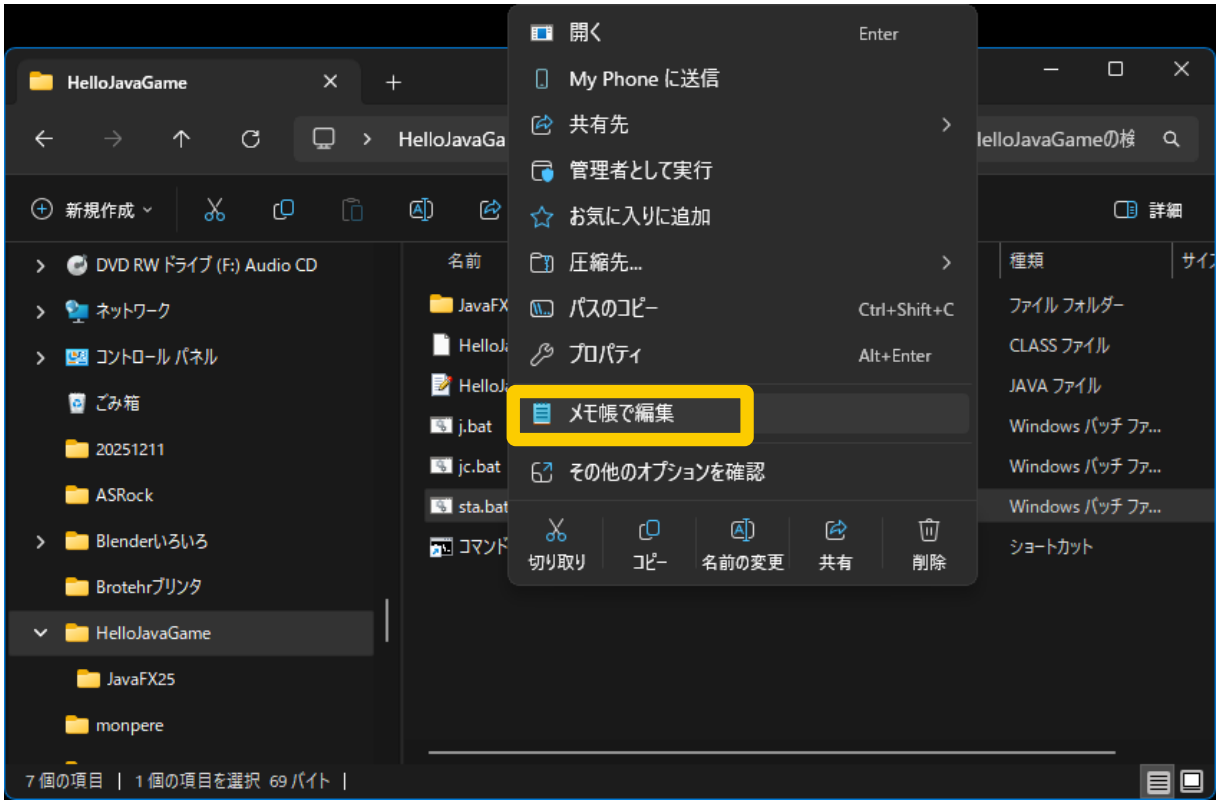
プログラムを組んだ後、実行するためには、コンパイルをしなくてはならない。
コンパイルとは、プログラムを実行できるように変換することである。
コンパイルをすると、「*.class」というファイルが作成される。

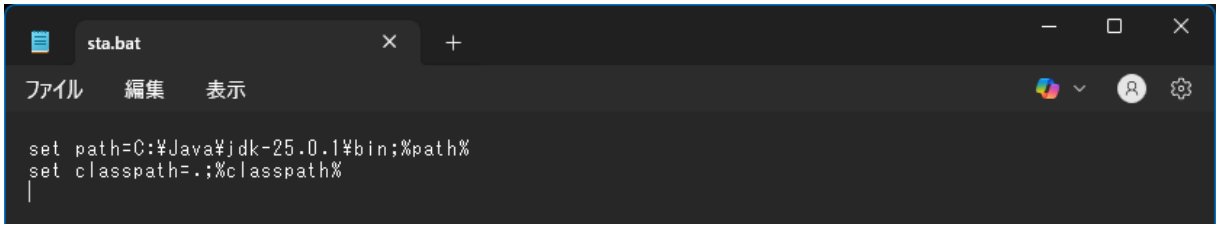
パスを変更しよう（Java8。Java11 以降は、先のページで紹介します）

コンパイルするために、パスの変更を行う。
HelloJavaGame フォルダの中に、「sta.bat」というのがある。



右クリックし、メモ帳で編集をクリックしよう。
（Windows10 の場合は「編集」かもしれません）
すると、メモ帳（などテキストエディタ）で開くはずだ。





内容は、２行書かれている。
これから編集をしていきますが、編集を間違えると、動かなくなってしまう。はまる可能性もあるので、気をつけて設定いこう。

まずは、内容について、説明する。

１行目から説明していこう

```
set path=C:¥Java¥jdk-25.0.1¥bin;%path%
```

まずは、いくつかの部分に分けて考えていくことができる。

「set path=」
set とはその通り、セットすることである。
環境変数と呼ばれる、プログラムの動作に必要な情報を保存する仕組みである。
ひのこのコマンドプロンプト内で有効となる設定となる。

path は何なのか？
path は、なにかを実行する時に（今回の場合は、Java のコマンドを実行する際に）、その右側に書かれているフォルダを見るように設定する。
見えるようにしないと、見てくれない。

「C:¥Java¥jdk-25.0.1¥bin」
この設定が、Java のコマンドが置いてある場所となる。
君が Java を置いた場所に編集しなおさなくてはいけない。
間違うと、Java のコマンドを実行しようとしても、動かないので気をつけよう。

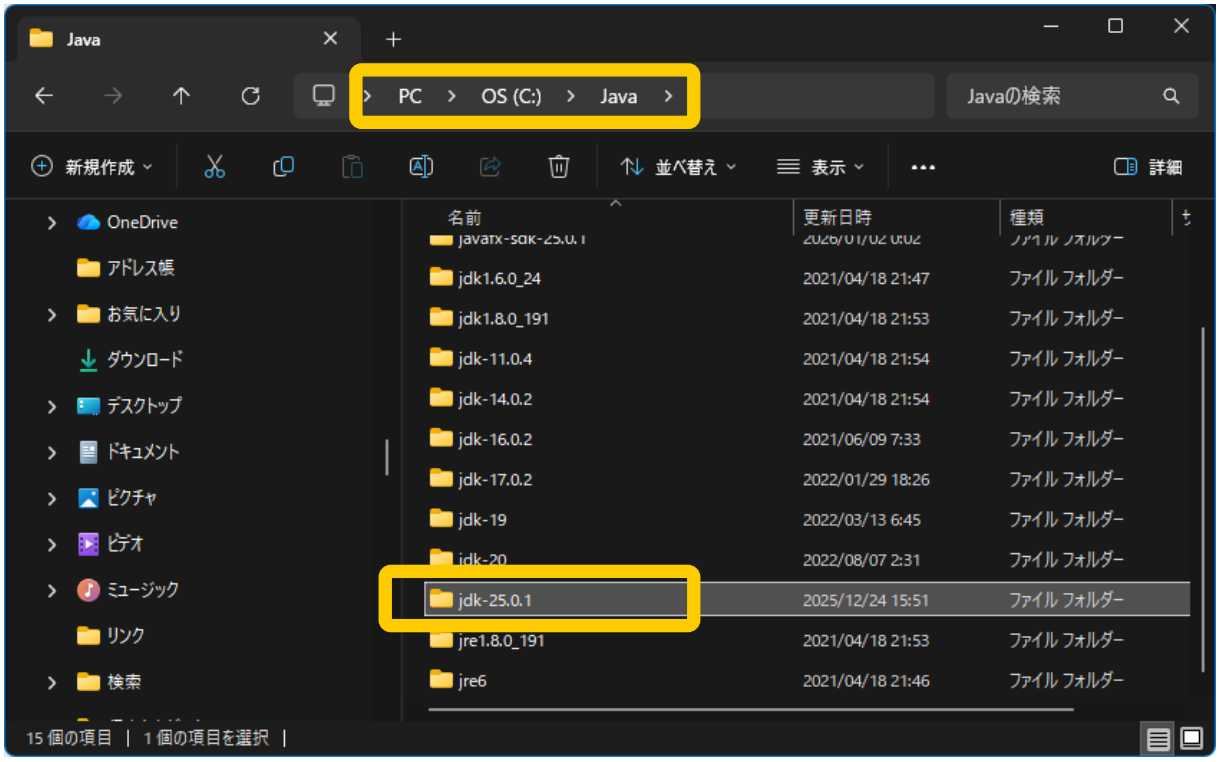
「;」
区切りとなる。複数のパス（場所）を設定できるということだ。

「%path%」
もともとのパスの設定を示す。この設定をする前に「path」という設定は、既にされている場合がある。
そのもともとの設定と合わせて新しい設定：path を作った（作り変えた）ということだ。
もともとのパスの前に、Java 用のパスを追加した格好となる。

先ほども書いたが、どこかひとつでも設定を間違えると、動かなくなってしまう可能性がある。
気をつけて設定をしてほしい。

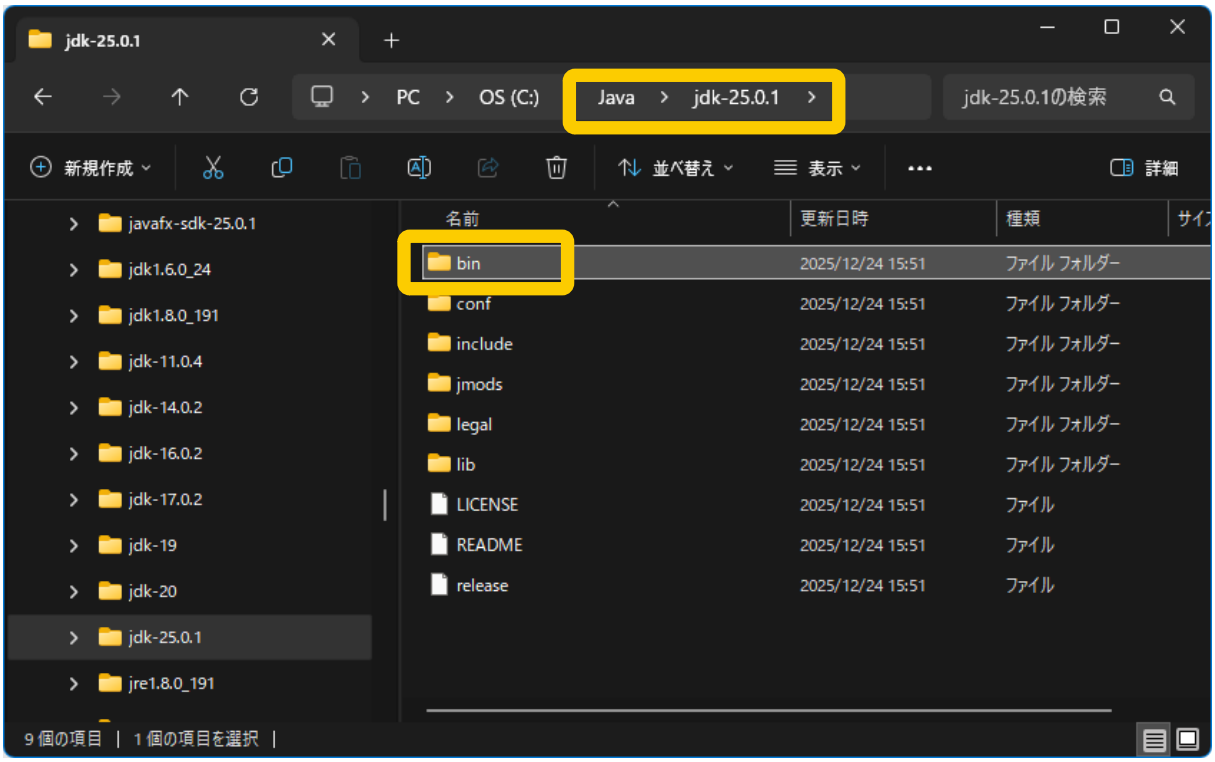
どこに Java をインストールしたかを確認しよう。
インストールした時に、フォルダの設定があったのだが、もう既に忘れているかもしれない。

わたしの設定では、c:¥Java フォルダの下に、いくつかのバージョンが置けるように設定している。（著書の関係上、複数バージョンの Java が置いてあるが、読者におかれては、ひとつのみあれば問題はない）

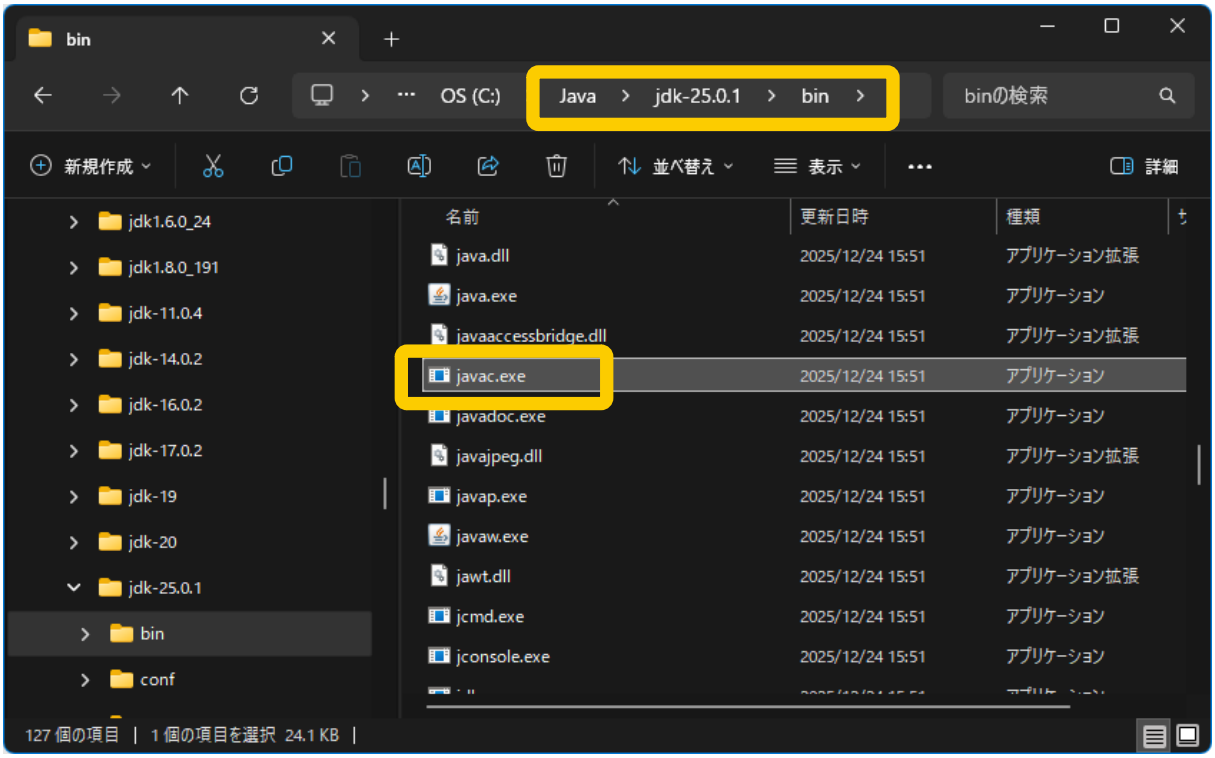


最新のバージョンをどこに置いたかとを確認し、自分の環境に合わせて sta.bat を修正してほしい。

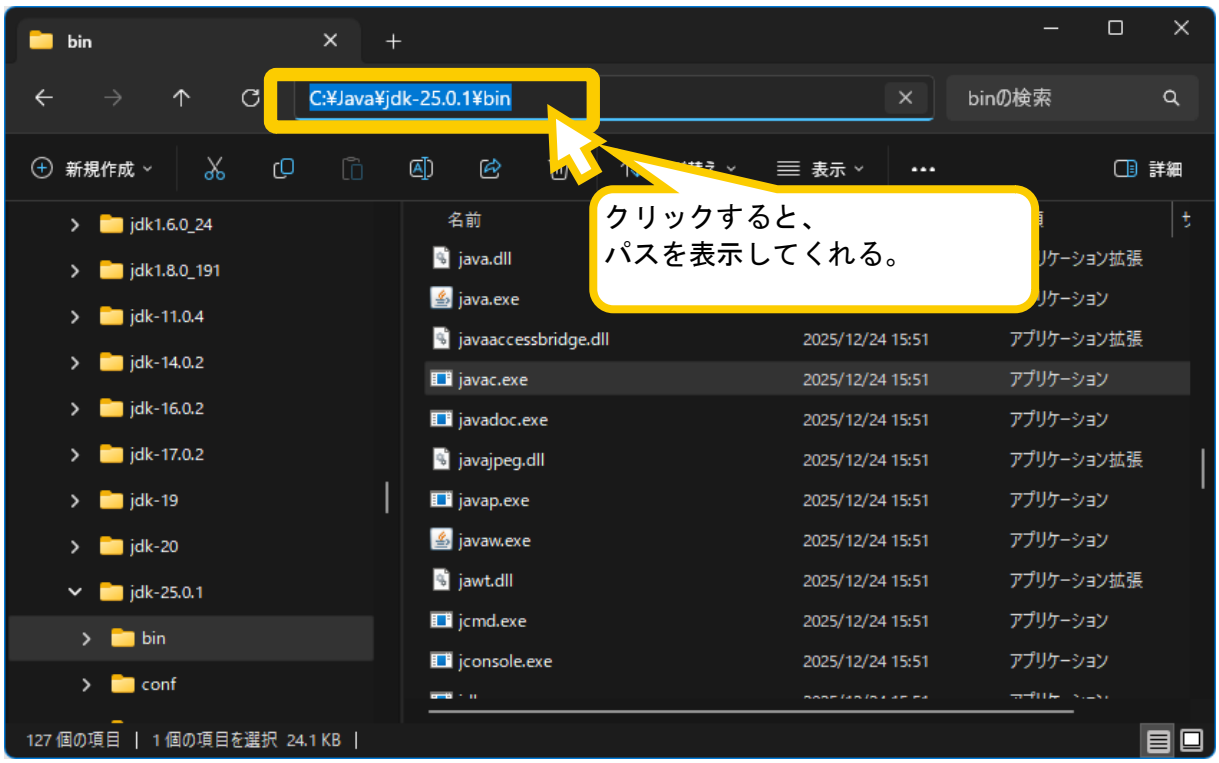
「bin」フォルダはあるだろうか？選択しよう。



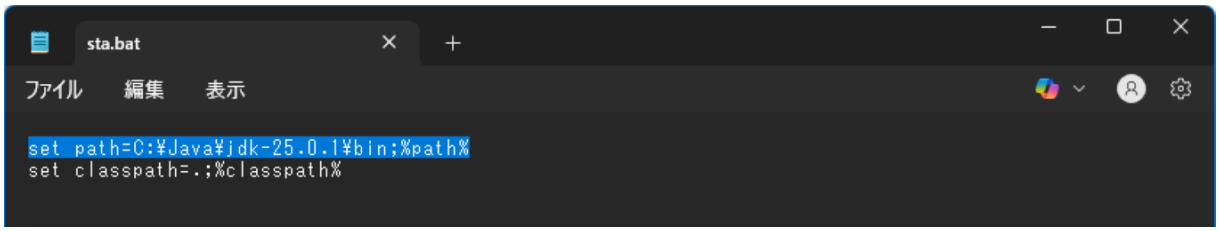
「javac. exe」はあるだろか？
このファイルが Java ファイルをコンパイルしてくれる実行ファイルとなる。



下の図のように、クリックすると、フォルダのパスを全て表示してくれる。
それをコピーする。
［Ctrl］＋［c］キーか、右クリックのコピーなどで、コピーしよう。

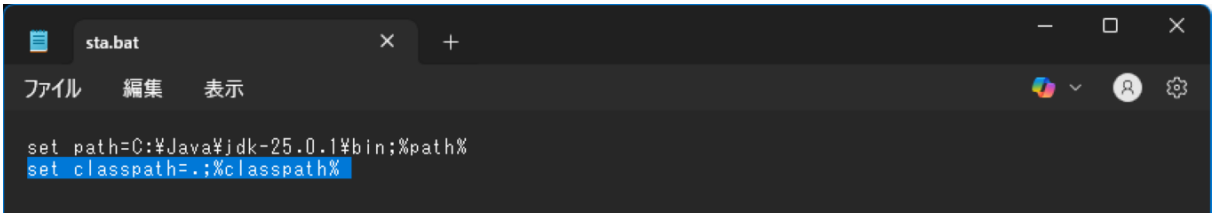


先ほど開いた「sta.bat」はまだ表示しているだろうか？
なければ、もう一度前に戻って、開いてほしい。



コピーしたフォルダパスに置き換える。
「;」セミコロンは残すように注意してほしい。無くなってしまうと、区切りの場所がわからなくなってしまう。Java のコンパイルも実行もできなくなるし、もともとのパスもおかしくなってしまって、影響がでてしまう。

間違いなく設定したら、忘れずに保存してほしい。

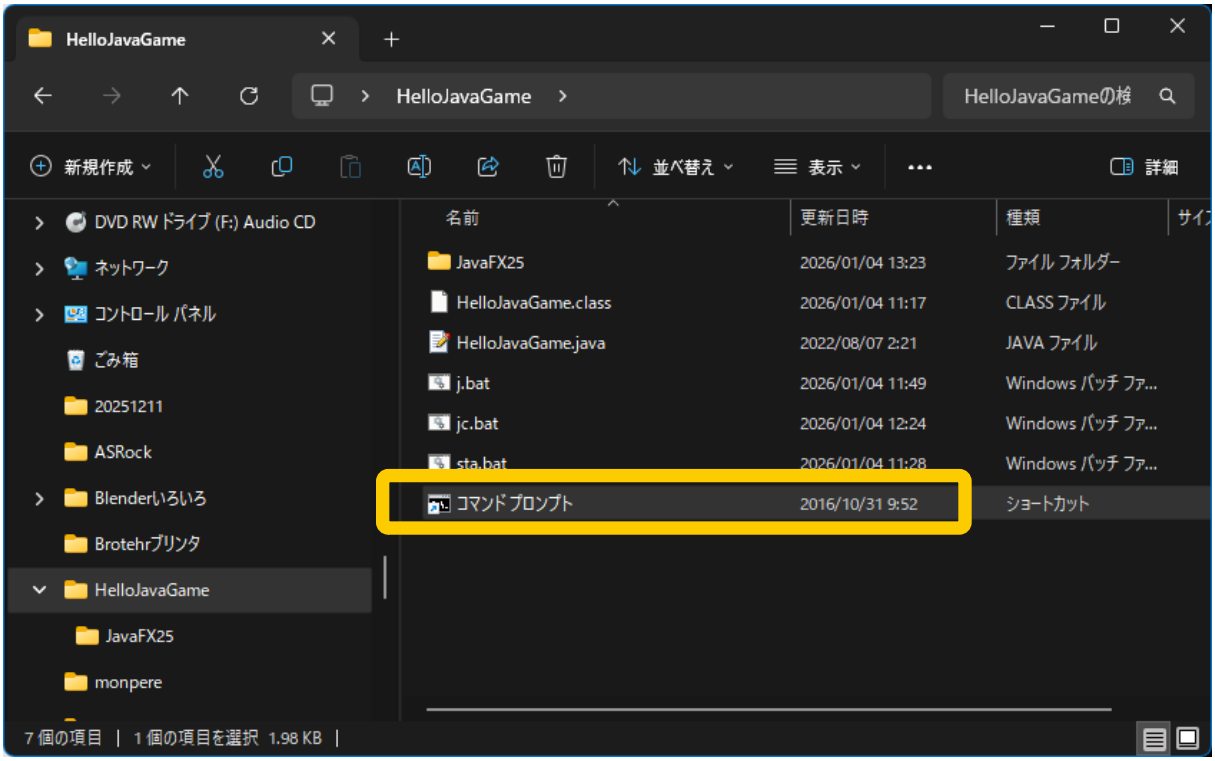


２行目は、クラスパスの設定である。
クラスパスとは、他のクラス（モジュール）を参照する必要がある場合はこの場所に書くことになる。

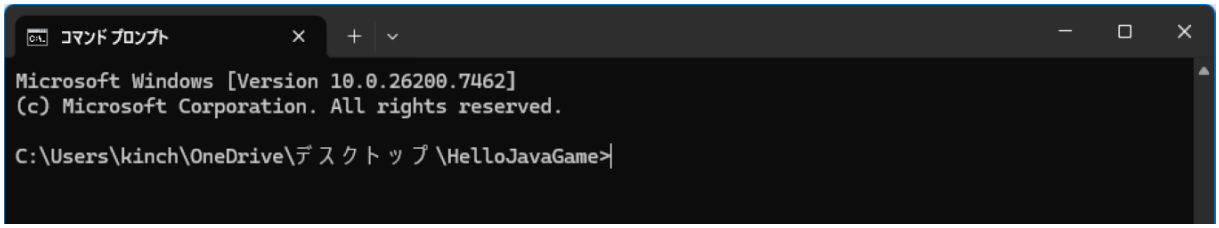
「.」の部分のみ追加している設定となるが、今自分自身がいる場所をクラスパスに追加する。ということになる。
どうも Java9 以降のようだが、設定／環境により、自分自身がいる場所についてもクラスパスに追加しておかないと、実行するファイルが見つからない（NoClassDefFoundError）ることがあるようだ。

おまいじないのように入れて置くのだ懸命だ。

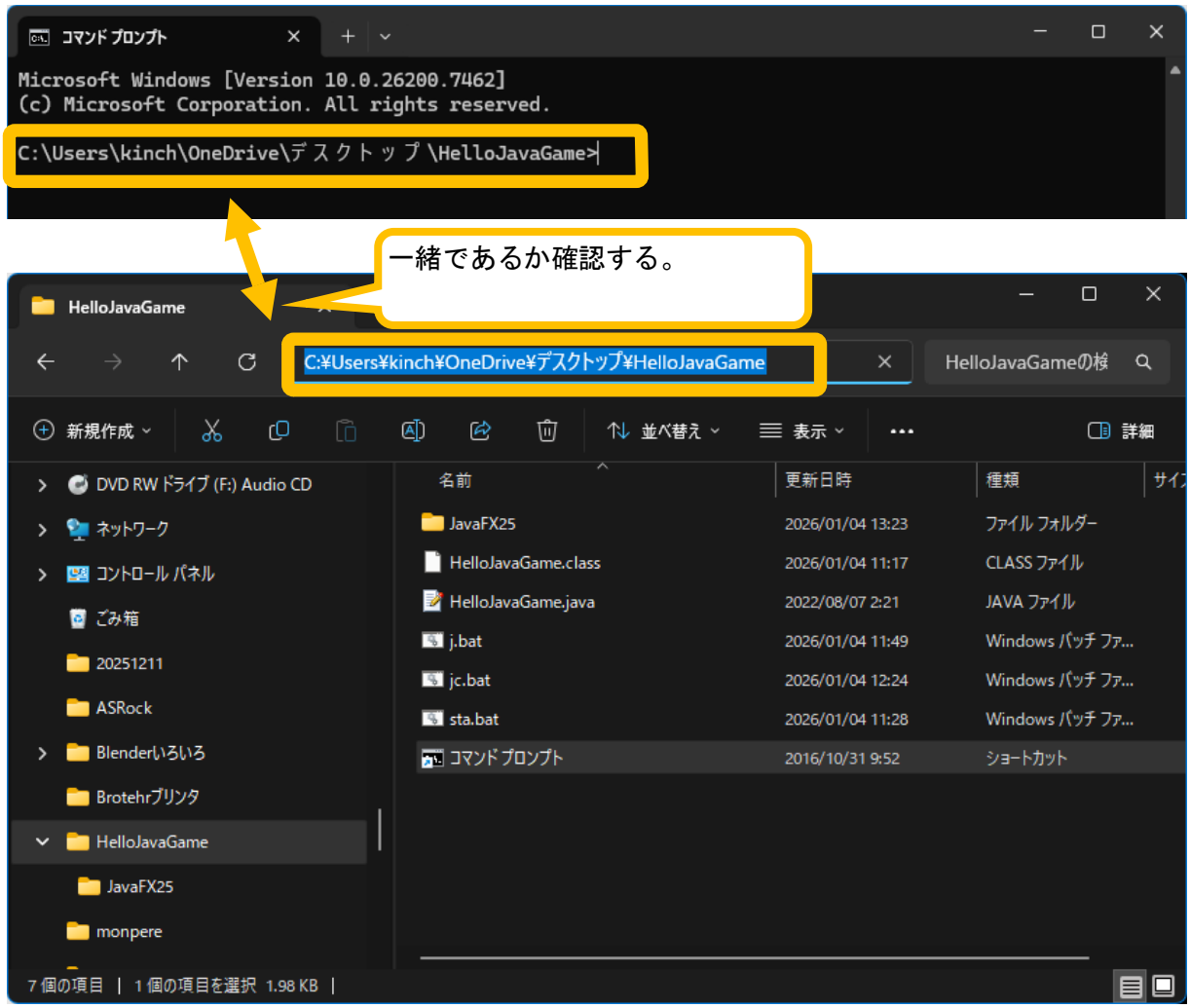
コマンドプロンプトを立ち上げる。
HelloJavaGame フォルダの中にショートカットを置いてある。



コマンドプロンプトが立ち上がった状態。

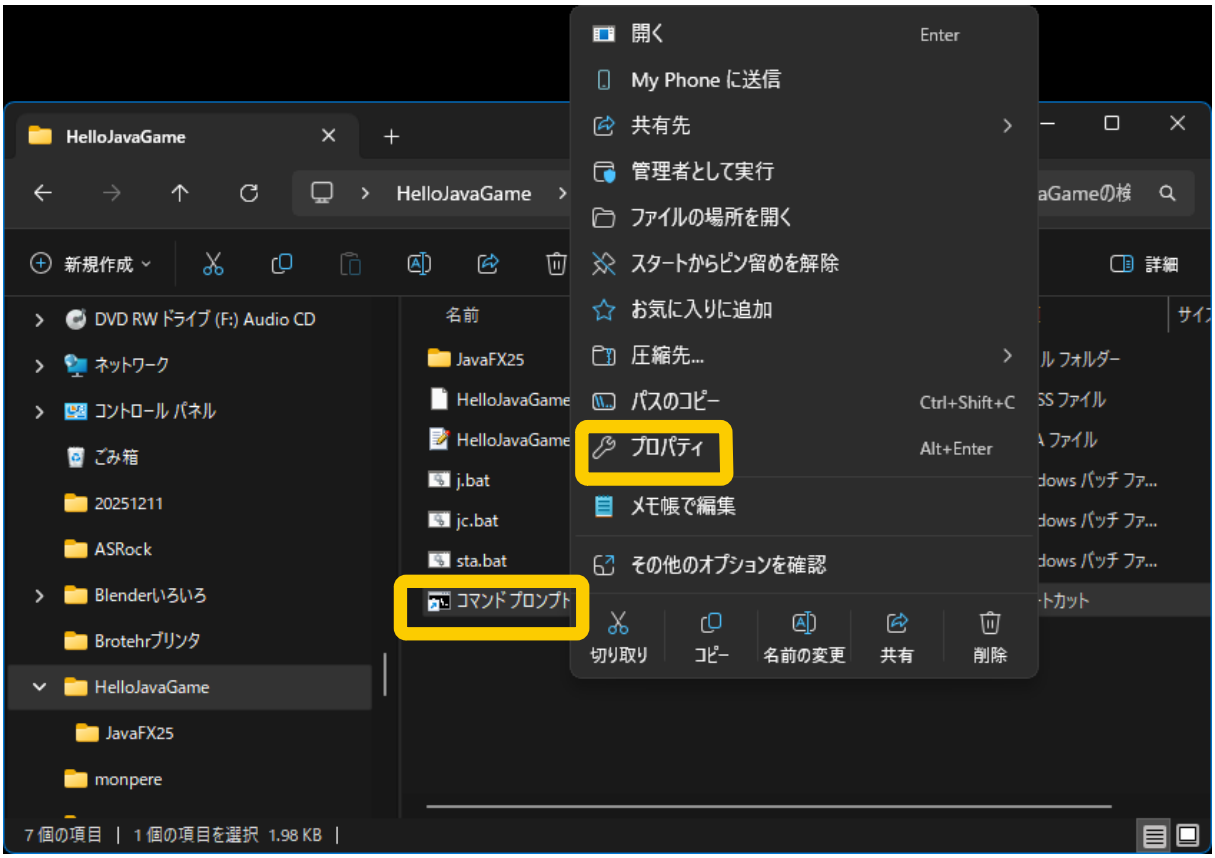


下の部分が一緒であるか確認する。
ファイルの置き場所は、人によって違うので、同じコマンドプロンプトを実行する場所とコマンドプロンプトのショートカットが置いてある場所が一緒であることを確認すればよい。
（下のように「C:¥Users¥kinch¥OneDrive¥デスクトップ¥HelloJavaGame>」ではなくてよいです）



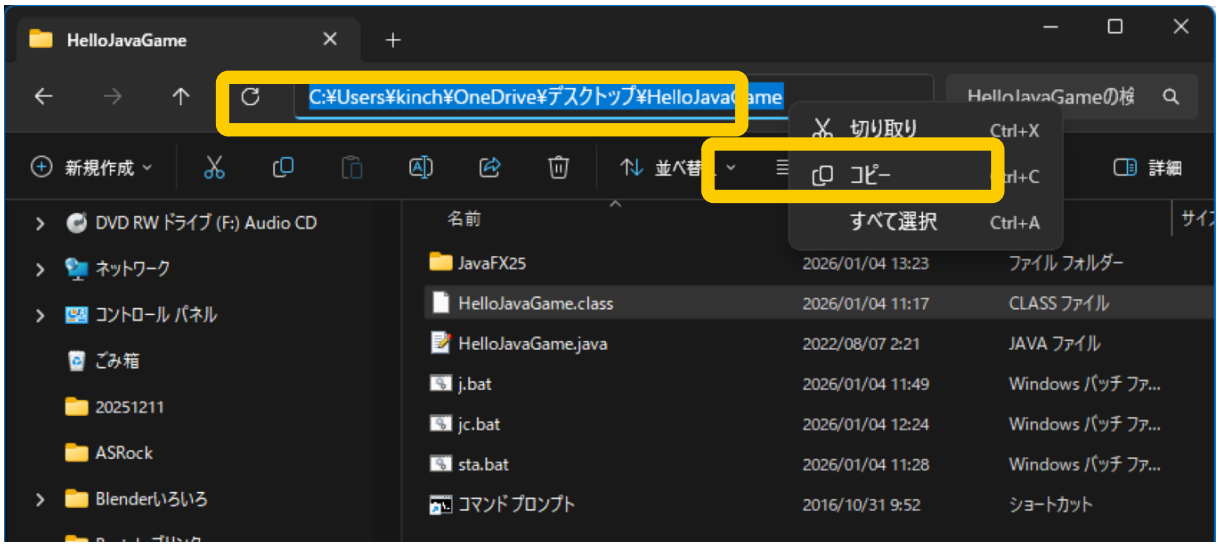
もし、違う場合は、コマンドプロンプトのプロパティを開き
（Windows10 では少しイメージが違いますが、プロパティを開いてください）

（※注意）
　フォルダ位置違う場合だけにしてください。今回は、いろいろなフォルダを使うことになるので、特殊な設定をすると、のちのちトラブルになる可能性があるためです。考えられるのは、コンパイルするが、いつまで経っても反映されない、実行しても直ってない・・・等。

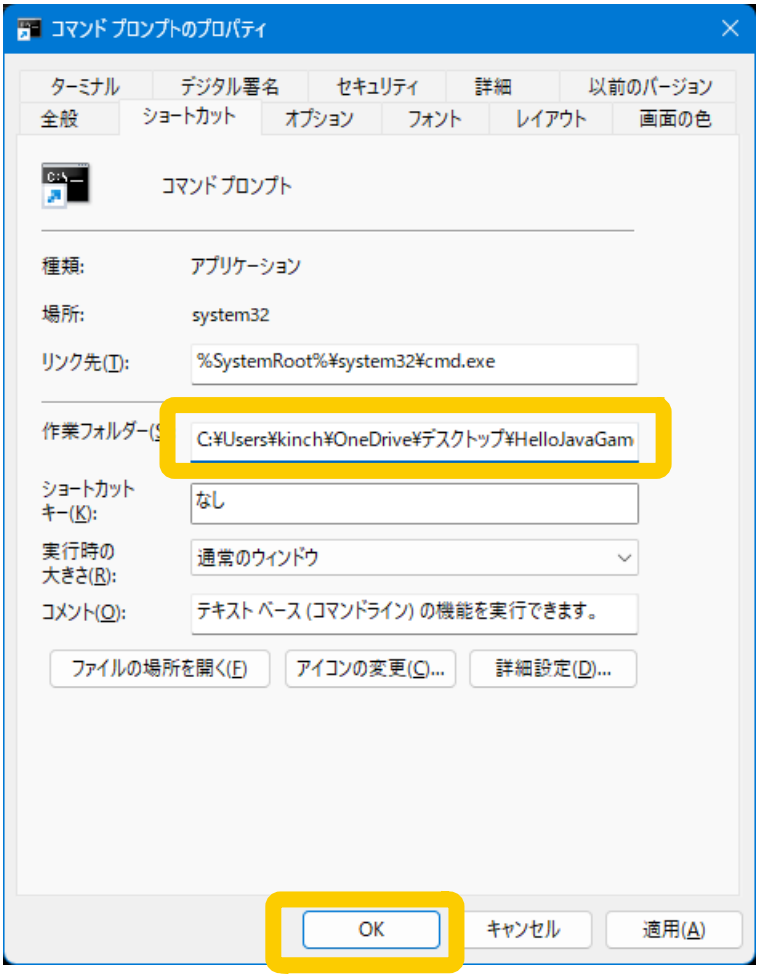


コマンドプロントのプロパティが開く

先ほどのパスをコピーする。

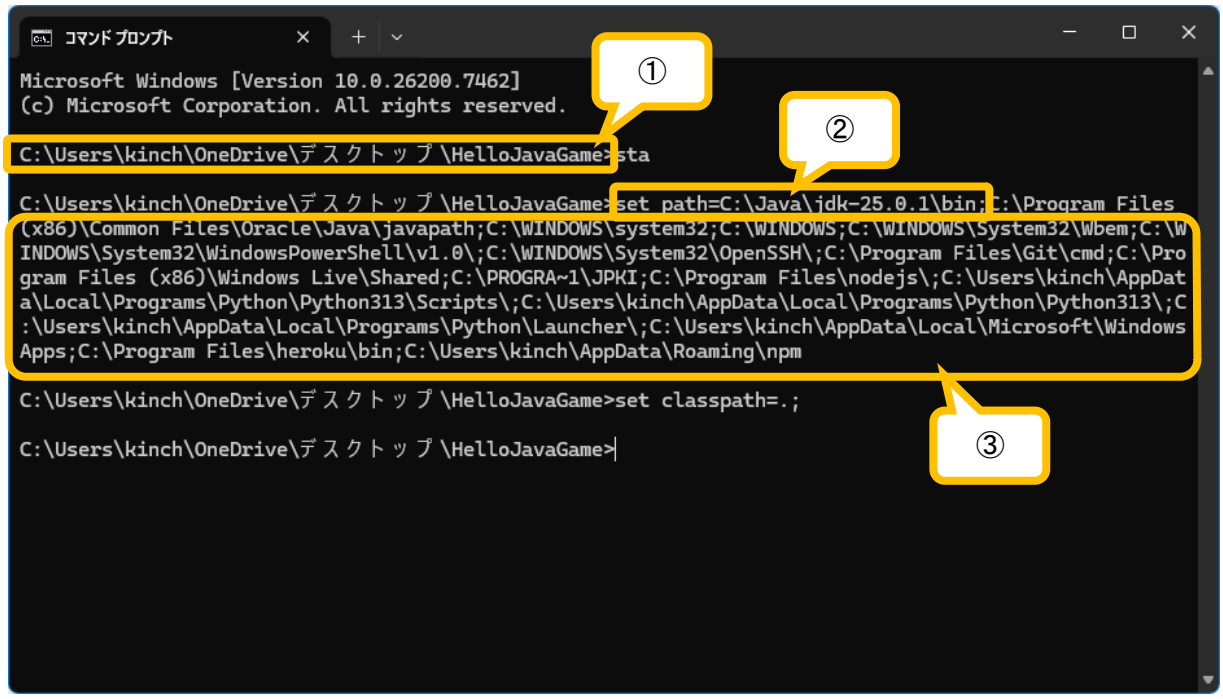


作業フォルダに貼り付ける。
貼り付けたら、一番下の［OK］を押す。



もう一度、コマンドプロンプトを開いて、パスが一緒になったかを確認する。
（確認方法は、先ほどのところに戻って、確認する）

コマンドプロンプトで「sta」と入力して、Enter キーを押してみよう。
すると、以下のような感じで表示されるはずだ。
表示される内容は人によって違う。

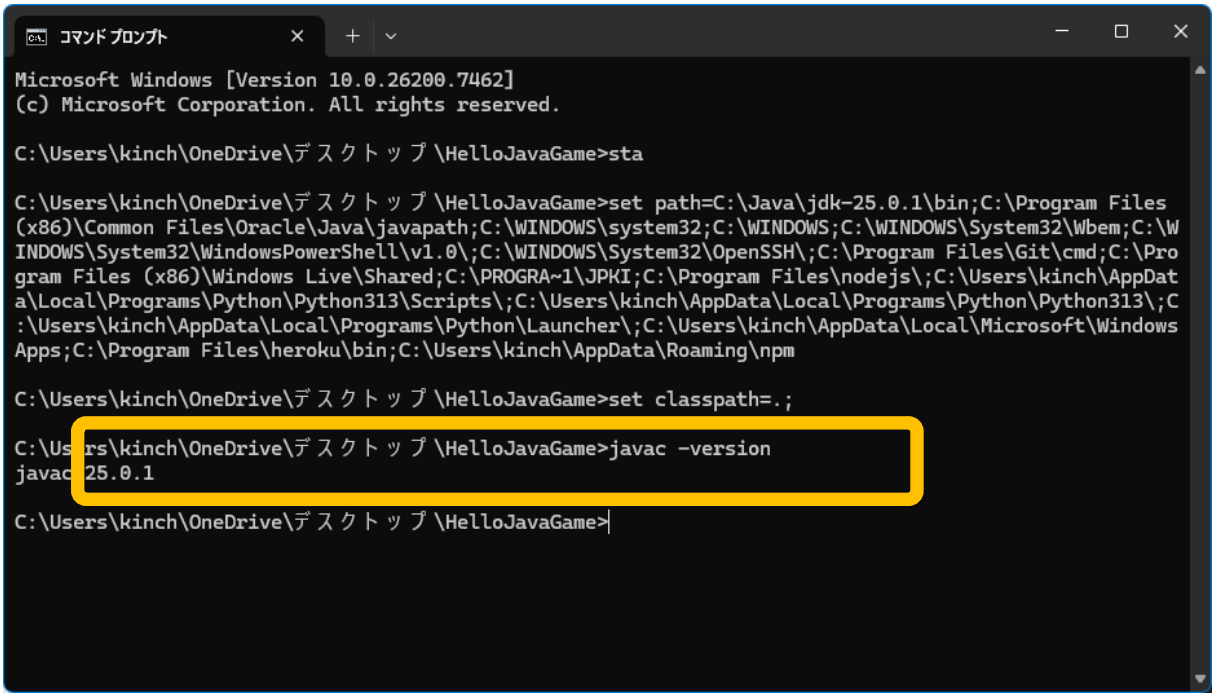


①は、先ほど変更した、JDK のパスになる。
②の部分は、先ほど修正をした Java のコンパイルや実行するファイルがある場所を示している。
③は、もともとのパスである。いろんなソフトをインストールしたりすると、追加されたりする。
「Java」だけではなく、いろんなパスが設定されていることが分かる。

この辺り（パス）の詳しい説明は、違うところで情報を仕入れてください。詳しく説明すると、それだけで一冊の本ができてしまいます。DOS プロンプトやバッチファイルに慣れていない人は、ぜひとも別に学んでほしいと思います。

このコマンドを実行したことで、コンパイルすることが可能になったはずです。
ちなみに、コマンドプロンプトを立ち上げなおしたら、「sta」を実行し直さなければいけません。設定の有効は、コマンドプロンプトの中だけとなるためです。

次は、下のコマンドを実行してみよう。
「javac -version」



バージョンが表示されるはずだ。
ここでは、「javac」は、「25.0.1」であることが分かる。
ここでは、コンパイルできるかどうかを確認しただけだ。
（コンパイルするための実行ファイルが見えていれば、バージョンが表示される）
毎回、この「javac -version」を行う必要はない。

「sta」の実行を忘れたり、「sta」の内容が間違ったりしていると、下のように認識していないというメッセージを返してくる。
このままでは、コンパイルはできない。
内容が間違っている場合は、見直しが必要だ。前に戻って確認をしてほしい。

いざ！コンパイル！

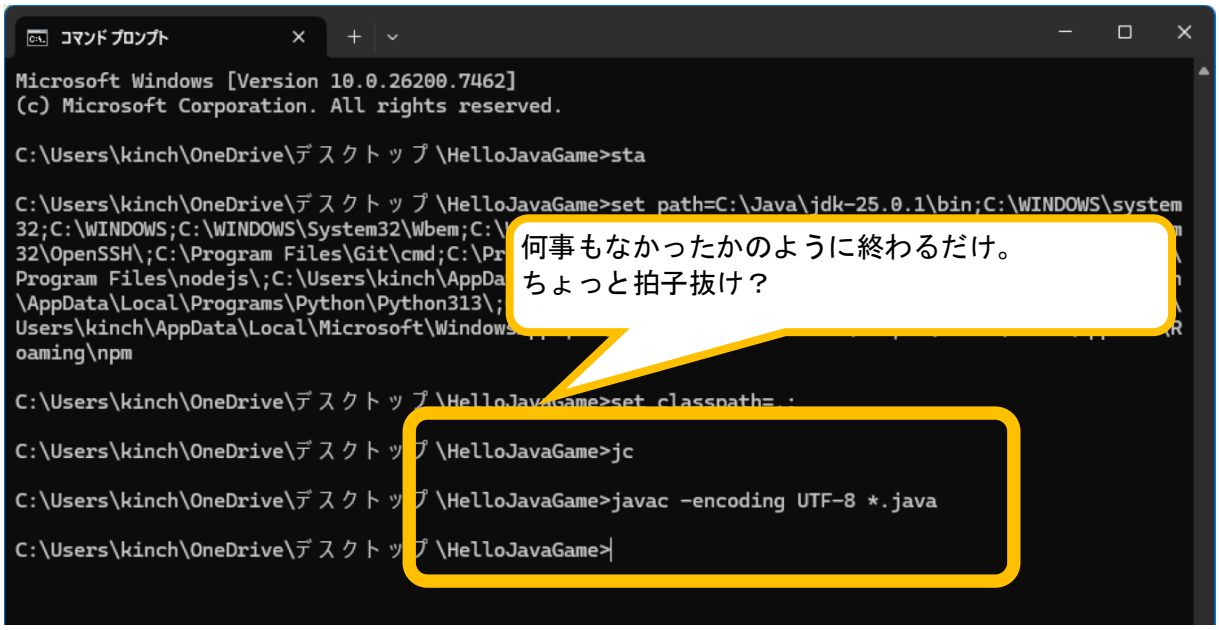
やっと、ここまでたどりついた。コンパイルしてみよう。

ここまでくれば簡単だ。「jc」と入力し実行する。

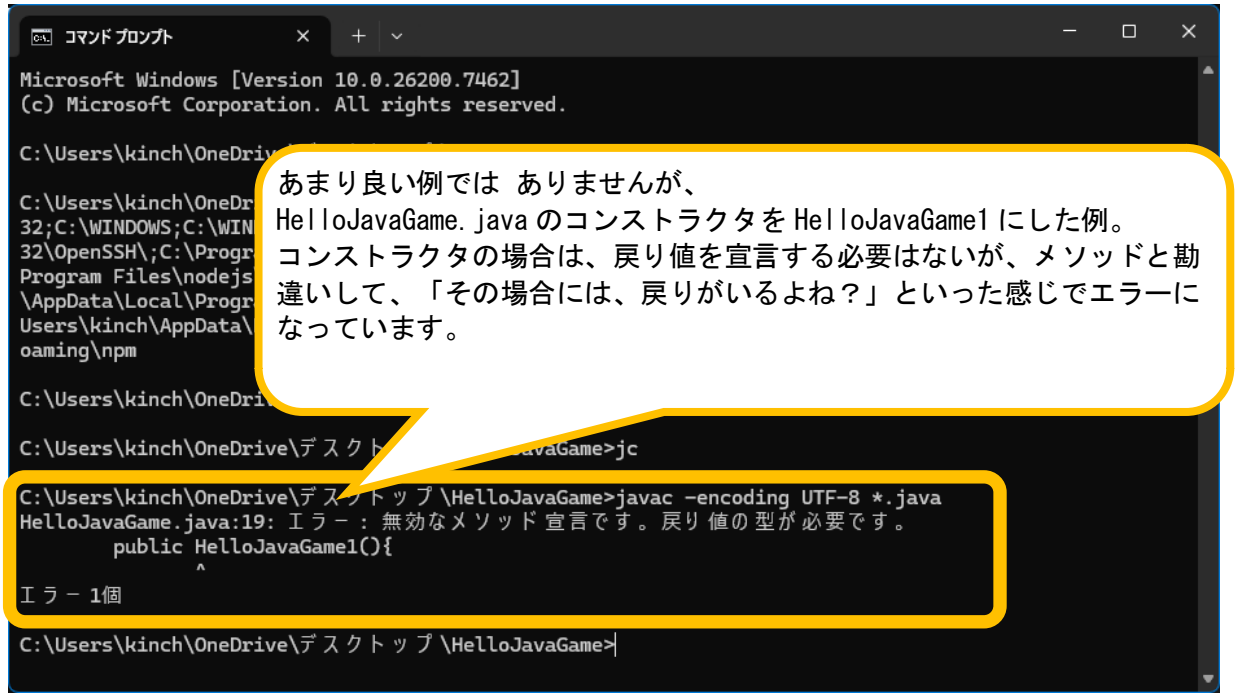
「jc」と入力すると、「javac」というコマンドを実行する。

エラーが無ければ、何事もなかったかのように終わる。

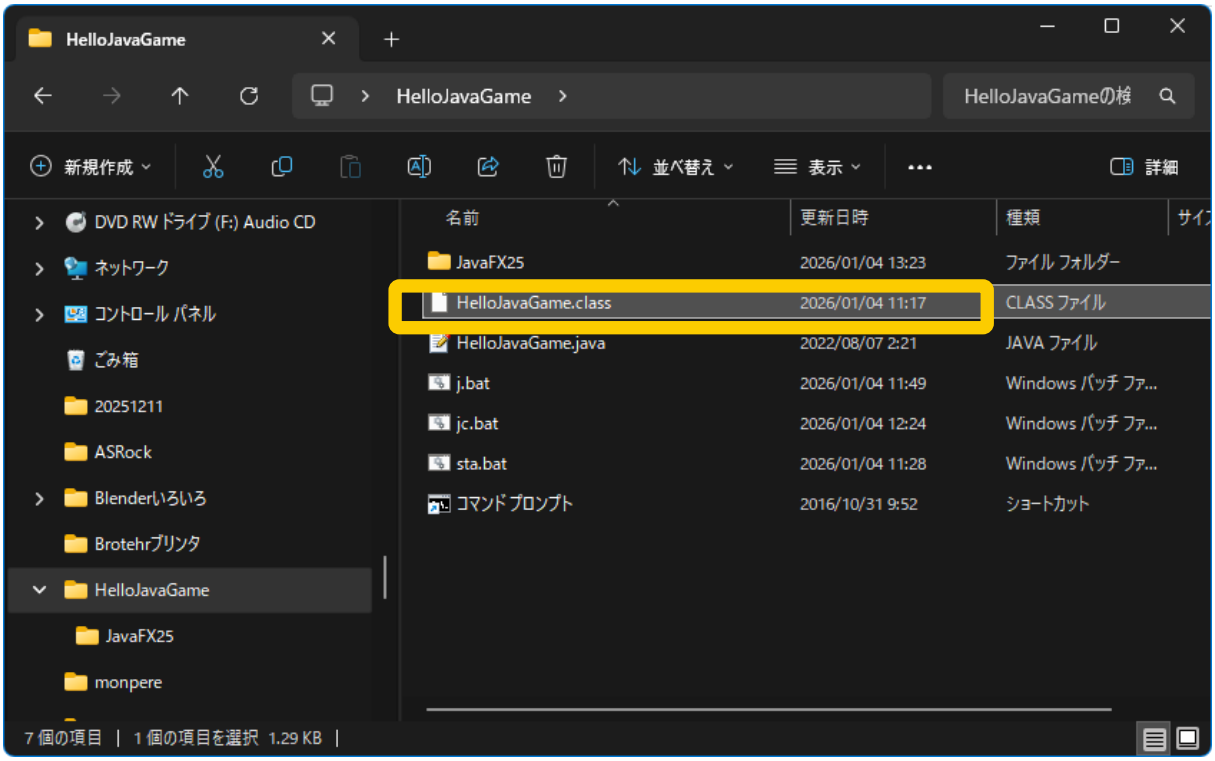
ちなみに「*. java」は、全ての Java ファイルという意味になる。全ての Java ファイルをコンパイルするということになる。



エラーがあったりすると、下のような感じで表示される。



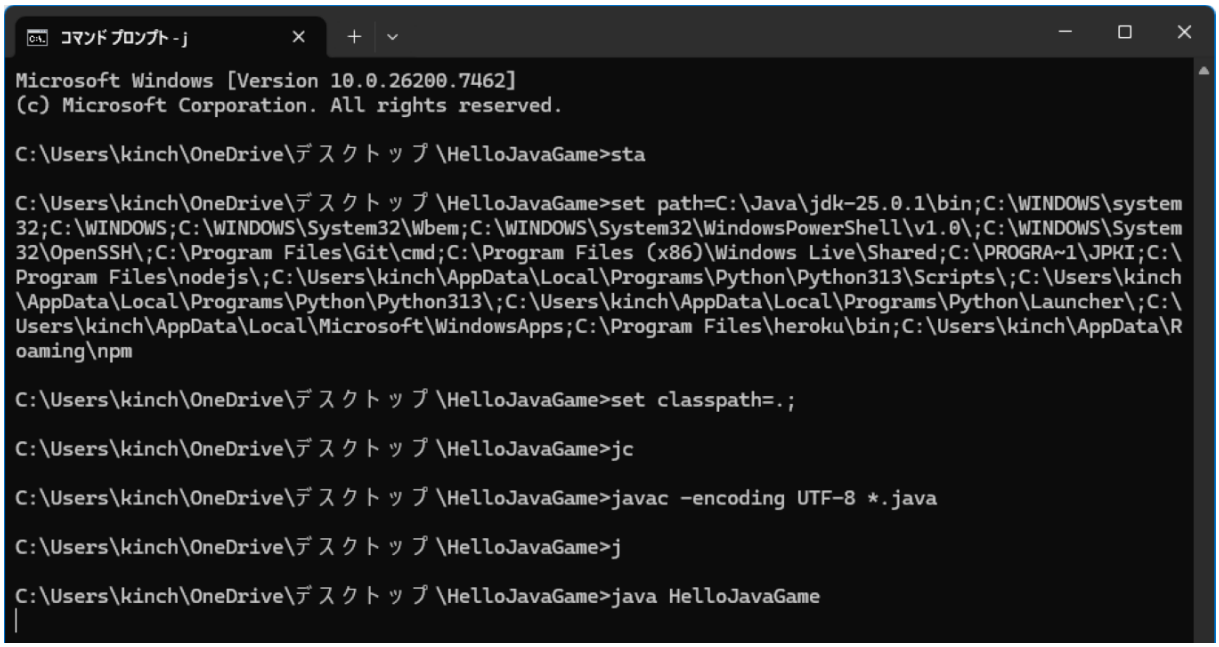
「class」クラスファイルの更新日時を見ると、コンパイルした日時に変わっているはずです。



ここまでくれば、コンパイルについては、バッチリです。

実行しよう！

では、実行してみよう！
「j」と打って、ENTER キーを押してみよう。

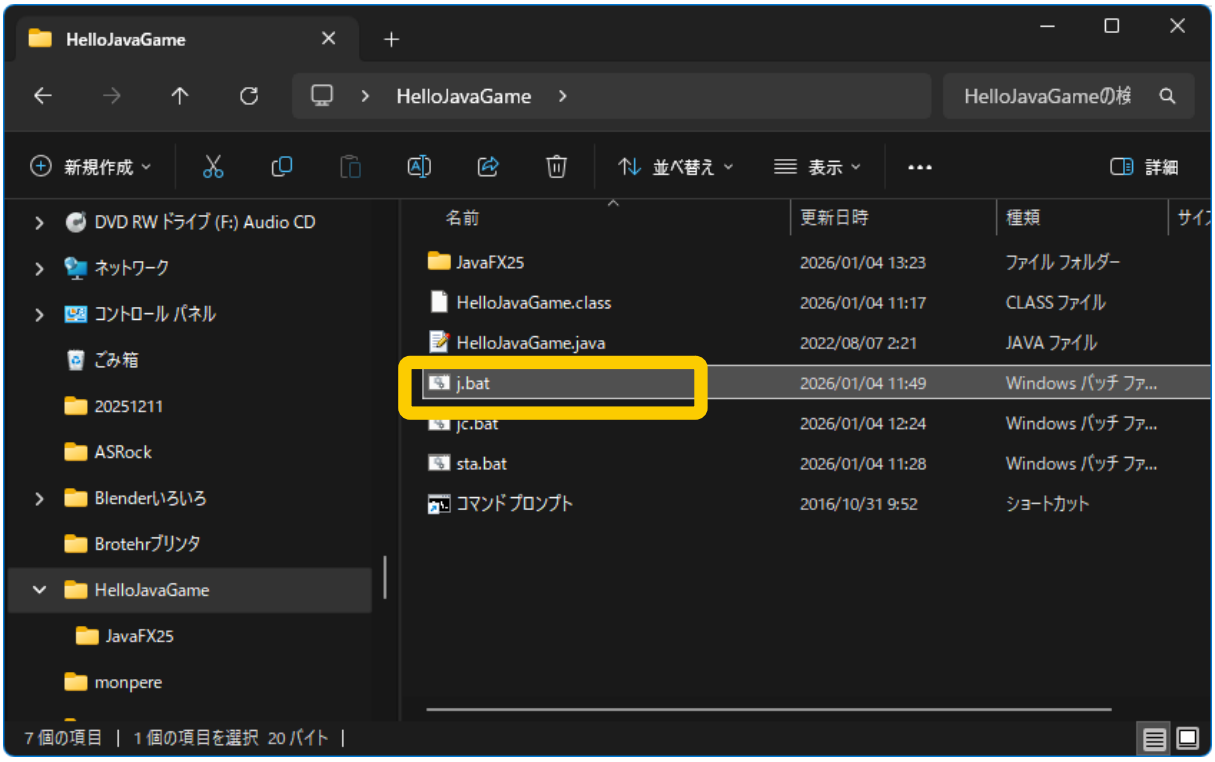


すると・・・

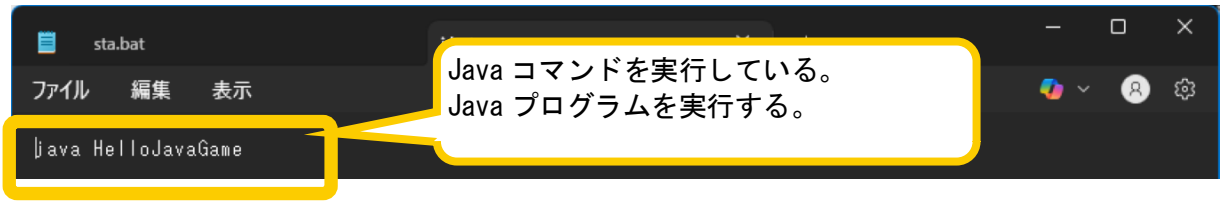
表示されたであろうか？
下のように表示されれば、Java は動いたことになる。

表示されなかった場合は、どこか問題があるはずだ。もう一度、確認してみよう。

実は、「j」を実行しときに、「j.bat」が起動されている。



「j.bat」の中身を見てみると、Java コマンドを実行していることが分かります。「HelloJavaGame」を起動する。ということになります。

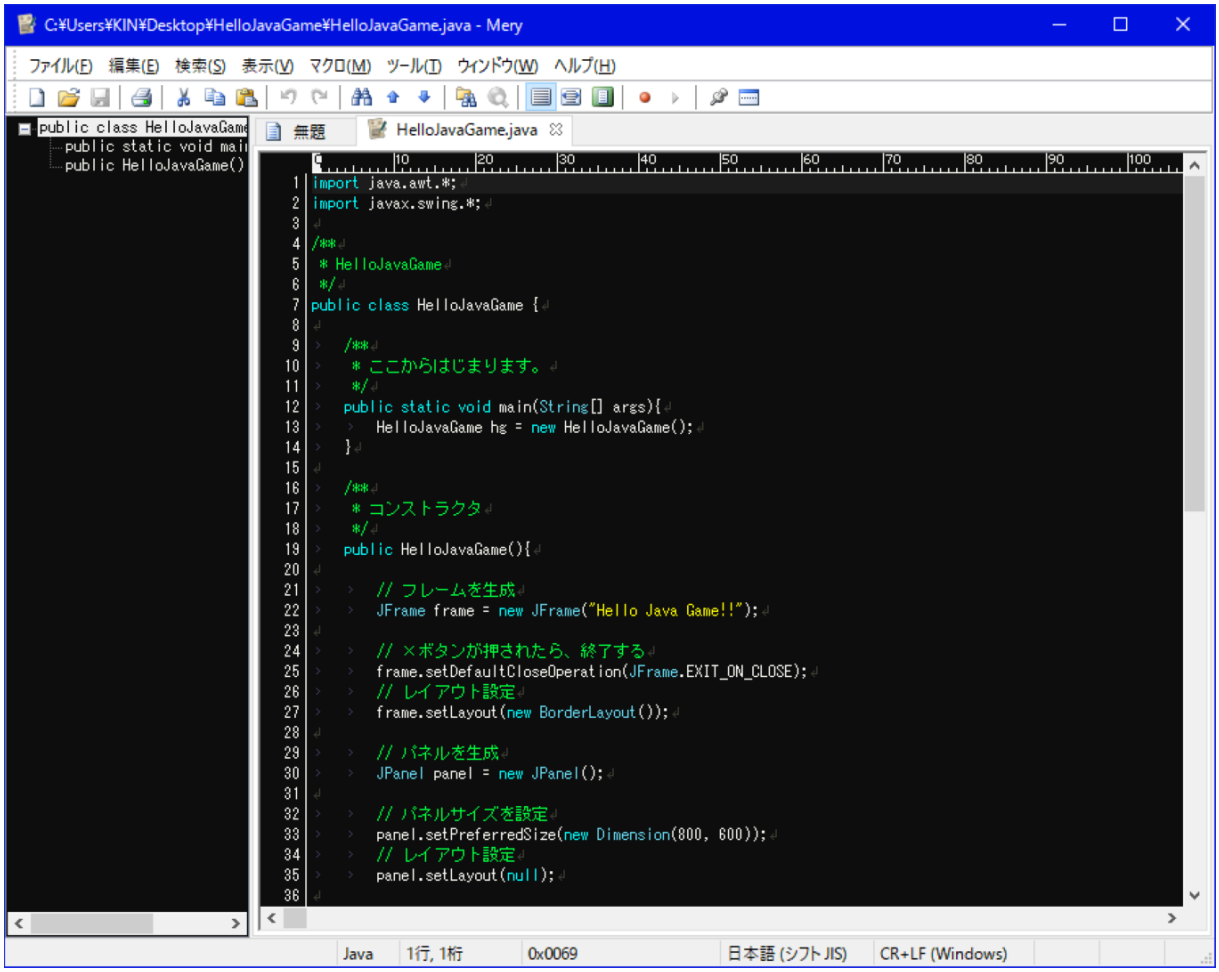


毎回「java HelloJavaGame」と入力するのは大変なので、「j」で起動できるようにしています。

１－３　プログラムを見てみよう

　では、プログラムを見ていこう。初級編（共通編）でいきなり「プログラムを組んでみよう」なんて言われると、ドキッとしてしまう人がいるかもしれない。少しずつでもいい。プログラムを眺めることに慣れていこう。アレルギーを起こさないように、少しずつ慣れていこう。はじめは流し読みでかまわない。

　「HelloJavaGame.java」を開いてみよう。初めての人は、ぜんぜん分からないコードが書いてあるかもしれないが、最初の方から説明していこう。



```
import java.awt.*;
import javax.swing.*;
```

import 文は、他の場所にあるプログラムを探すために設定しています。「java…」となっていることから分かるだろうか？Java（JDK）のインストールされた中に含まれているモジュールを利用できるように宣言している。例えば「Font」クラスは「java.awt」パッケージに、「JFrame」クラスは「javax.swing」パッケージに入っている。

22 行目で「JFrame」を利用しているが、import で先に宣言しておくことにより、「JFrame」のみの記述でよくなる。もし、import を宣言していないと、

```
 javax.swing JFrame frame = new javax.swing JFrame("Hello Java Game!!");
```

と、実際の場所を記述しなくてはいけなくなる。

```
/**
 * HelloJavaGame
 */
```

/** ～ */ の間は「コメント」となる。クラスやメソッドの始まりや定数、インスタンス変数などは、こんな感じで書いておくと良いだろう。

プログラムの中では、「//」で書くことが多い。「//」は１行のコメントになる。

あとあとプログラムを見返して、なにをやっているのか分かりやすいように、コメントを入れておくことは大事だ。大体こういうところは手を抜いてしまうが、大きなプログラムを作るときは、必ず書いておいた方がよい。

もっと大きなプログラムだと、何人もの人で作ったりするが、コメントが入っていなかったら、他の人から「え～っ！！」って、思われるかもしれない。

コメントの部分は、紙面の都合上、飛ばしていくことする。

```
public class HelloJavaGame {
```

「public class」は、公開クラスであること（他からも利用可能なクラスであること）を示している。

「HelloJavaGame」は、このクラス名（ファイル名と一緒）になる。

```
    /**
     * ここからはじまります。
     */
    public static void main(String[] args){
        HelloJavaGame hg = new HelloJavaGame();
    }
```

これは、メソッドと呼ばれるものとなる。「public static」では、クラスを new（インスタンス化）することなく、呼び出しができるメソッドとなる。「インスタンス」という言葉が出てきたが、まだ覚えなくてもよい。いつかまた、出てくることになる。
「void」は、返却が無い（リターンが無い）メソッドを表す。

「main」は、「main メソッド」であることを表す。

カッコ内は引数と呼ばれる。必要な情報を渡すときに使う。例では「String」という文字型の配列を渡すことになっている（約束をしている）。[]中カッコは配列であることを表します。String を複数渡しているということになる。

実はこの「main」メソッドは特別な意味がある。Java を実行する時に最初に呼び出されるのが、このメソッドとなる。覚えておこう。

次の行はこの「HelloJavaGame」を生成している文となる。「new HelloJavaGame()」で呼び出されて生成される。生成されたオブジェクトは左辺である「hg」に格納されます。「hg」は自分で決めます。今回は「hg」にしてみた。

最後の行「 } 」は、このメソッドの終わりを表す。

「main」メソッドはこれで終わりとなる。

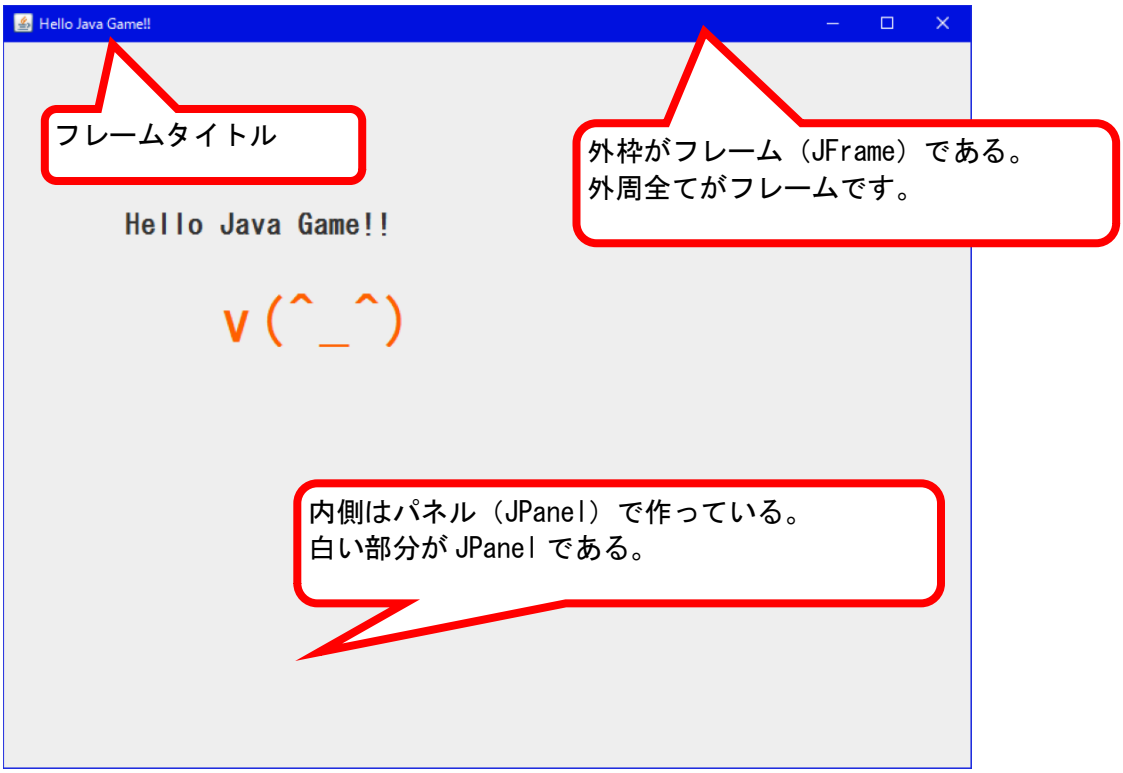
```
    /**
     * コンストラクタ
     */
    public HelloJavaGame(){
```

コンストラクタ。「main」メソッドで生成した時に呼び出される場所となる。
「new HelloJavaGame()」で呼び出されれる。
コンストラクタにリターンはない（返却はない）。そのため、main メソッドのように「void」は書く必要はない。
（実際には、そのインスタンス自体が返却されてくる）

```
// フレームを生成
JFrame frame = new JFrame("Hello Java Game!!");

// ×ボタンが押されたら、終了する
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
// レイアウト設定
frame.setLayout(new BorderLayout());
```

フレームを生成する。フレームとは、Java のフレームを表す。



「new JFrame」で JFrame オブジェクトをインスタンス化している。「インスタンス化」と、わけの分からない言葉かもしれないが、とにかく「new」をしたというです。
中の引数は、フレームのタイトルとなる。（決まっている）

次の行は、×ボタンが押されたら、終了するようにしている。この設定をしておかないと、フレームは消えるが、コマンドプロンプトに戻ってこない状況となってしまう。
引数の「JFrame.EXIT_ON_CLOSE」がコマンドプロンプトにまで戻ってくる設定となります。

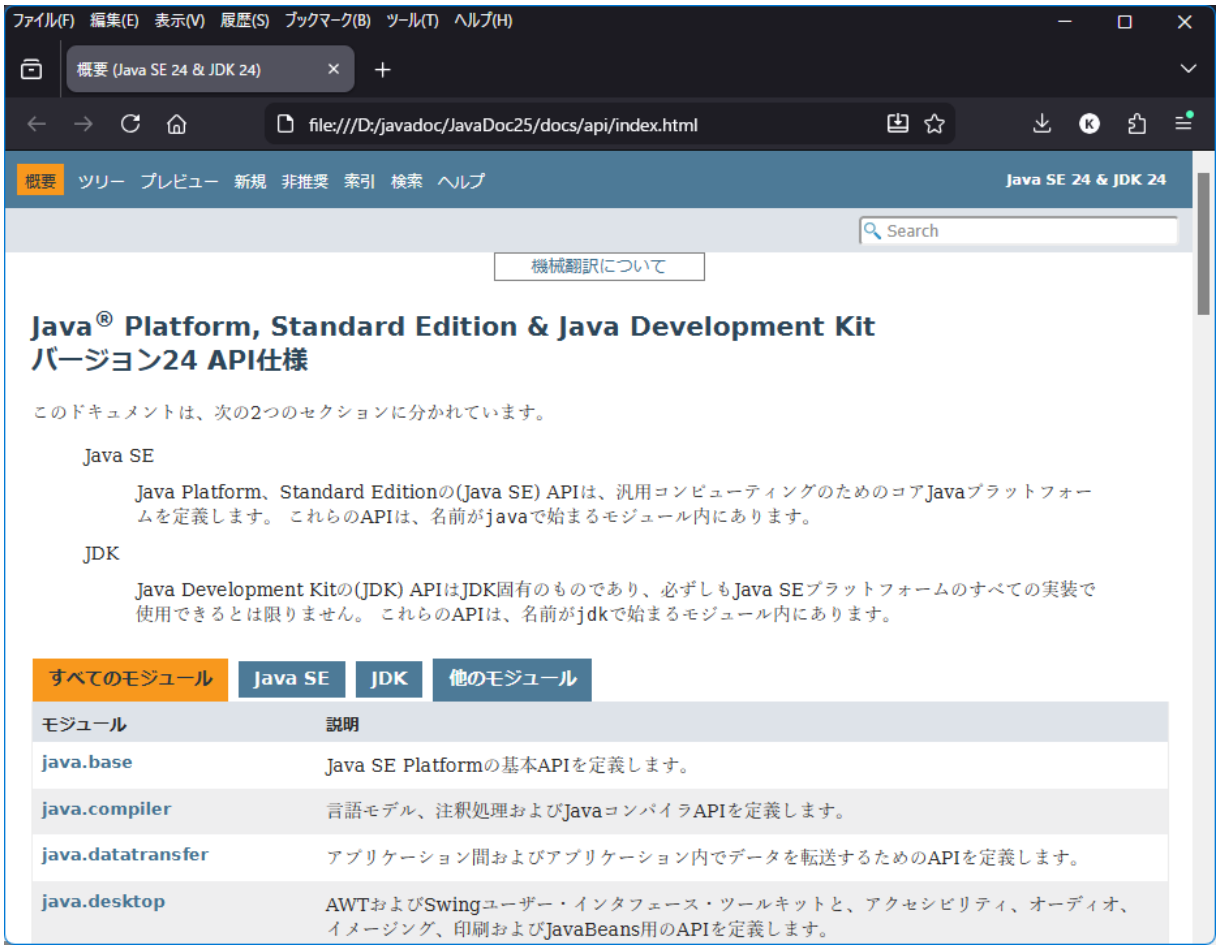
詳しくは「JavaDoc」を見てほしい。と、いうことで、JavaDoc の紹介をしていく。

すでに
Install_yyyymmdd.pdf
（yyyymmdd は最新バージョンと読み替えてください）
を見てくれた方は、JavaDoc を入れてくれているはずだ。

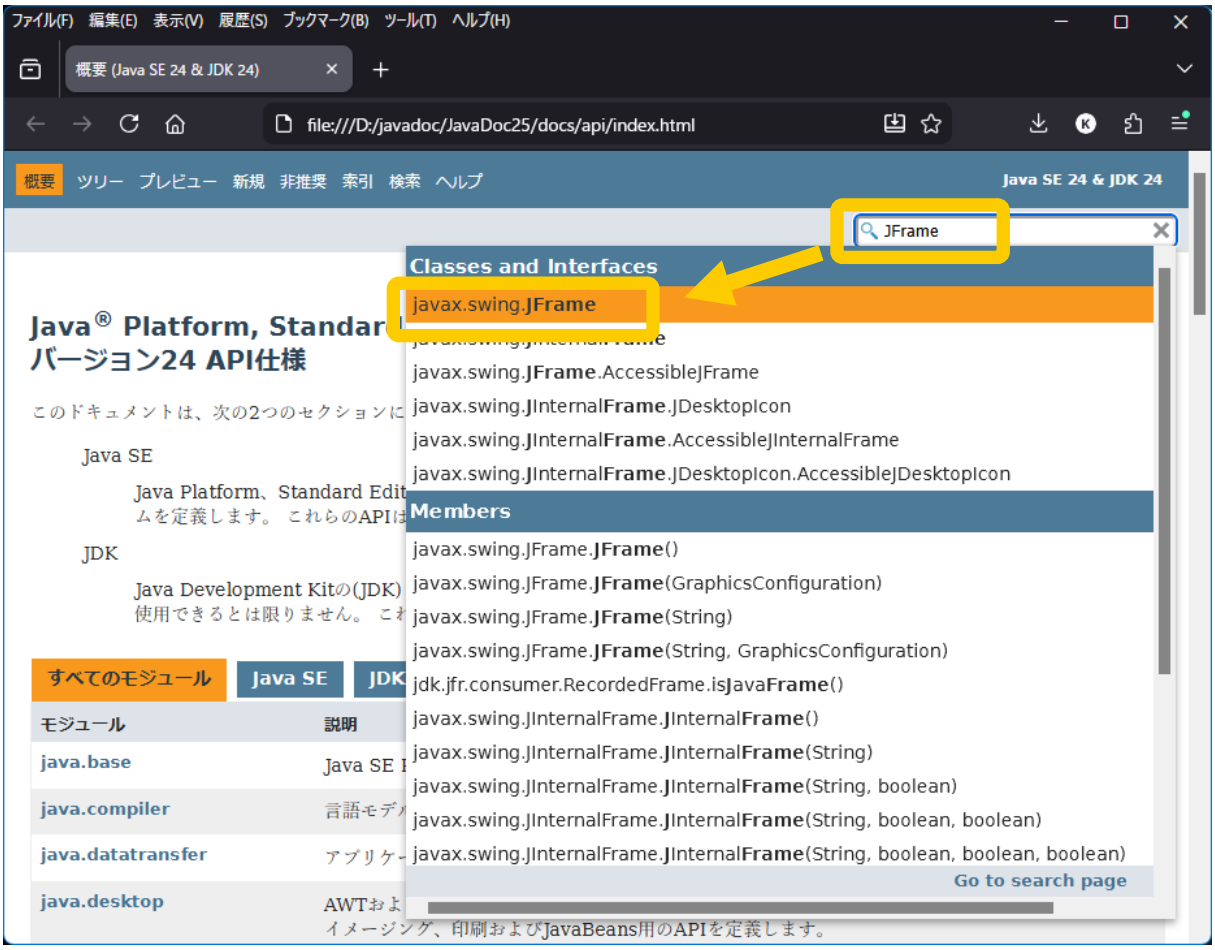
まだ入れていない方は、この機会に入れてほしい。
（上記のファイルを参照し、入れてほしい）

　すでに index.html のショートカットを用意してくれているはずだ。
開いてみよう。

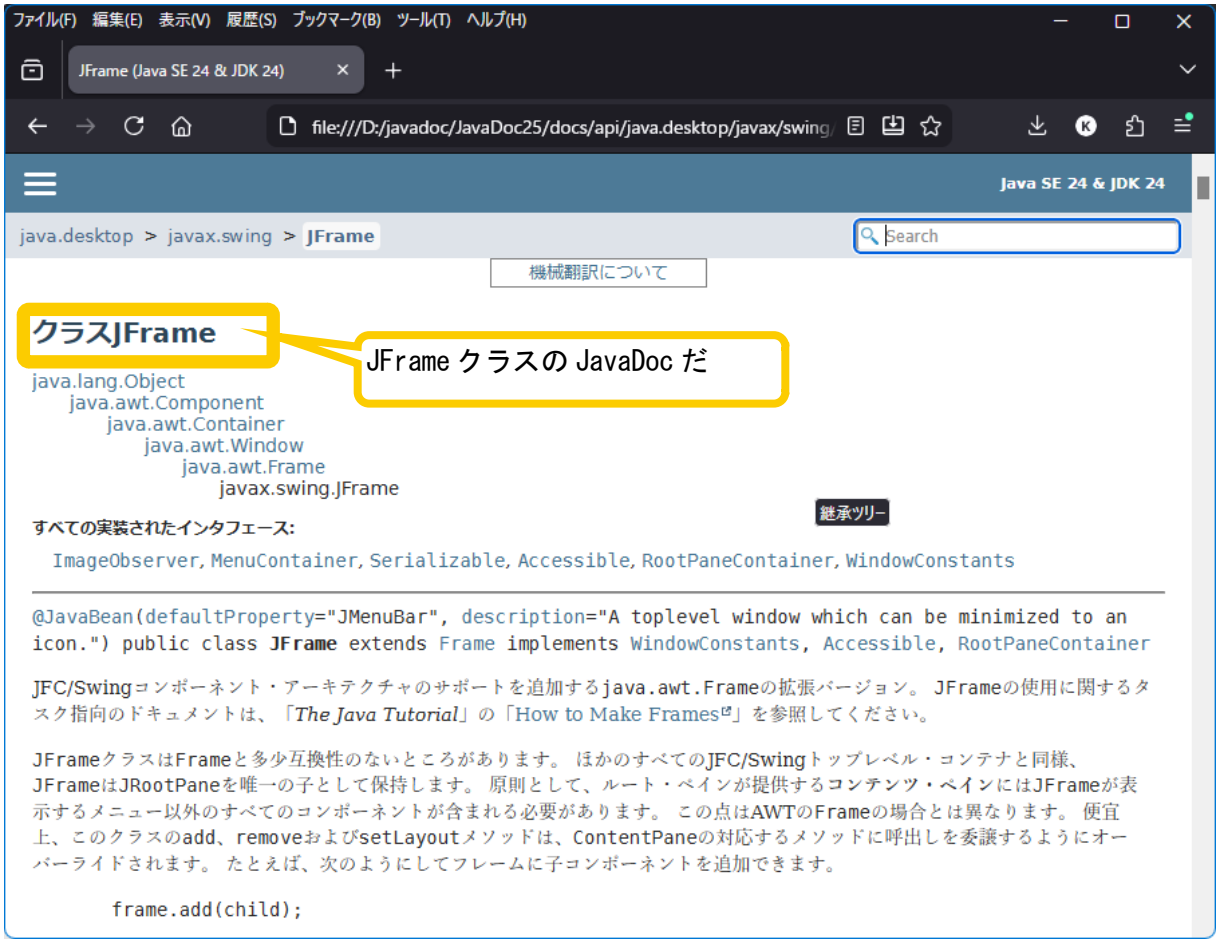
JavaDoc が開いた。



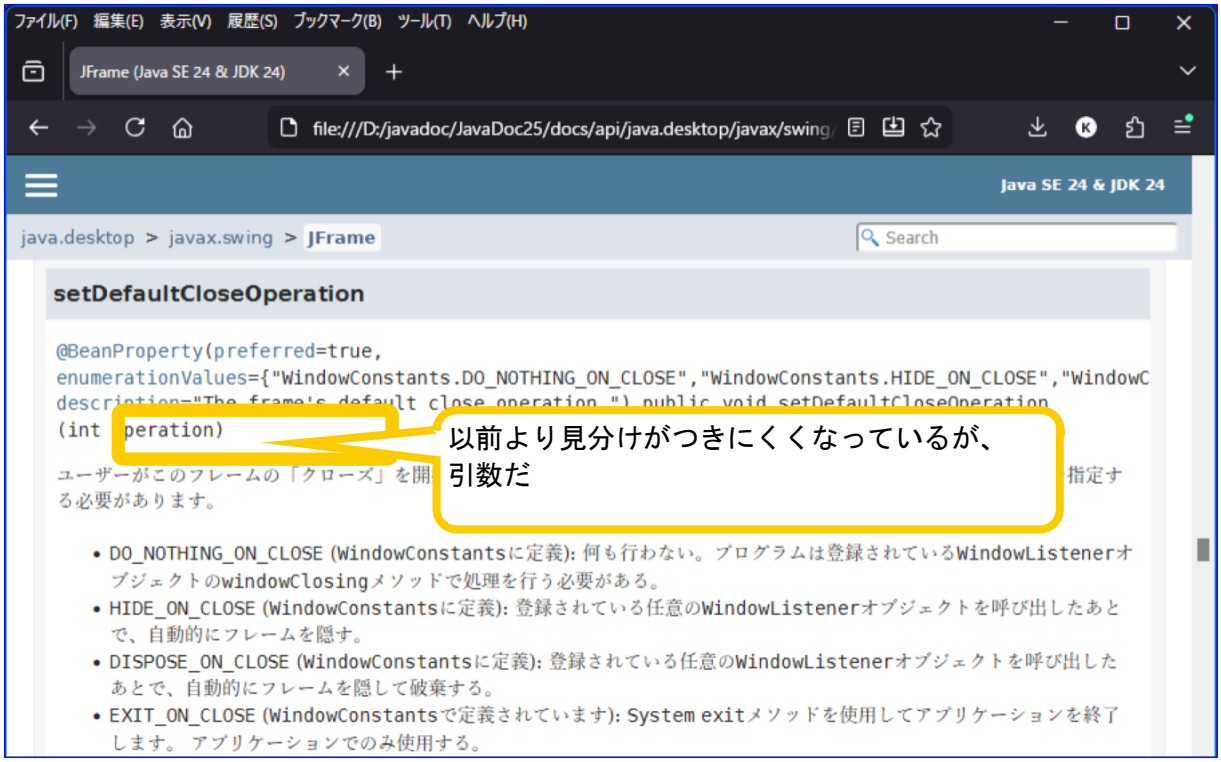
JFrame クラスを見てみよう。
右側の検索窓に JFrame と入力し、javax.swing.JFrame を選択しよう。



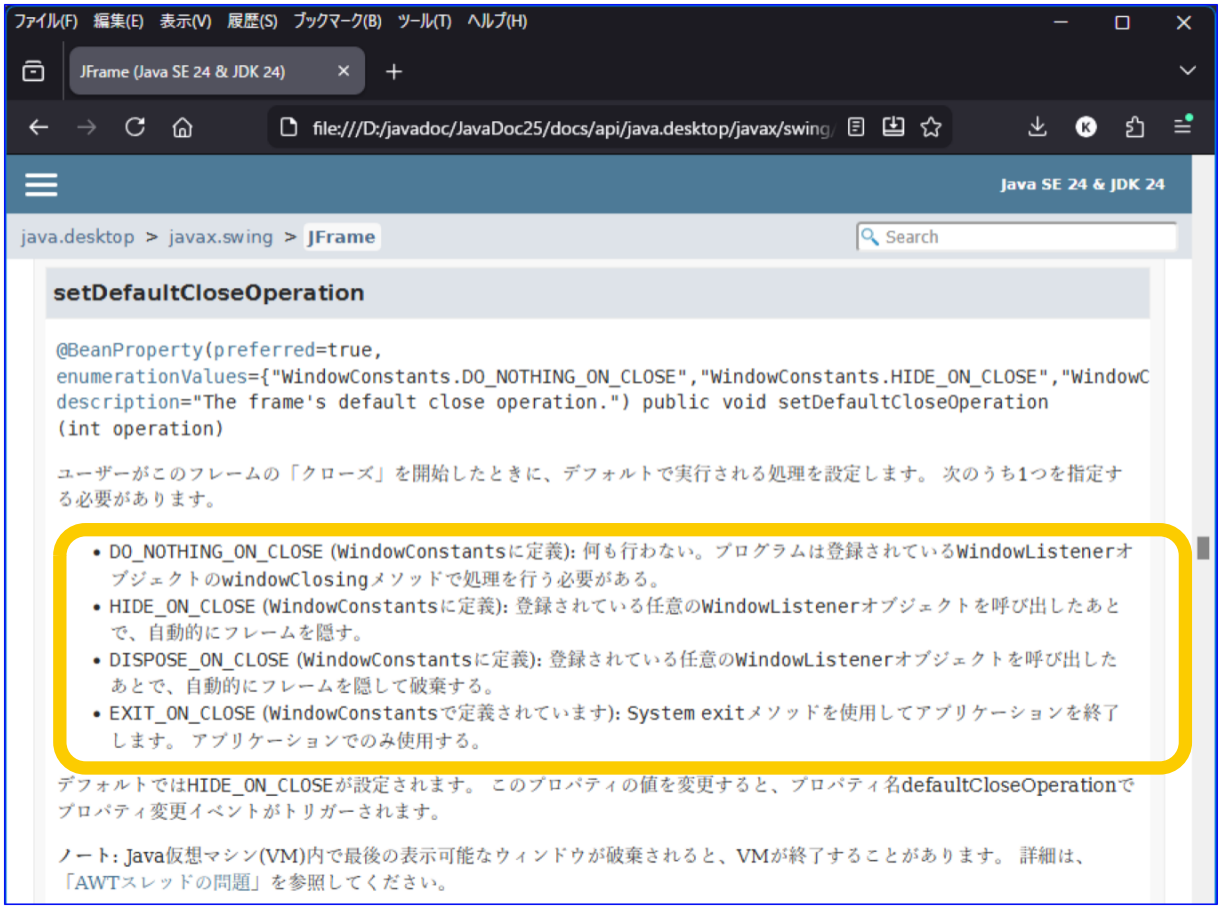
すると、JFrame の JavaDoc が開く



スクロールすると、setDefaultCloseOperation メソッド（今回、呼び出しているもの）を見つけることができるはずだ。



もう少し下を見てみると、引数の詳細が書いてある。
元々英語の JavaDoc のため、分かりにくい表現もあるかもしれないが、多少はガマンしよう。それでも分からなければ、ネット等を利用してみよう。



「EXIT_ON_CLOSE」は、「System の exit メソッドを使用してアプリケーションを終了する」とある。つまり、アプリケーションを終了することなる。
デフォルトの場合は、「自動的にフレームを隠す」と言っているが、アプリケーションを終了するとは言っていない。

数ページをつかって JavaDoc の紹介をしてきた。
話を戻そう。

次はレイアウトの設定を行っていく。「BorderLayout」というレイアウトを利用している。ボーダーレイアウトの詳細な説明は、話が複雑になるのでプチコラムに書いておくが（とりあえずは分からなくてもよい）、フレームのレイアウトはボーダーレイアウトを利用している。

プチコラム １

ボーダーレイアウトとは、５つの領域に分けて管理するレイアウトである。中央、北、南、東、西に配置することができるが、それ以外ができないレイアウトとなる。
パネルを中央に配置しているが、パネル側では大きさを指定できないレイアウトとなる。つまり、レイアウトがパネルの大きさを決めている。
レイアウトの中央のみに配置すると、パネルの大きさはフレームの内枠までの大きさとなる。フレームの大きさを変えたとしても、それにともなってパネルの大きさも自動で変わる。
他のレイアウトを選んでも良いですが、今回はボーダーレイアウトを選んでみた。

```
// パネルを生成
JPanel panel = new JPanel();

// パネルサイズを設定
panel.setPreferredSize(new Dimension(800, 600));
// レイアウト設定
panel.setLayout(null);

// フレームにパネルを設定
frame.setContentPane(panel);
```

次にパネルを作っていく。
パネルを生成して、パネルサイズを設定し、フレームに載せていく。
パネルの大きさは、フレームのサイズで決まると先ほど書いたが、最初の設定のみ行う。なぜかは後ほど説明する。
レイアウトの設定は null としている。パネルについては、特別なレイアウトは設定しない。この設定でパネル内のコンポーネント（今回はラベル）を自分自身で配置する必要がある。もう少し言えば、自分自身で自由に配置できるということになる。

```
// ラベルを生成
JLabel label1 = new JLabel("Hello Java Game!!");

// フォントの設定
label1.setFont(new Font("ＭＳ ゴシック", Font.BOLD, 24));

// パネルにラベルを追加
panel.add(label1);

// ラベルの位置を設定
label1.setBounds(100, 100, 300, 30);
```

ラベルを追加し、パネル上に配置している。「Hello Java Game!!」と表示している部品だ。
１行目でラベルを生成し、表示する文字列を引数で渡している。
２行目（コメント行を除く）でフォントの設定を行っている。フォントは「ＭＳ　ゴシック」、BOLD を利用し、サイズは 24 にしている。（JavaDoc を見てみよう）
３行目でパネルに追加し、４行目でラベルを表示する位置と大きさを設定している。
４行目の引数の意味は、x, y, width, height となる。

```
// ラベルを生成
JLabel label2 = new JLabel("v(^_^)");

// フォントの設定
label2.setFont(new Font("ＭＳ ゴシック", Font.BOLD, 48));

// パネルにラベルを追加
panel.add(label2);

// ラベルの色を設定
label2.setForeground(new Color(255, 96, 0));

// ラベルの位置を設定
label2.setBounds(180, 180, 300, 50);
```

「v(^_^)」のマークを表示する部分になる。先ほどの処理と、ほとんど変わらないが、4つ目の処理が増えている。「setForeground」メソッドは、文字の色を決めるメソッドだ。「newColor」で３つの引数を与えている。これは、Red、Green、Blue（RGB）で、数字が大きくなるほど明るくなる設定となる。ここでは、赤色と緑を混ぜた色を作成している。

```
    // フレームを表示
    frame.setVisible(true);
    // サイズを最適化する
    frame.pack();

} // end HelloJavaGame
```

１行目で、フレームを表示する。同時に add してきた部品も全て表示されることになる。
2行目で表示されたフレームをサイズ変更している。ここでどの大きさにすればよいかを判断するために、パネルの大きさを利用している。パネルの大きさに合わせてフレームの大きさが調整されることになる。
３行目でコンストラクタは終了する。

このクラスは終わりとなる。
どうだっただろうか？インストール、コンパイル、プログラムまで見てきた。

プログラムは初めての人には難しいかもしれない。すぐに理解は難しいかもしれない。今は分からなくても、また、いつか戻ってきて、見直しをしてみてほしい。

JavaDoc を見ることに慣れて、ぜひとも新しいクラス、メソッドを使ってみてほしい。

また、本書のみによらず、良書を探すのが良いと思う。初心者向けの本を一冊読むといいだろう（本書のみで全てを紹介できるとは思ってはいない）。基本を別の本で理解しながら、それを実際に応用していく。習うのは別の良書、慣れるのは本書でやっていけば、身についていくと思っている。

わたしの著書「The Java」も出版させていただいた。
こちらも初心者向けに作成させていただいた。ぜひとも、検討いただければと思っています。

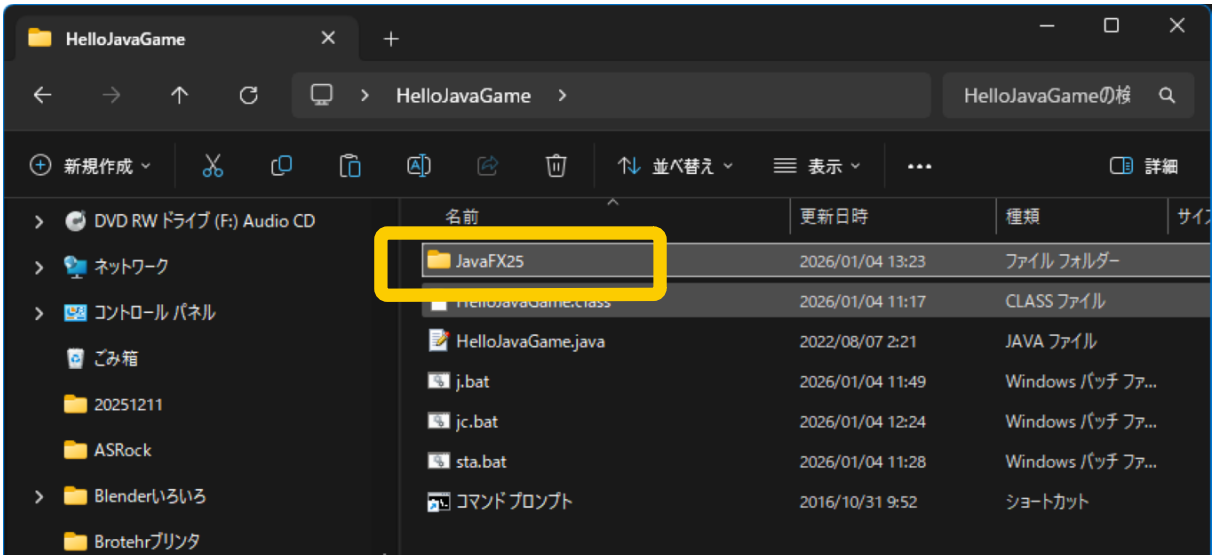
プチコラム ２

プログラムが出てきて、大変だったかもしれない。でも、どこかを修正したり、文字をちょっと変えてみたり・・・そうすれば、自分の作ったプログラムができる。
ぜひとも怖がらずに、次の一歩を進んでみよう。少しずつでいいから進んでいってみよう。
きっと、楽しくなること、間違いなしだ。

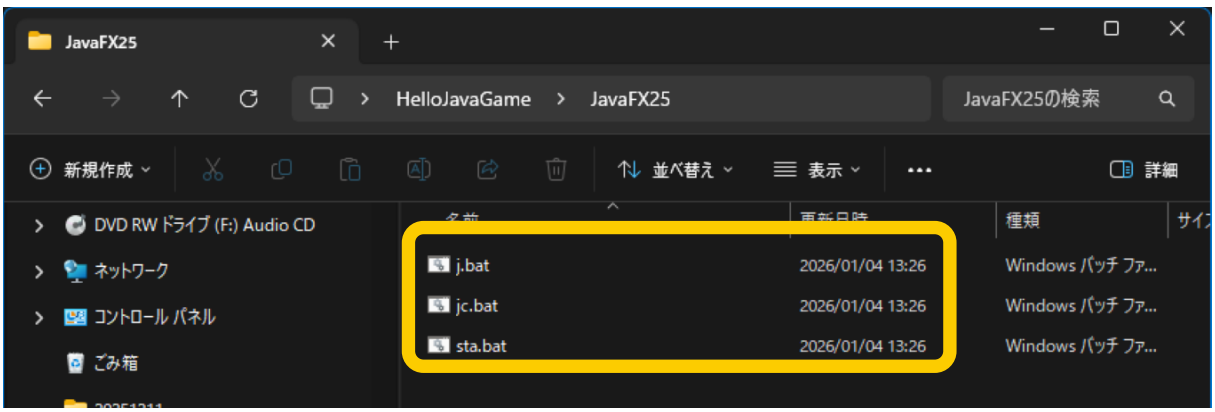
JavaFX を学ぶ方へ

パスを変更しよう

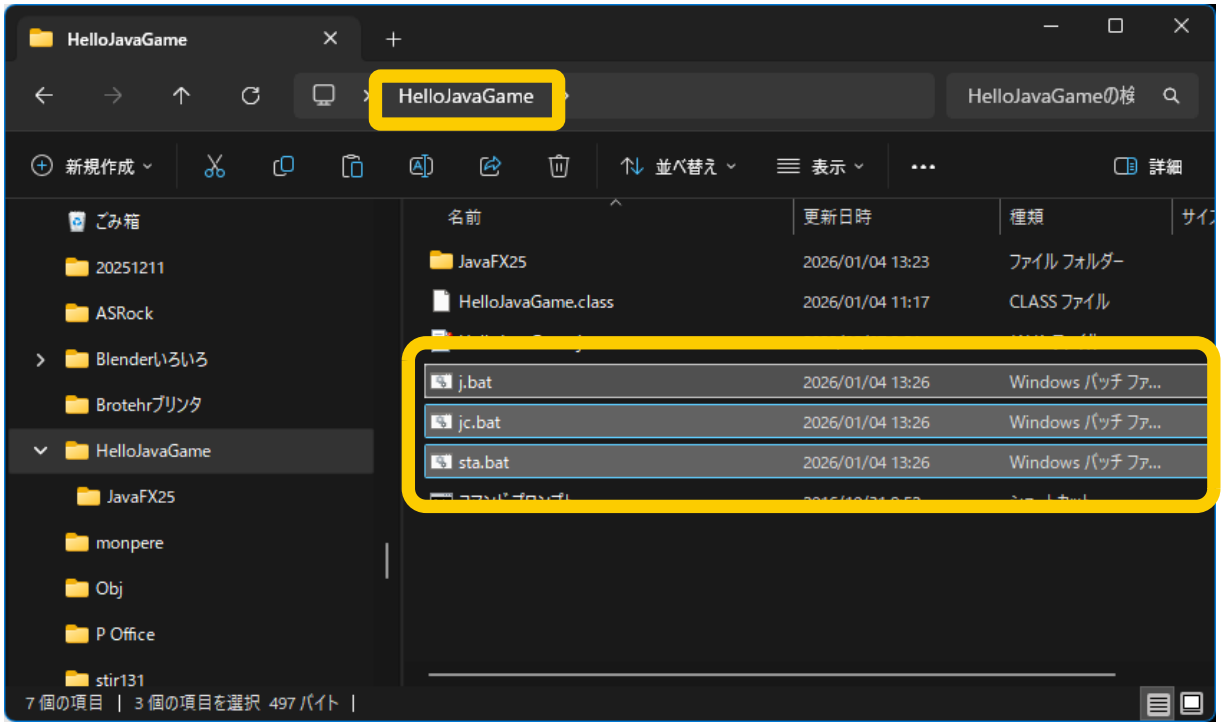
　新しく「JavaFX25」というフォルダを用意している。
JavaFX をコンパイル・実行する人は、その中身を見ていこう。
（Java11 から JavaFX は Java とは別梱包となっています。
　Java10 以前で JavaFX をコンパイル・実行する人は、この部分の反映は必要ありません）



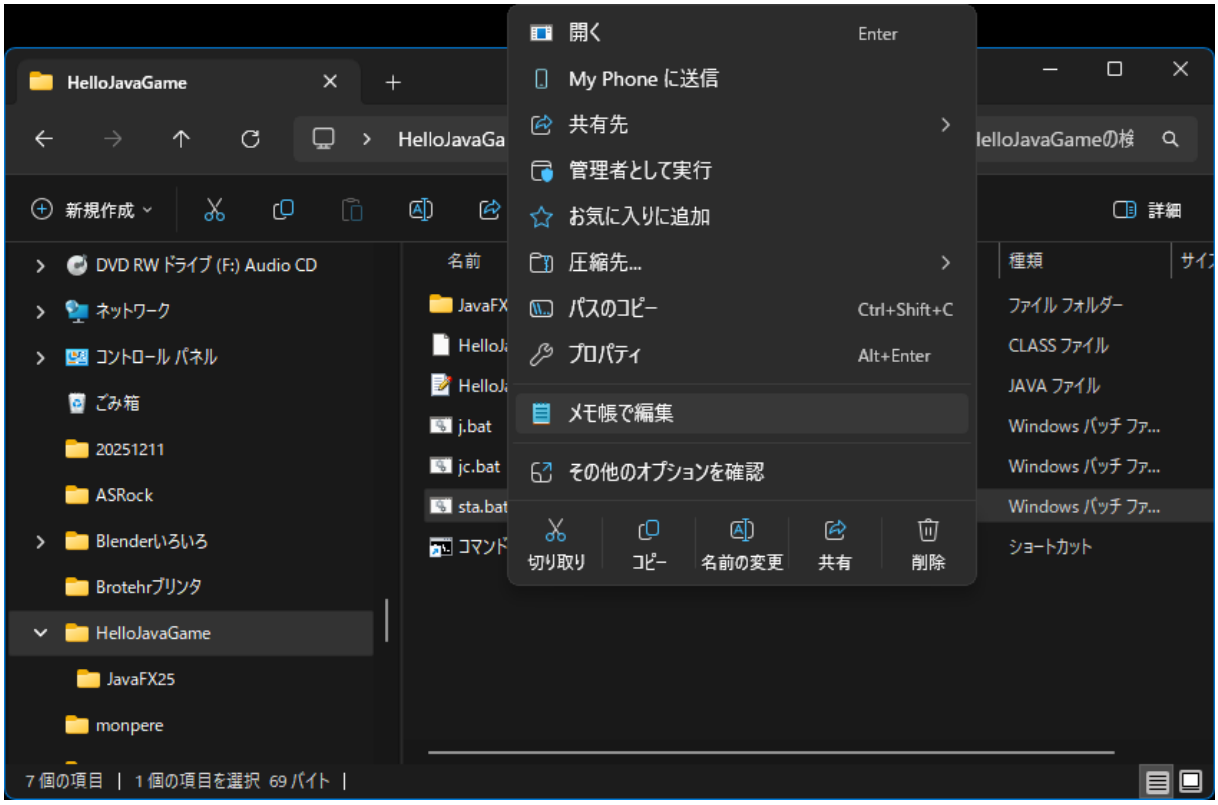
　「j.bat」「jc.bat」「sta.bat」の３つのファイルがある。
ひとつ上のフォルダへコピーしよう。



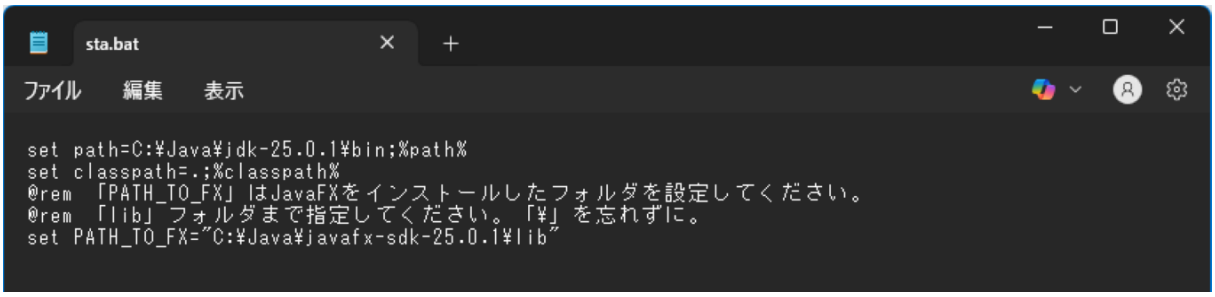
　上のフォルダにコピーしました。
更新日付が変わりました。



「sta.bat」を選択して、右クリックの「メモ帳で編集」をクリックしよう。
（Windows10 は編集かも）
メモ帳などで開くはずです。



すると、「sta.bat」の内容が表示される。



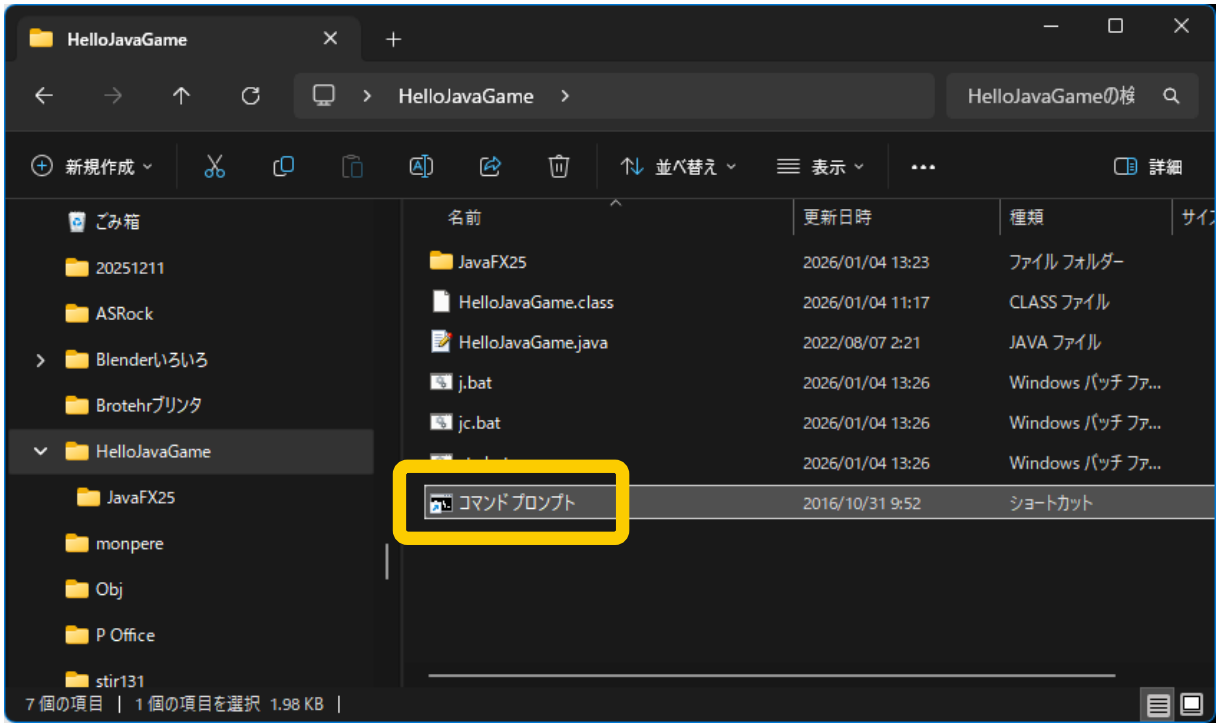
１行目と２行目の設定内容の詳細については、「Java」側で紹介しているので、そちらを見てください。

最終行は「JavaFX」のフォルダの「lib」フォルダを指定します。

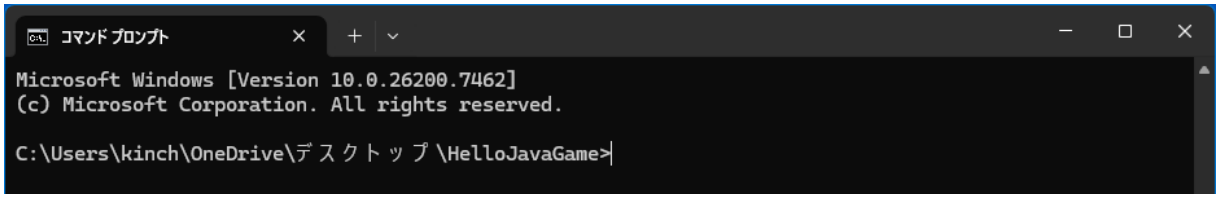
（JavaFX をインストールをした場合にしてください。していない場合は、JavaFX の部分は放っておいても良いし、削除しても良いです）

フォルダ名を合わせる。
間違えるとはまるので注意して設定する。コンパイル通らない、または実行できない状態になるので気を付けよう。
「JavaFX」は「%path%」の設定は必要はありません。「%path%」は、もともとの設定を表していて、「PATH_TO_FX」は、JavaFX 専用として ここで作成するパスとなるためだ。

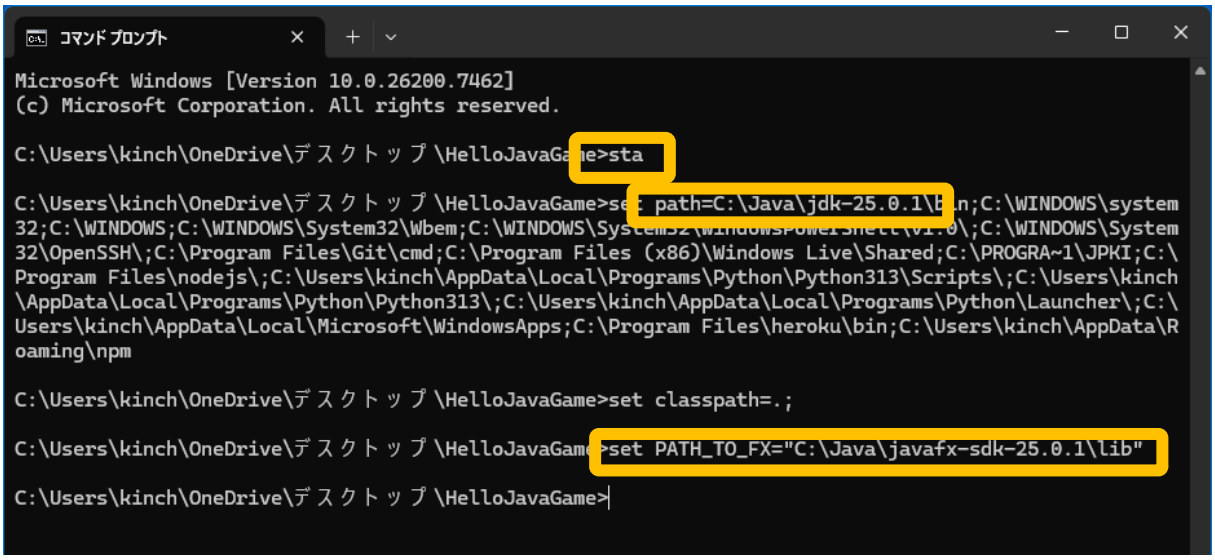
コマンドプロンプトを起動しよう。ダブルクリックだ。



コマンドプロンプトが起動される



「sta」と入力して、「Enter」キーを押してみよう。



実行された内容が表示されていますが、さきほどの Java の設定の時と同じように、「path」の方は、長い文の中に、先ほどのパスの内容が設定されてる。後ろに続いているパスの内容は、元々のパスとなっていて、くっつけている。

「PATH_TO_FX」は、そのままの設定がされているのが分かる。

これで JavaFX を利用できる環境は整った。

プチコラム ３

ここでは JavaFX の動作確認までは行わない。

続きは、「Java でゲームを作ろう３　JavaFX シューティングゲーム編」「Java でゲームを作ろう４ JavaFX ３Ｄゲーム編」で行ってほしいと思う。

１－４　利用ツールのご紹介

基本的に無料のツールばかり利用しています。君の慣れたツールがあれば、それを使えば構わない。もし、初めてやってみるような場合や、ツールを見比べたり、便利そうなツールがあるようであれば、その部分だけ使ったり、とかもありだ。

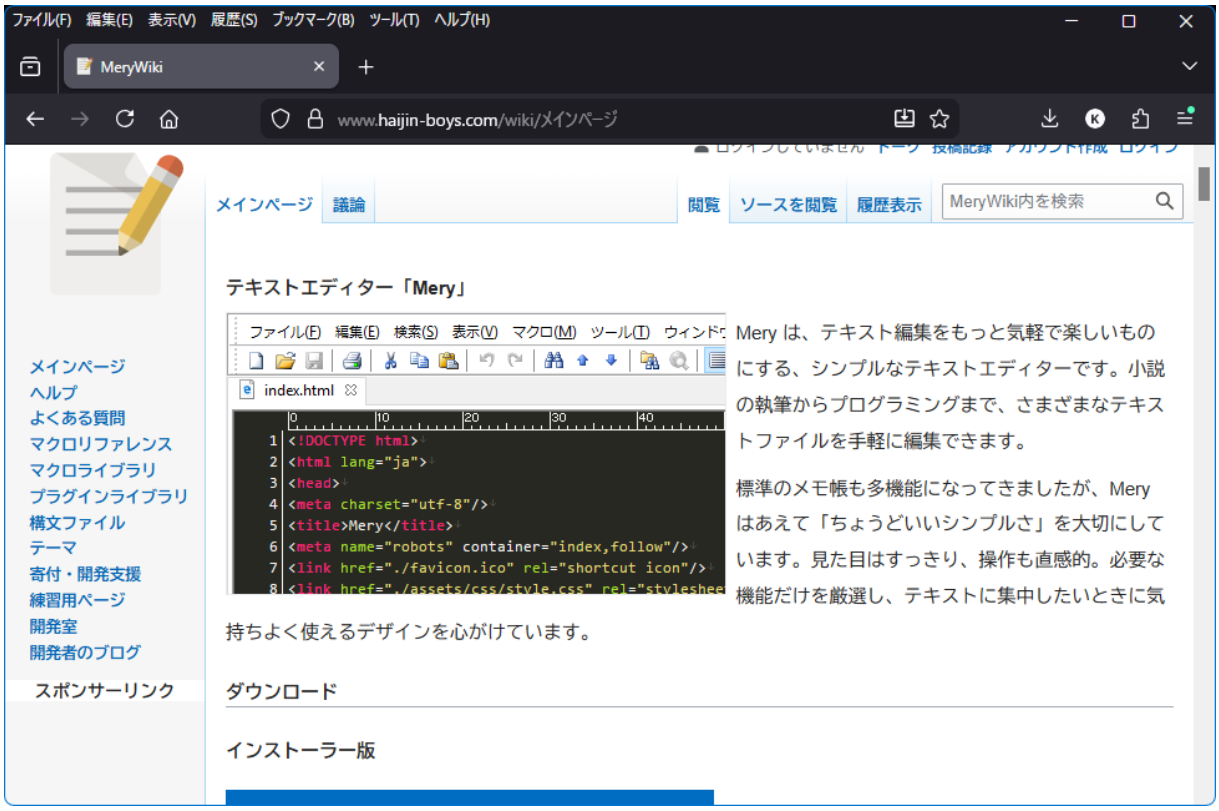
お金を極力かけずにやりたいという、わたしの思いもあるので、心配せず参考にしてほしい。

テキストエディタの紹介

まずはテキストエディタの紹介だ。Windows に添付されているメモ帳ではきついです。テキストエディタはいろいろとあるので、自分のあったものを探してください。

著者は今回の執筆を機会に、テキストエディタを変えてみた。
「Mery」というテキストエディタです。窓の杜や Vector でダウンロード可能のようです。
ホームページは
http://www.haijin-boys.com/wiki/メインページ
となっています。

ホームページ



他にもいろいろなツールがあるので（フリーでもいろいろとある）、自分に合ったエディタを見つけてみよう！

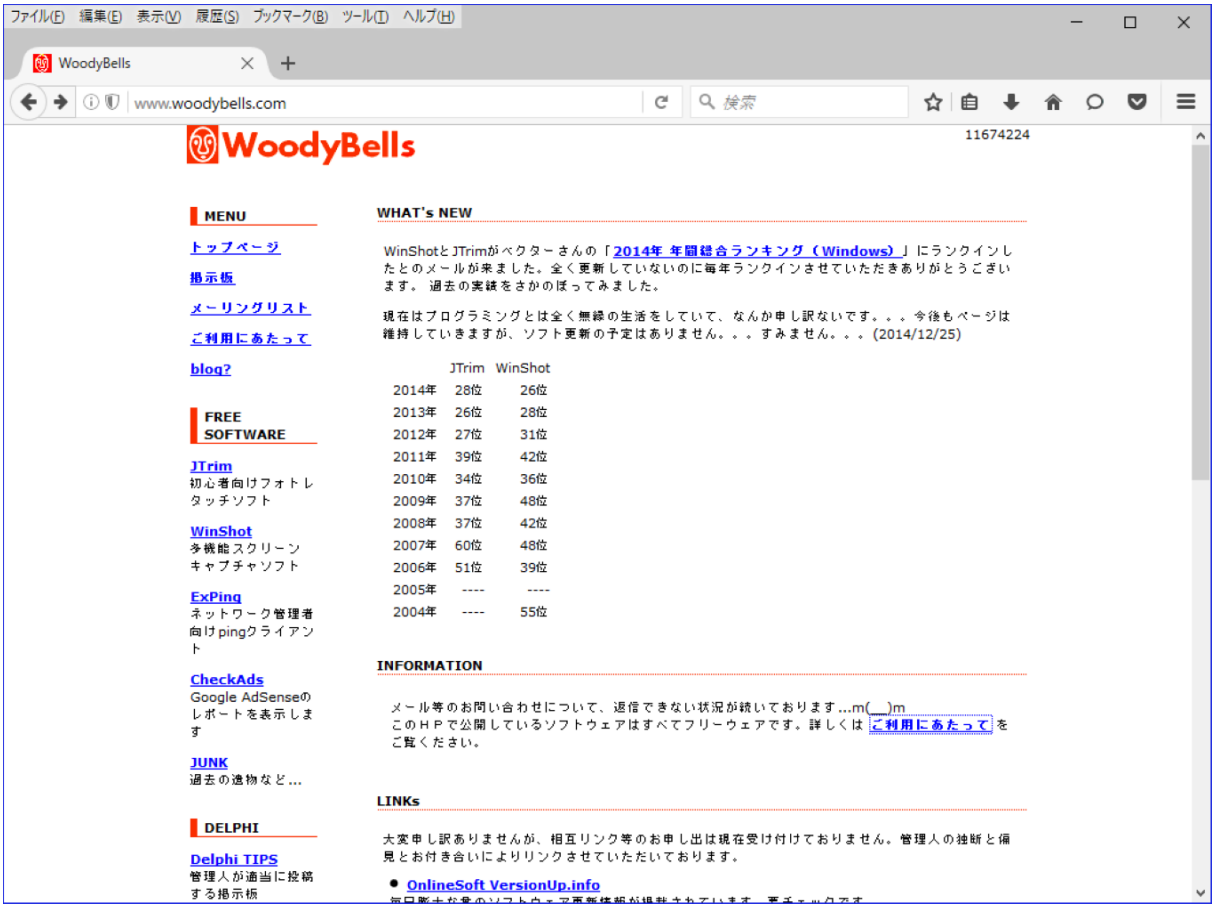
画像編集

画像編集は、Windows に付いているペイントと、JTrim というツールを利用しています。ペイントはご存知と思います。絵を書くときはペイントを利用しています。そして、ペイントでは出来ない部分は、JTrim を利用しています。JTrim は画像の一部を透明にすることができます。このことを利用して、画像を重ねた時に、後ろの画像や背景を透けて見せることができるようになります。

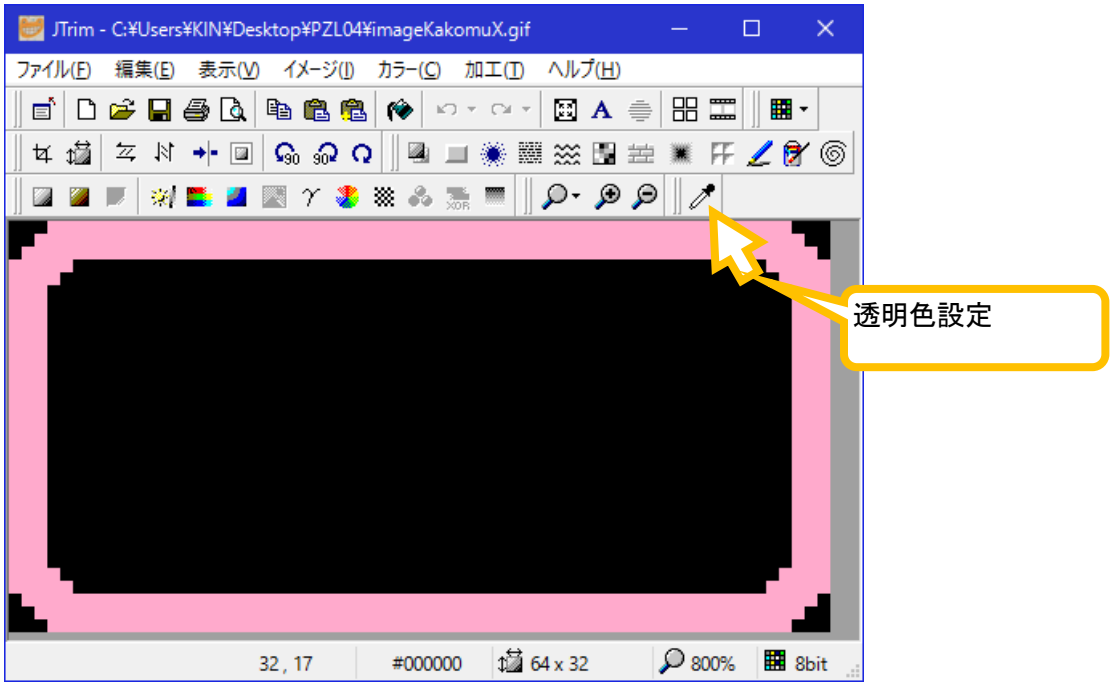
ホームページは
http://www.woodybells.com/
と、なっている。

なお、更新はストップしているようだが、Windows10/11 でも問題なく使えているようなので、そのまま利用させていただいています。

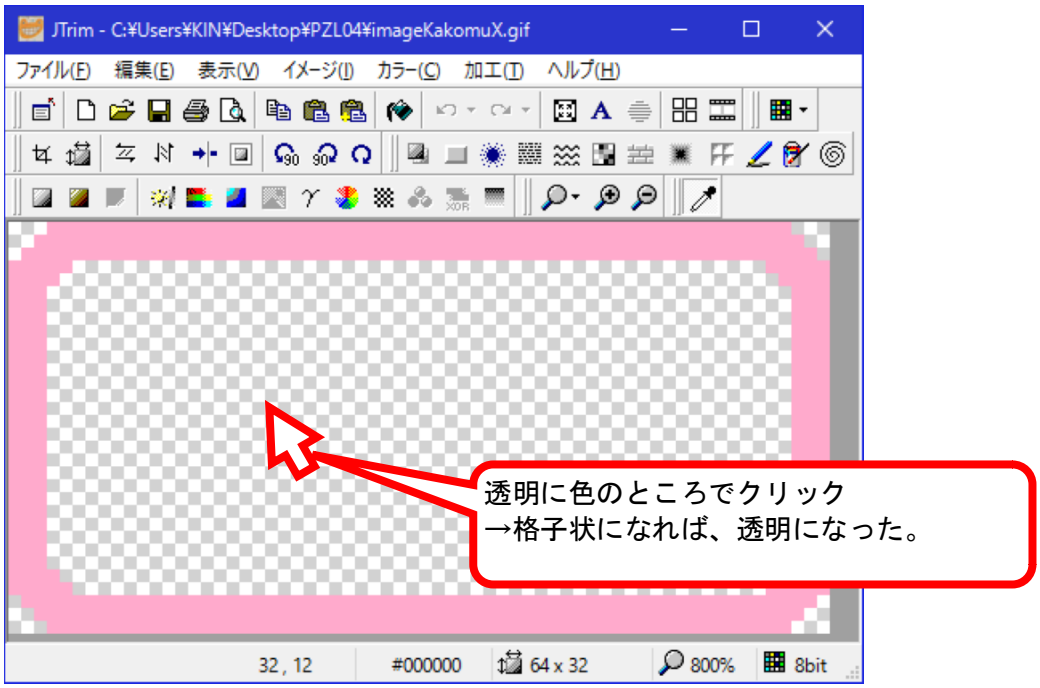
ホームページ



使い方は簡単です。
透明にしたい画像開き、「透過色設定」をクリックします。



透明にしたい部分をクリックすれば、色が白と灰色の格子状になります。



保存を忘れないようにしよう。
ちなみに、透過できる画像フォーマットは、gif と png のようです。

ここまでよく読んでくれた！
本編に進んでいこう！