

0. 授業の概要

- Java 言語によるプログラミングの演習を行う。
- 授業各回でレポート課題に取り組み, プログラム, 実行結果, 考察を manaba 上に提出する。
- 本演習で作成するプログラムやクラスファイルを適切に管理すること. 提出ファイルの相違は原則 0 点とする。
- 成績評価は, 各回の授業で提出されたレポート(100%)で行う. 当初, 予定していた到達度確認は実施しない。

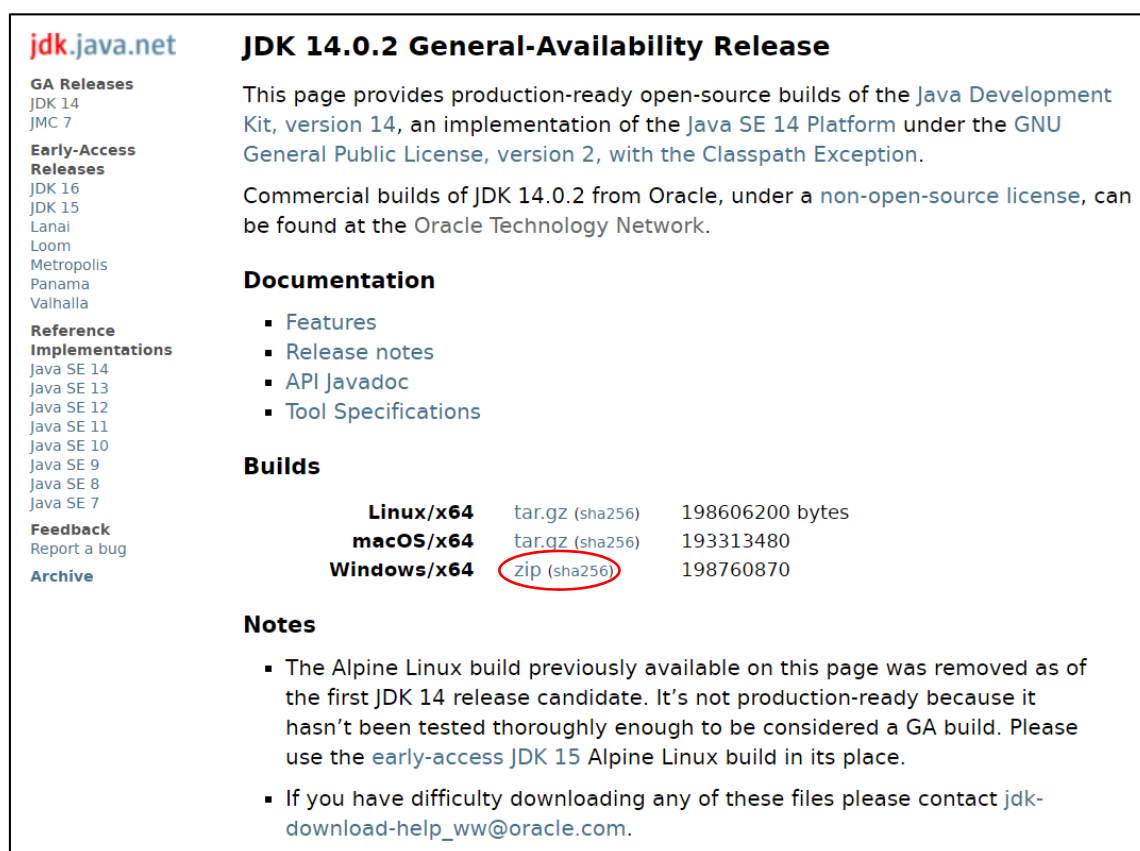
1. 開発環境の構築

本演習では Java 言語の開発環境として, OpenJDK と OpenJFX を利用する. 前期に開講された「基盤系オブジェクト指向プログラミング演習」にて OpenJDK をインストール済みであると思われるが, 改めて環境構築手順を示す. OpenJDK の動作に問題がなければ, (3) OpenJFX の入手まで飛ばして良い. OpenJFX は, JavaFX を利用するために導入する. JavaFX は GUI (Graphical User Interface) のアプリケーションを作成するためのライブラリである。

なお, 本資料はオペレーティングシステム(OS)として Windows10 を想定して作成されている. ほかの OS で環境を構築する場合は, 資料を適宜読み替えること。

(1) OpenJDK の入手

URL: <https://jdk.java.net/14/>



jdk.java.net

GA Releases
JDK 14
JMC 7

Early-Access Releases
JDK 16
JDK 15
Lanai
Loom
Metropolis
Panama
Valhalla

Reference Implementations
Java SE 14
Java SE 13
Java SE 12
Java SE 11
Java SE 10
Java SE 9
Java SE 8
Java SE 7

Feedback
Report a bug

Archive

JDK 14.0.2 General-Availability Release

This page provides production-ready open-source builds of the [Java Development Kit, version 14](#), an implementation of the [Java SE 14 Platform](#) under the [GNU General Public License, version 2](#), with the [Classpath Exception](#).

Commercial builds of JDK 14.0.2 from Oracle, under a [non-open-source license](#), can be found at the [Oracle Technology Network](#).

Documentation

- [Features](#)
- [Release notes](#)
- [API Javadoc](#)
- [Tool Specifications](#)

Builds

Linux/x64	tar.gz (sha256)	198606200 bytes
macOS/x64	tar.gz (sha256)	193313480
Windows/x64	zip (sha256)	198760870

Notes

- The Alpine Linux build previously available on this page was removed as of the first JDK 14 release candidate. It's not production-ready because it hasn't been tested thoroughly enough to be considered a GA build. Please use the [early-access JDK 15](#) Alpine Linux build in its place.
- If you have difficulty downloading any of these files please contact jdk-download-help_ww@oracle.com.

図1. OpenJDK のダウンロードサイト

(2) OpenJDK のインストール(Windows10 の場合)

- ① (1)でダウンロードしたファイル(openjdk-14.0.2_windows-x64_bin.zip)をデスクトップなどに解凍する
→解凍先に「jdk-14.0.2」というフォルダが作成される
- ② エクスプローラを起動し、Cドライブにある「Program Files」というフォルダを開く
- ③ 「Program Files」の下に「Java」というフォルダを作成する
- ④ ③で作成した Java フォルダの下に、①で生成された jdk-14.0.2 フォルダを移動させる
- ⑤ エクスプローラで「PC」を右クリックし、「プロパティ」を選択する
- ⑥ 開いた「システム」ウインドウ(図 2)の左側にある「システムの詳細設定」を選択する
- ⑦ 開いた「システムのプロパティ」ウインドウ(図 3)の下にある「環境変数」を選択する
- ⑧ 開いた「環境変数」ウインドウの下半分にある「システム環境変数」の中から「Path」を選択する
- ⑨ 開いた「環境変数名の編集」ウインドウ(図 4)の右上にある「新規」ボタンを押し、
「C:¥Program Files¥Java¥jdk-14.0.2¥bin」を入力する
- ⑩ 「OK」ボタンを何度か押して、全てのウインドウを閉じる

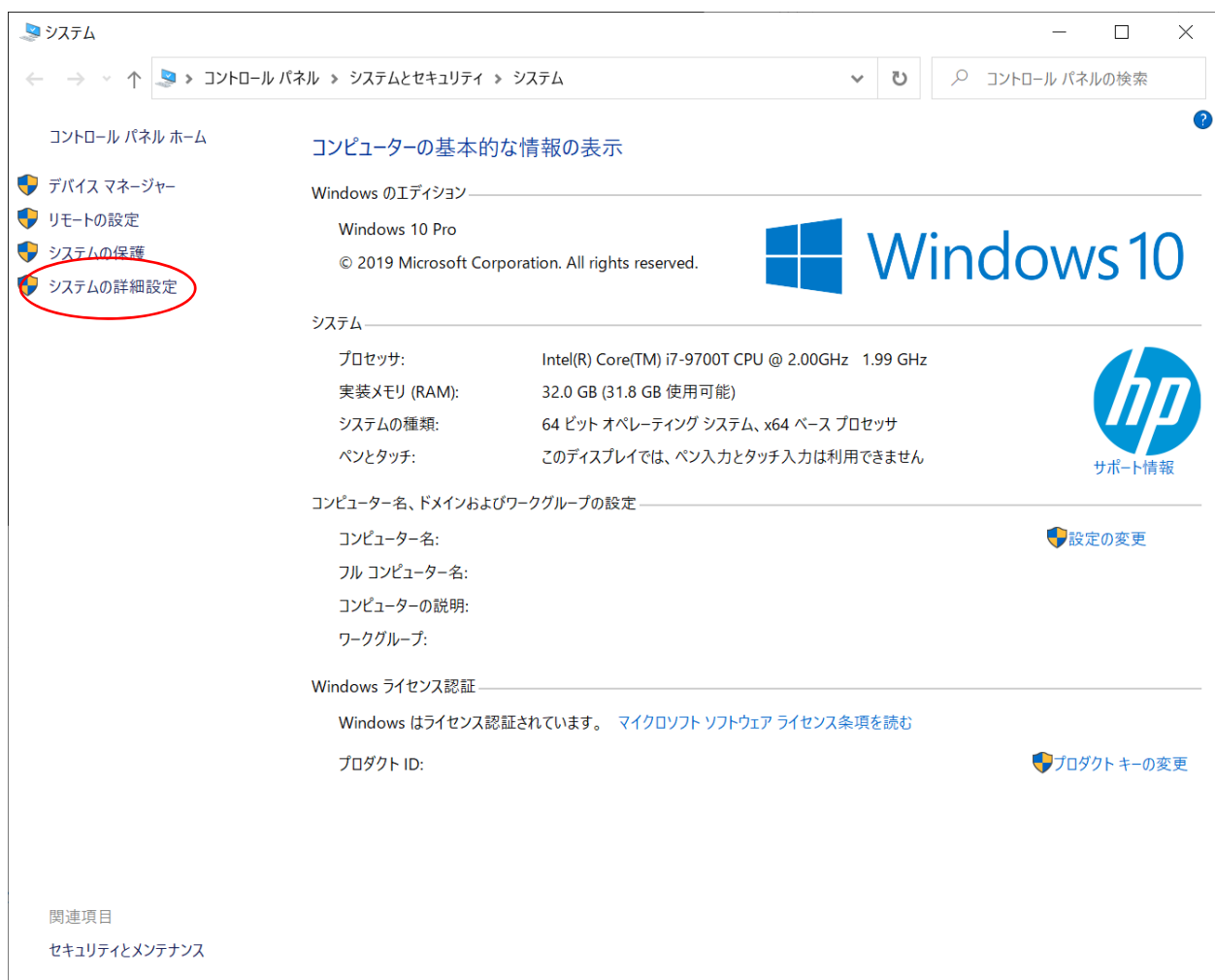


図 2. システムウインドウ



図 3. システムのプロパティウインドウ

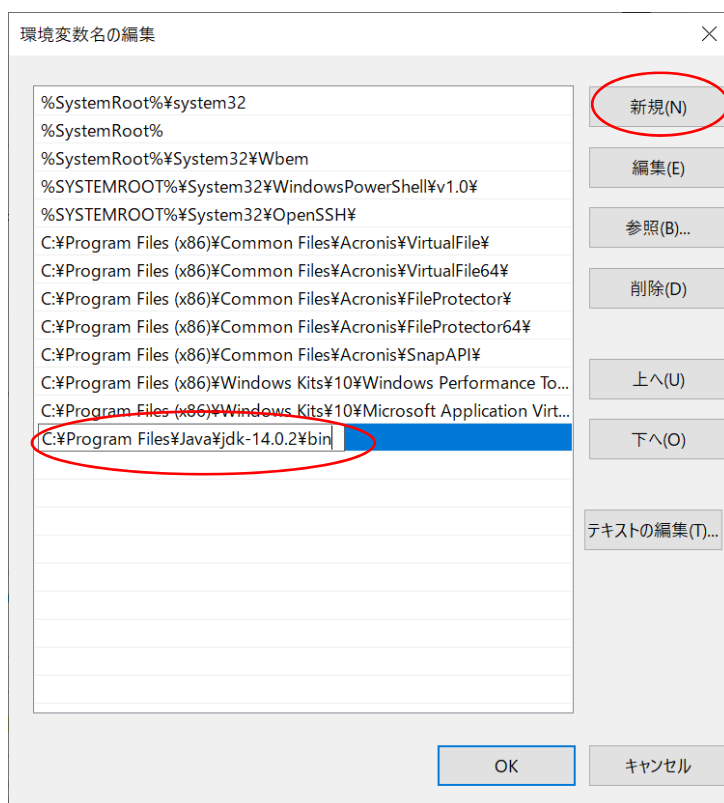
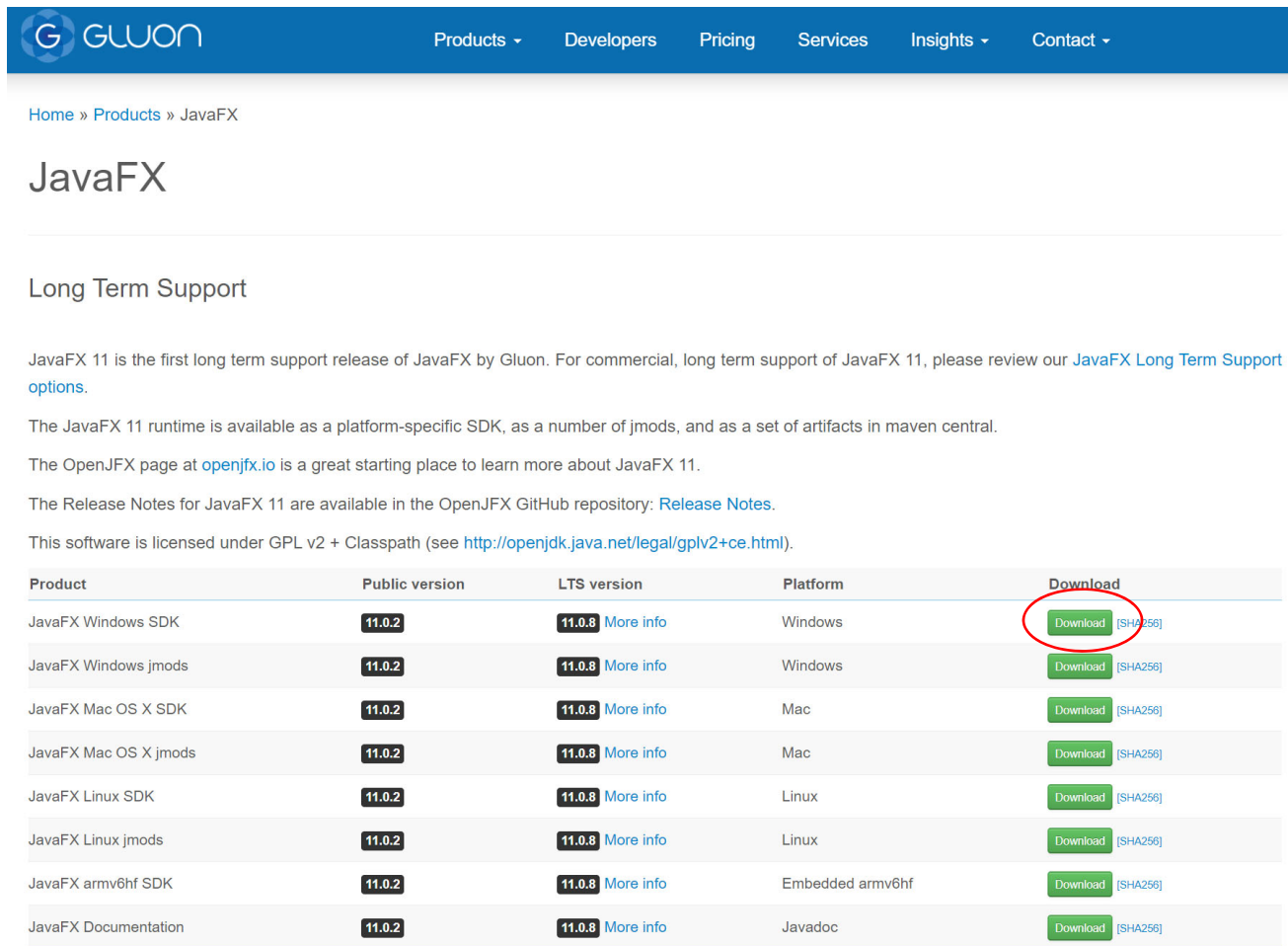


図 4. 環境変数名の編集ウインドウ

(3) OpenJFX の入手

URL: <https://gluonhq.com/products/javafx/>



The screenshot shows the Gluon JavaFX website. The navigation bar includes links for Products, Developers, Pricing, Services, Insights, and Contact. The main heading is 'JavaFX'. Below it, there is a section for 'Long Term Support' with text explaining that JavaFX 11 is the first long-term support release. It mentions that the runtime is available as a platform-specific SDK, jmods, or Maven artifacts. It also points to the OpenJFX page at openjfx.io and the Release Notes for JavaFX 11. At the bottom, there is a table with download links for various platforms.

Product	Public version	LTS version	Platform	Download
JavaFX Windows SDK	11.0.2	11.0.8 More info	Windows	Download [SHA256]
JavaFX Windows jmods	11.0.2	11.0.8 More info	Windows	Download [SHA256]
JavaFX Mac OS X SDK	11.0.2	11.0.8 More info	Mac	Download [SHA256]
JavaFX Mac OS X jmods	11.0.2	11.0.8 More info	Mac	Download [SHA256]
JavaFX Linux SDK	11.0.2	11.0.8 More info	Linux	Download [SHA256]
JavaFX Linux jmods	11.0.2	11.0.8 More info	Linux	Download [SHA256]
JavaFX armv6hf SDK	11.0.2	11.0.8 More info	Embedded armv6hf	Download [SHA256]
JavaFX Documentation	11.0.2	11.0.8 More info	Javadoc	Download [SHA256]

図 5. OpenJFX のダウンロードサイト

(4) OpenJFX のインストール (Windows10 の場合)

- ① (3)でダウンロードしたファイル(openjfx-11.0.2_windows-x64_bin-sdk.zip)をデスクトップなどに解凍する
→解凍先に「javafx-sdk-11.0.2」というフォルダが作成される
- ② (2)と同様に、①で生成されたフォルダを「C:\Program Files\Java」の下に移動させる
- ③ さらに、(2)と同様に、「環境変数」ウインドウを開き、上半分にある「(ログインユーザ名)のユーザー環境変数」にある「新規」ボタンを押す
- ④ 開いた「新しいユーザー変数」ウインドウ(図 6)の「変数名」に「JAVA_FX」(JAVA と FX の間にアンダーバーがあることに注意！)、変数値に「C:\Program Files\Java\javafx-sdk-11.0.2\lib」を入力し、「OK」ボタンを押す

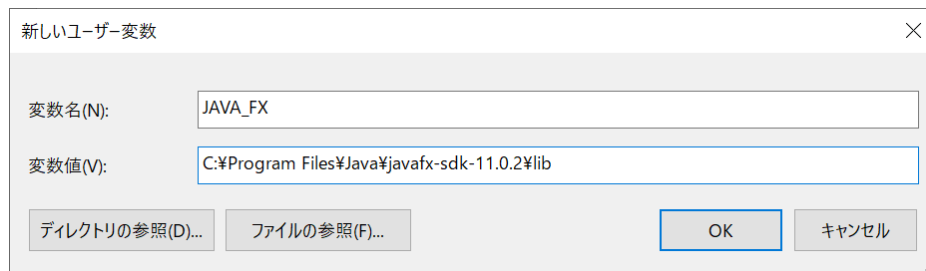


図 6. 新しいユーザー変数ウィンドウ

(5) OpenJDK の確認

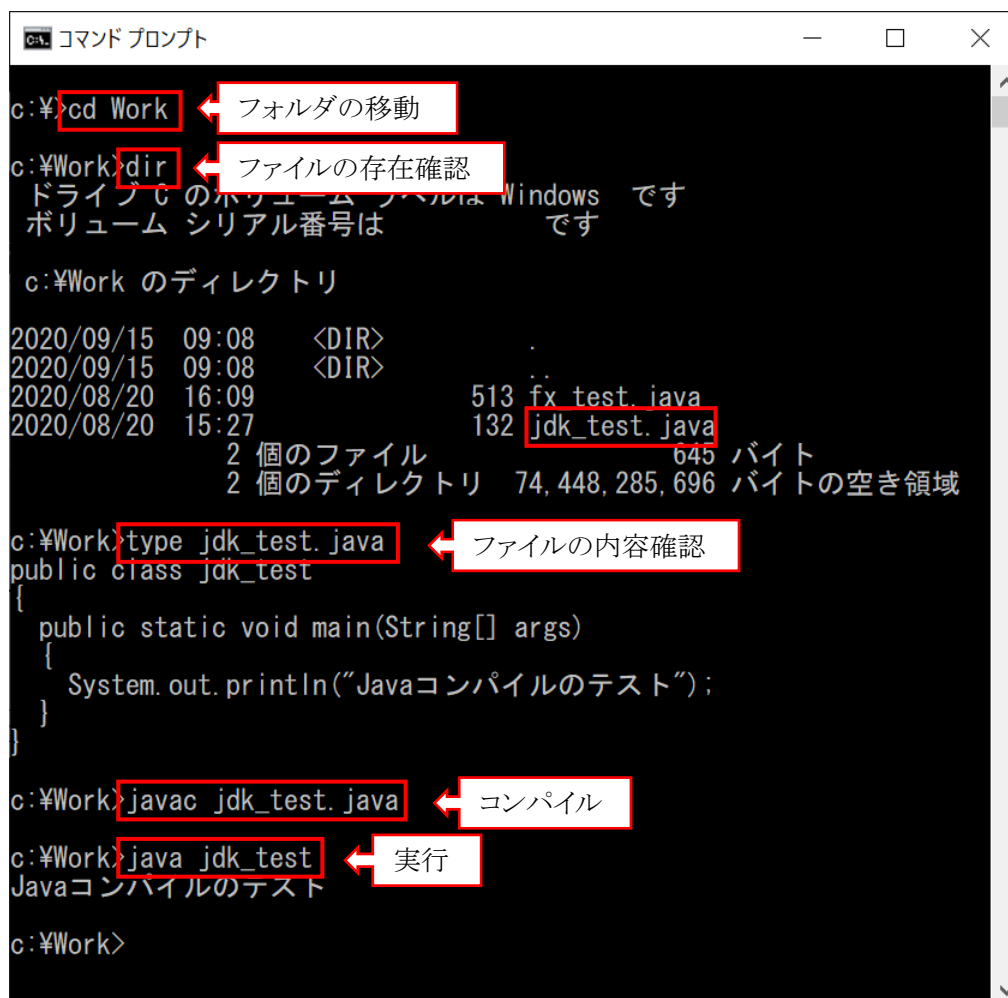
以下のプログラムをテキストエディタ等で実装し、コマンドプロンプト上でコンパイルと実行せよ。

```

1: public class jdk_test {
2:   static public void main(String[] av){
3:     System.out.println("Java コンパイルのテスト");
4:   }
5: }

```

コンパイル方法: `javac jdk_test.java` 実行方法: `java jdk_test`



(6) OpenJFX の確認

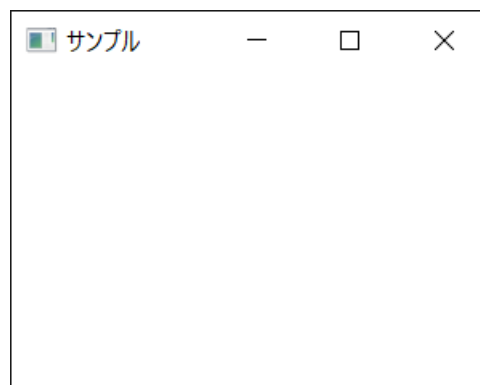
以下のプログラムをテキストエディタ等で実装し、コマンドプロンプト上でコンパイルと実行せよ。

```
1: import javafx.application.*;
2: import javafx.stage.*;
3: import javafx.scene.*;
4: import javafx.scene.control.*;
5: import javafx.scene.layout.*;
6:
7: public class fx_test extends Application
8: {
9:     public static void main(String[] args)
10:    {
11:        launch(args);
12:    }
13:    public void start(Stage stage)throws Exception
14:    {
15:        BorderPane bp = new BorderPane();
16:
17:        Scene sc = new Scene(bp, 300, 200);
18:
19:        stage.setScene(sc);
20:        stage.setTitle("サンプル");
21:        stage.show();
22:    }
23: }
```

コンパイル方法: `javac -p "%JAVA_FX%" --add-modules javafx.controls fx_test.java`

実行方法: `java -p "%JAVA_FX%" --add-modules javafx.controls fx_test`

実行結果の例:



<補足>

オプション `-p` で JavaFX のインストール場所を指定している。 `%JAVA_FX%` は (4) で設定した環境変数である。ダブルクォーテーションで囲むのは、環境変数に空白が含まれているためである。また、オプション `--add-modules` で JavaFX のモジュールを指定している。一般的な JavaFX プログラムでは、ここで示した `javafx.controls` モジュールを指定する。ハイフン、パーセント、半角空白などに注意して入力すること。

```

c:\>cd Work
c:\Work>dir
c:\Work>type fx_test.java
javac -p "%JAVA_FX%" --add-modules javafx.controls fx_test.java
java -p "%JAVA_FX%" --add-modules javafx.controls fx_test

```

Labels in the image:

- フォルダの移動 (Folder move) - points to `cd Work`
- ファイルの存在確認 (File existence check) - points to `dir`
- ファイルの内容確認 (File content check) - points to `type fx_test.java`
- コンパイル (Compile) - points to `javac`
- 実行 (Execute) - points to `java`

コンパイルおよび実行の際、オプションの指定が面倒であれば、事前に

- ```

1: @echo off
2: javac -p "%JAVA_FX%" --add-modules javafx.controls %1

```

と記述された(バッチ)ファイルを作成し、`javac_fx.bat` という名前(本来はファイル名は任意)で環境変数 `Path` が設定されているフォルダに保存しておけば、コマンドの指定を省略できる。

```
C:\> コマンド プロンプト

c:\¥Work> dir
ドライブ C のボリューム ラベルは Windows です
ボリューム シリアル番号は です

c:\¥Work のディレクトリ

2020/09/15 09:38 <DIR> .
2020/09/15 09:38 <DIR> ..
2020/08/20 16:09 513 fx_test.java
2020/09/15 09:23 66 javac_fx.bat
2020/09/15 09:23 65 java_fx.bat
2020/08/20 15:27 132 jdk_test.java
 4 個のファイル 776 バイト
 2 個のディレクトリ 73,746,665,472 バイトの空き領域

c:\¥Work> type javac_fx.bat
@echo off
javac -p "%JAVA_FX%" --add-modules javafx.controls %1

c:\¥Work> javac_fx fx_test.java

c:\¥Work> dir
ドライブ C のボリューム ラベルは Windows です
ボリューム シリアル番号は です

c:\¥Work のディレクトリ

2020/09/15 09:38 <DIR> .
2020/09/15 09:38 <DIR> ..
2020/09/15 09:38 752 fx_test.class
2020/08/20 16:09 513 fx_test.java
2020/09/15 09:23 66 javac_fx.bat
2020/09/15 09:23 65 java_fx.bat
2020/08/20 15:27 132 jdk_test.java
 5 個のファイル 1,528 バイト
 2 個のディレクトリ 73,745,539,072 バイトの空き領域

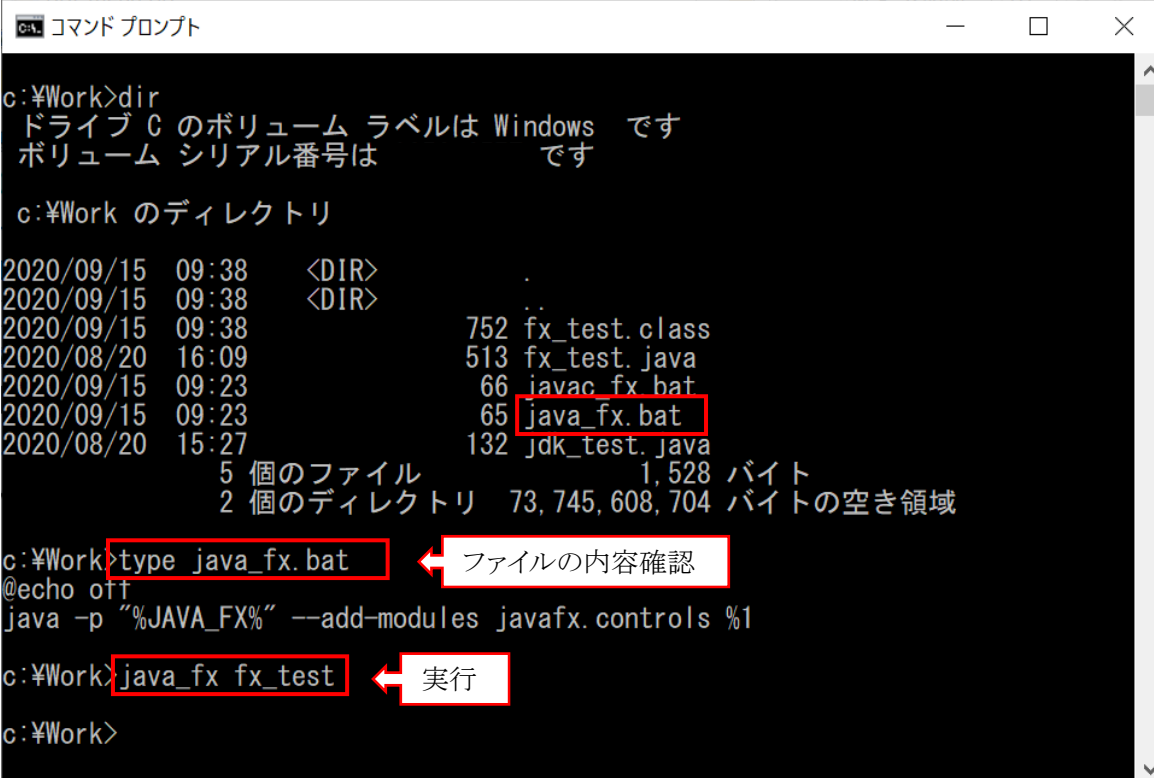
c:\¥Work>
```

実行も同様に,

```
1: @echo off
2: java -p "%JAVA_FX%" --add-modules javafx.controls %1
```

と記述されたファイルを作成し, `java_fx.bat` という名前で環境変数 `Path` が設定されているフォルダに保存しておけばよい.





```
C:\> コマンド プロンプト

c:\¥Work>dir
ドライブ C のボリューム ラベルは Windows です
ボリューム シリアル番号は です

c:\¥Work のディレクトリ

2020/09/15 09:38 <DIR> .
2020/09/15 09:38 <DIR> ..
2020/09/15 09:38 752 fx_test.class
2020/08/20 16:09 513 fx_test.java
2020/09/15 09:23 66 javac_fx.bat
2020/09/15 09:23 65 java_fx.bat
2020/08/20 15:27 132 jdk_test.java
 5 個のファイル 1,528 バイト
 2 個のディレクトリ 73,745,608,704 バイトの空き領域

c:\¥Work>type java_fx.bat ← ファイルの内容確認
@echo off
java -p "%JAVA_FX%" --add-modules javafx.controls %1

c:\¥Work>java_fx fx_test ← 実行
c:\¥Work>
```

【課題 1-1】(6)で実装したプログラムを実行した際のスクリーンショットを画像ファイル(png, jpeg, gif, bmp もしくは pdf のいずれかの形式)として, レポート管理システムに提出せよ.