

並列プログラミング Parallel Programming

2022 2Q

演習 第2回

情報理工学院 情報工学系

本日の流れ

- 課題内容の説明
- 演習に取り組む

● 目的

- キーボードイベント駆動プログラムを作成する
- Java の復習

● 題材

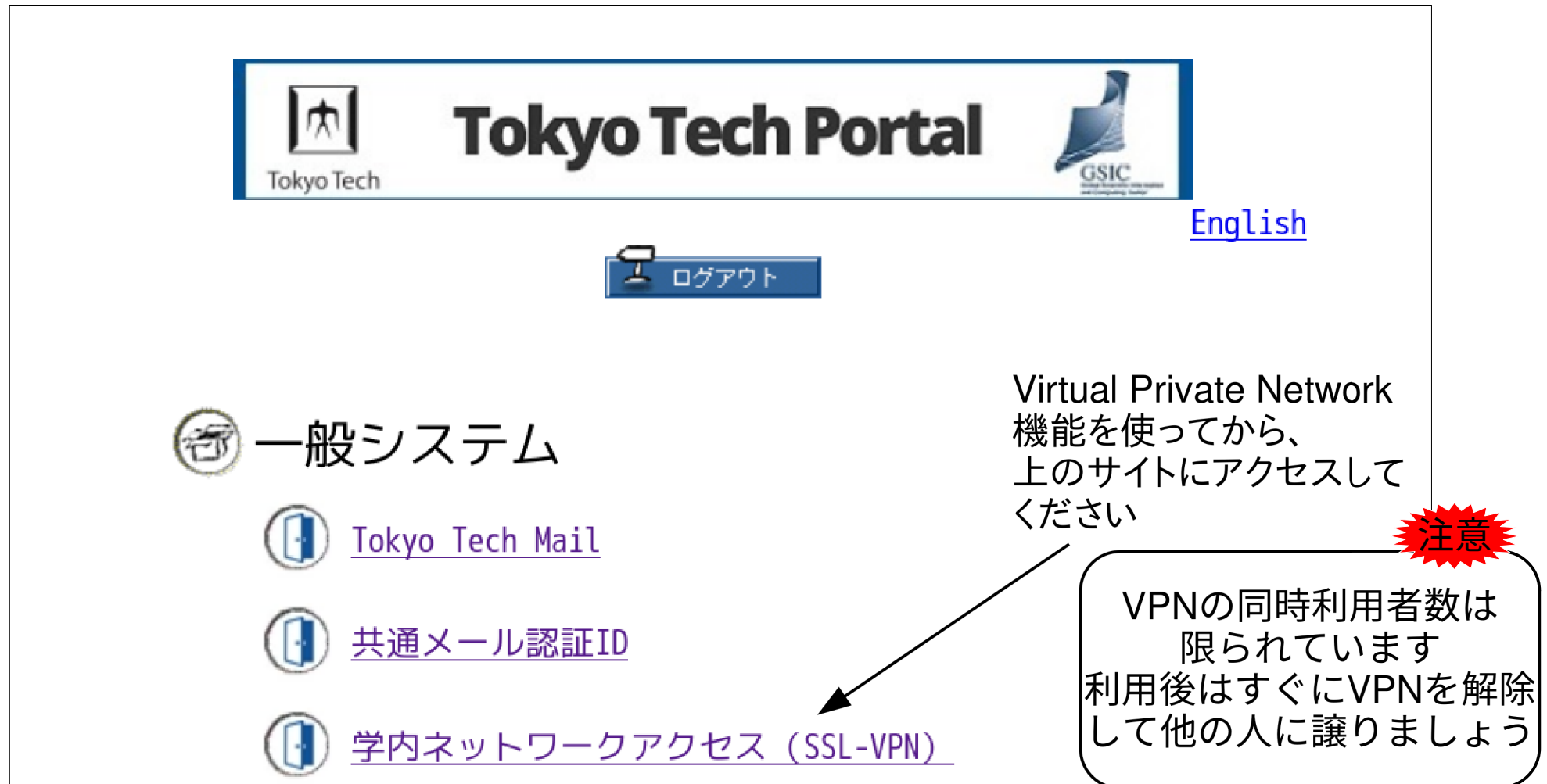
- 電卓プログラム
- 図形表示プログラム

課題のダウンロード

以下からダウンロードしてください

学内アクセス

● www.img.cs.titech.ac.jp/lecture/para/



The screenshot shows the Tokyo Tech Portal homepage. At the top, there is a header with the Tokyo Tech logo, the text "Tokyo Tech Portal", and the GSIC logo. Below the header, there is a "ログアウト" (Logout) button. On the left side, there is a "一般システム" (General System) section with three links: "Tokyo Tech Mail", "共通メール認証ID", and "学内ネットワークアクセス (SSL-VPN)". An arrow points from the "学内ネットワークアクセス (SSL-VPN)" link to a callout box on the right. The callout box contains the text "VPNの同時利用者数は限られています 利用後はすぐにVPNを解除して他の人に譲りましょう" (The number of simultaneous VPN users is limited. After use, please immediately cancel the VPN and let others use it). A red starburst with the word "注意" (Attention) is next to the callout box. Above the callout box, there is text that says "Virtual Private Network 機能を使ってから、上のサイトにアクセスしてください" (Please use the Virtual Private Network function to access the site above).

Tokyo Tech Portal

English

ログアウト

一般システム

[Tokyo Tech Mail](#)

[共通メール認証ID](#)

[学内ネットワークアクセス \(SSL-VPN\)](#)

Virtual Private Network
機能を使ってから、
上のサイトにアクセスして
ください

注意

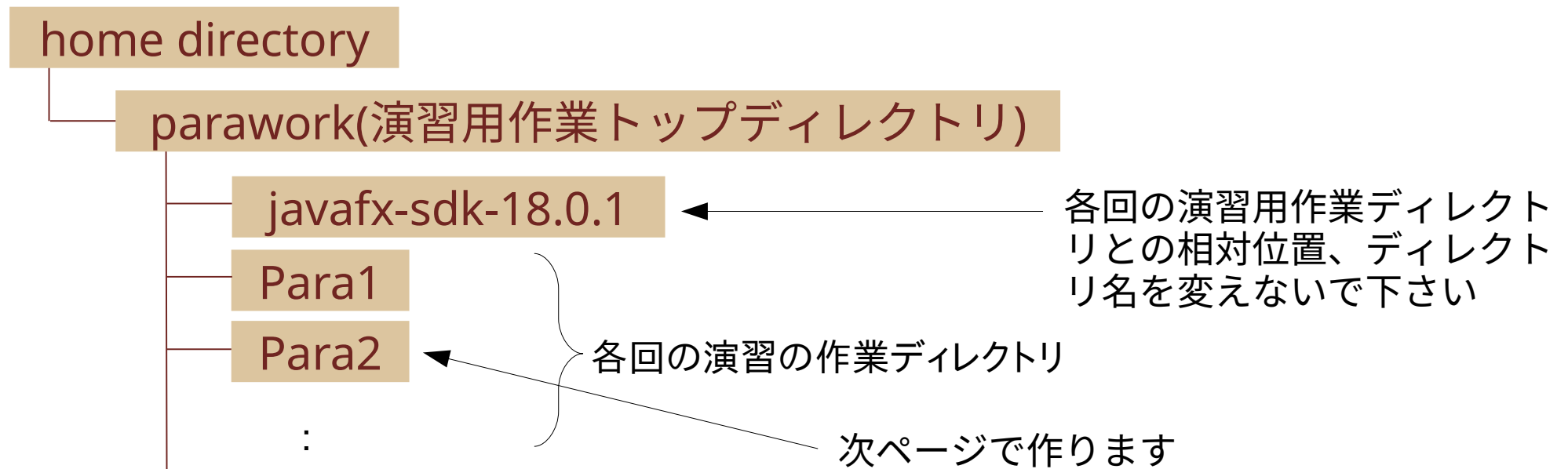
VPNの同時利用者数は
限られています
利用後はすぐにVPNを解除
して他の人に譲りましょう

準備 (0)

● 今後の演習回共通

- 演習用作業ディレクトリの展開位置の確認 Para1 の隣
- java17 インストール確認 AdoptiumOpenJDK17
- JavaFX17 以降の sdk のインストール確認

● 解凍後のディレクトリ



準備 (1)

- ダウンロードした **para2.zip** を展開する

```
unzip para2.zip
```

- 解凍後のディレクトリ

home directory

parawork(演習用作業トップディレクトリ)

javafx-sdk-18.0.1

Para1

Para2

Makefile (後述)

README 説明文

sweet.png

bin

クラスファイル (*.class) が格納される (解凍直後は空)

javadoc

ドキュメントが格納される (解凍直後は空)

src

ソースファイル

lib

実行に必要なjavaのライブラリ集(jarファイル) (今回は空)

準備 (2)

- Para2 以下の `._*` ファイルを再帰的に消去

```
cd Para2  
make cleanall
```

- MacOS では、`._` で始まるファイルが作成されることがありますが、コンパイル時の障害になるので消去

準備 (3)

- ソースファイルを **javac** コマンドでコンパイルしてクラスファイルを作る

今回の演習では Para2 ディレクトリで

Calculatorが他のクラスに依存する場合、順次コンパイルしてくれる

```
javac -d bin -encoding UTF-8 --module-path ../javafx-sdk-18.0.1/lib
--add-modules javafx.controls,javafx.swing -sourcepath src -classpath
lib/*: src/para/calc/Calculator.java
```

として下さい

Windowsでは:を;に置換

実際は一行で書く

-d bin	コンパイル後のクラスファイルをディレクトリ bin に置く
-encoding UTF-8	ソースファイルの文字コードが UTF-8 であることを示す
-sourcepath src	ソースファイルがディレクトリ src 以下にあることをコンパイラに教える
--module-path	モジュールファイルがあるディレクトリをコンパイラに教える
--add-modules	依存するモジュールを列挙する
-classpath	依存するクラスやライブラリのありかをコンパイラに教える

※ <https://docs.oracle.com/javase/jp/17/docs/specs/man/javac.html>
でその他のオプションを確認すること

準備 (4)

● java コマンドでクラスファイルを実行する

- 今回の演習では Para2 ディレクトリにて

```
java --module-path ../javafx-sdk-18.0.1/lib --add-modules  
javafx.controls,javafx.swing -cp bin:lib/*: para.calc.Calculator
```

パッケージ名 起点となるクラスの名前

として下さい (デモ用プログラムは para.calc.Calculator と para.Main0? があります)

--module-path javacのオプションと同じ役割

--add-modules javacのオプションと同じ役割

-cp bin:lib/*:

-cp は -classpathの短縮形

実行に必要なコンパイル済みクラスファイルがディレクトリbin 以下に置かれていること、標準以外のjavaライブラリファイル(jarファイル)がlib/に置かれていることを java コマンドに教える

※<https://docs.oracle.com/javase/jp/17/docs/specs/man/java.html>
でその他のオプションを確認すること

準備 (5)

- javadoc コマンドでソースファイルのコメント文から HTML のドキュメントファイルをつくる

```
package para.paint;

import javafx.application.Application;

/** JavaFXで作成するお絵描きプログラム. */
public class Paint extends Application
{
    /** 描画領域. */
    Canvas canvas;
```

HTML文書の出力先
ディレクトリ

実際は一行で
書く

今回の演習では、Para2 直下

```
javadoc -html5 -charset utf-8 -encoding UTF-8 -d javadoc -sourcepath src
--module-path ../javafx-sdk-18.0.1/lib --add-modules
javafx.controls,javafx.swing
-link https://docs.oracle.com/javase/jp/17/docs/api
-link https://openjfx.io/javadoc/18
-package para.calc para para.graphic.shape para.graphic.target
para.graphic.parser
```

外部javadocのURI
リンクを作成する
には必要

パッケージ名
としてください

※<https://docs.oracle.com/javase/jp/17/docs/specs/man/javadoc.html>
でその他のオプションを確認すること

準備 (6)

利用前にMakefile内に一ヶ所
jdkのインストールされている
ディレクトリの場所の記述は
各自で編集

● コマンドをいちいちタイプするのが面倒 ...

今回は Makefile を用意したので make コマンドで
javac , javadoc の実行が簡単に行える

make Calculator	Calculatorについてコンパイルして、実行
make Main00	Main00 をコンパイルして、実行
make Main01	Main01 をコンパイルして、実行
make Main02	Main02 をコンパイルして、実行
make Main03	Main03 をコンパイルして、実行
make clean	bin 以下のクラスファイルをすべて削除
make doc	javadoc コマンドを実行

ソースコードを更新してもmakeが
感知しない場合
があるので、コード
を書き換えても結
果に変化がない場
合は、一度
make clean
して再コンパイル
して下さい

上を実行すると実際に発行されたコマンドが表示される
Makefile を自分好みに変更してよいです

Makefileの記述ではタブ\tは意味があります。スペースで置き換え
ると、makeが正しく解釈できません。
Makefileの書き方は各自調べて下さい

準備 (6.1)

🍪 Makefile の編集

```
#####
## if you want to specify a directory java installed explicitly,
## set the java binary directory here
#####
JAVABIN=
# JAVABIN=$(HOME)/parawork/jdk-17.0.3/bin/
# JAVABIN=/usr/lib/jvm/default-java/bin/

#####
## set javafx-sdk directory your installed
#####
JAVAFXMODULE=../javafx-sdk-18.0.1/lib

#####
## set proxy server address and port number, if your machine connects
## to the internet via proxy
JAVADOCPROXY=
# JAVADOCPROXY=-J-Dhttp.proxyHost=proxy.csc.titech.ac.jp -J-Dhttp.proxyPort=8080

#####
## csc room setting
#####
#JAVABIN=/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/
#JAVAFXMODULE=/Library/Java/JavaFX/javafx-sdk-17.0.2/lib

# OS-dependent commands and separator
ifeq ($(OS),Windows_NT)
    SEP=;
    RM=rd /s /q
    FIND=echo
else
    SEP=:
    RM=rm -rf
```

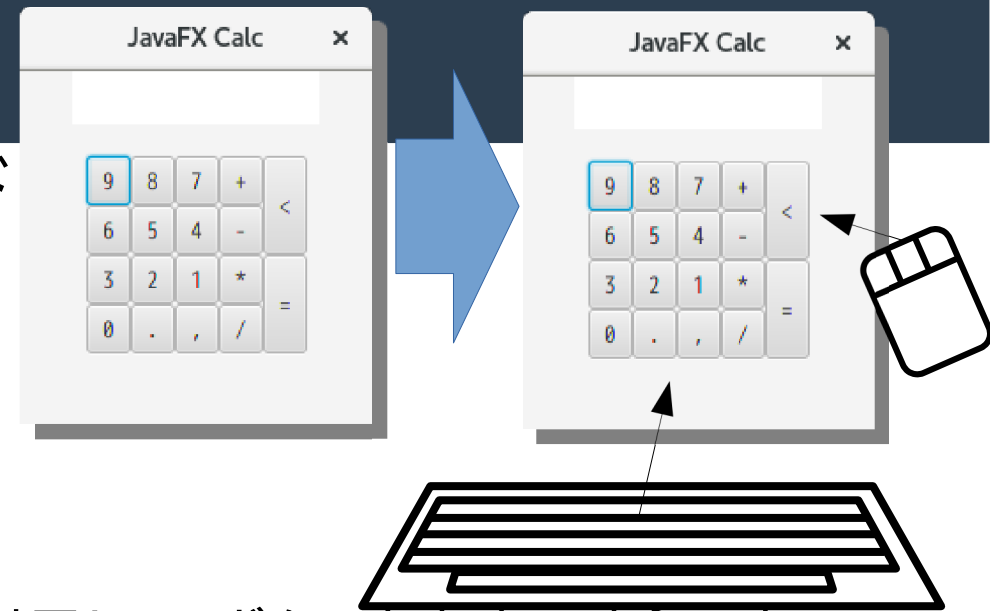
5行目
定数 JAVABINにjavacコマンドがある
ディレクトリを絶対パスで設定する必要
がある場合もある

課題 1

- `para.calc.Calculator` にキーボード入力機能を追加せよ

小数も入力可能

“<” のキー入力は文字列の末尾の1文字を削除する機能とする



今回は`javafx.scene.layout.GridPane`を利用して ボタンを右上のようにすでに配置済みです

1.1) ボタンのクリックを繰り返すことで、数値と四則演算記号を「, 」区切りの**逆ポーランド記法**で表記する文字列データを作成する機能を前回付け加えましたが、同様にキーボード入力でも文字列データを作成する機能も追加せよ

今回も一文字入力される毎に、更新された文字列が、Label inputに表示されるようにすること

hint 文字列データの保持には `java.lang.StringBuilder` を使うと便利

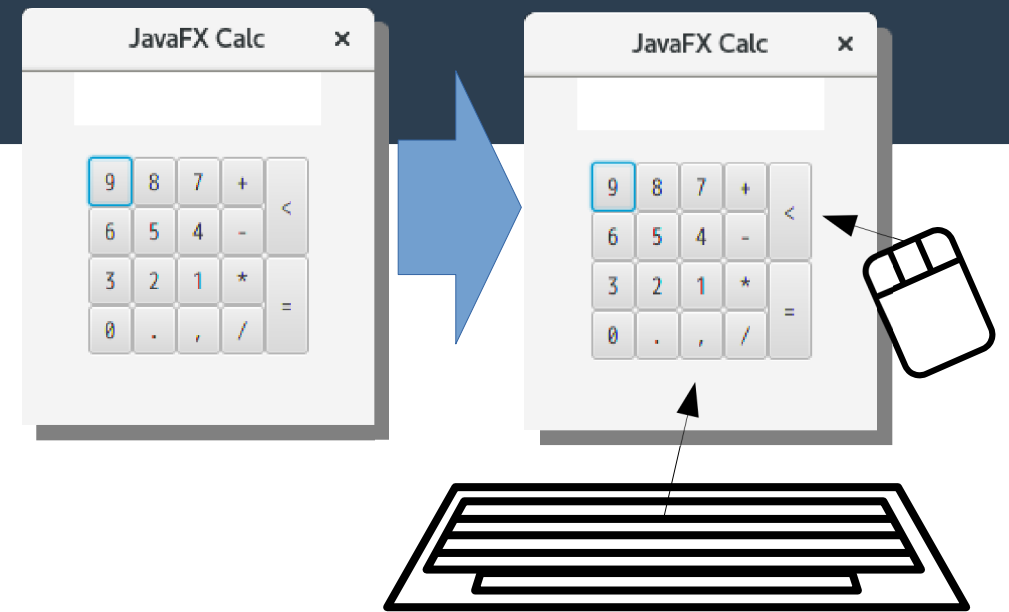
hint シーングラフのどのオブジェクトにイベントハンドラを登録すればユーザにとって便利かを考える

hint `Event` クラスが持っているキー入力情報を使う

またキーボードの“=”を打つことで、作成した文字列をStringクラスのインスタンスにして、`para.calc.Executor`の継承クラスの`operation()`に渡し、演算結果をLabelクラスの `output`を使って表示する一連の処理が行われるようにせよ

課題 1

- `para.calc.Calculator` にキーボード入力機能を追加せよ



1.2) シーングラフのどのオブジェクトにイベントハンドラを登録したかを述べて、その実装法がユーザの様々な操作において便利であることの理由を説明せよ

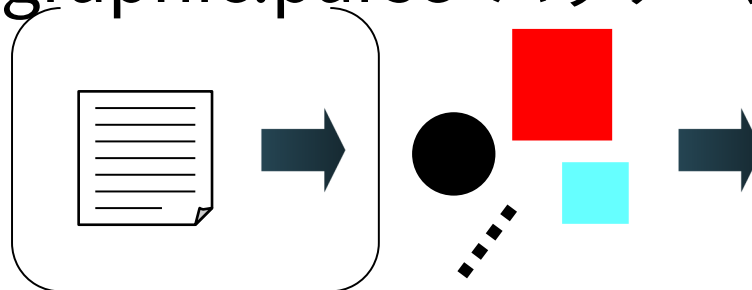
課題 2 以降のプログラムについて

- 今後の演習で利用するプログラムに慣れるため、オブジェクト指向のプログラミングの復習と、作られたプログラムの部品を使った短いプログラムの改造をします

課題 2 以降のプログラムの説明 (1) 概要

● プログラムは4つのパッケージから成る

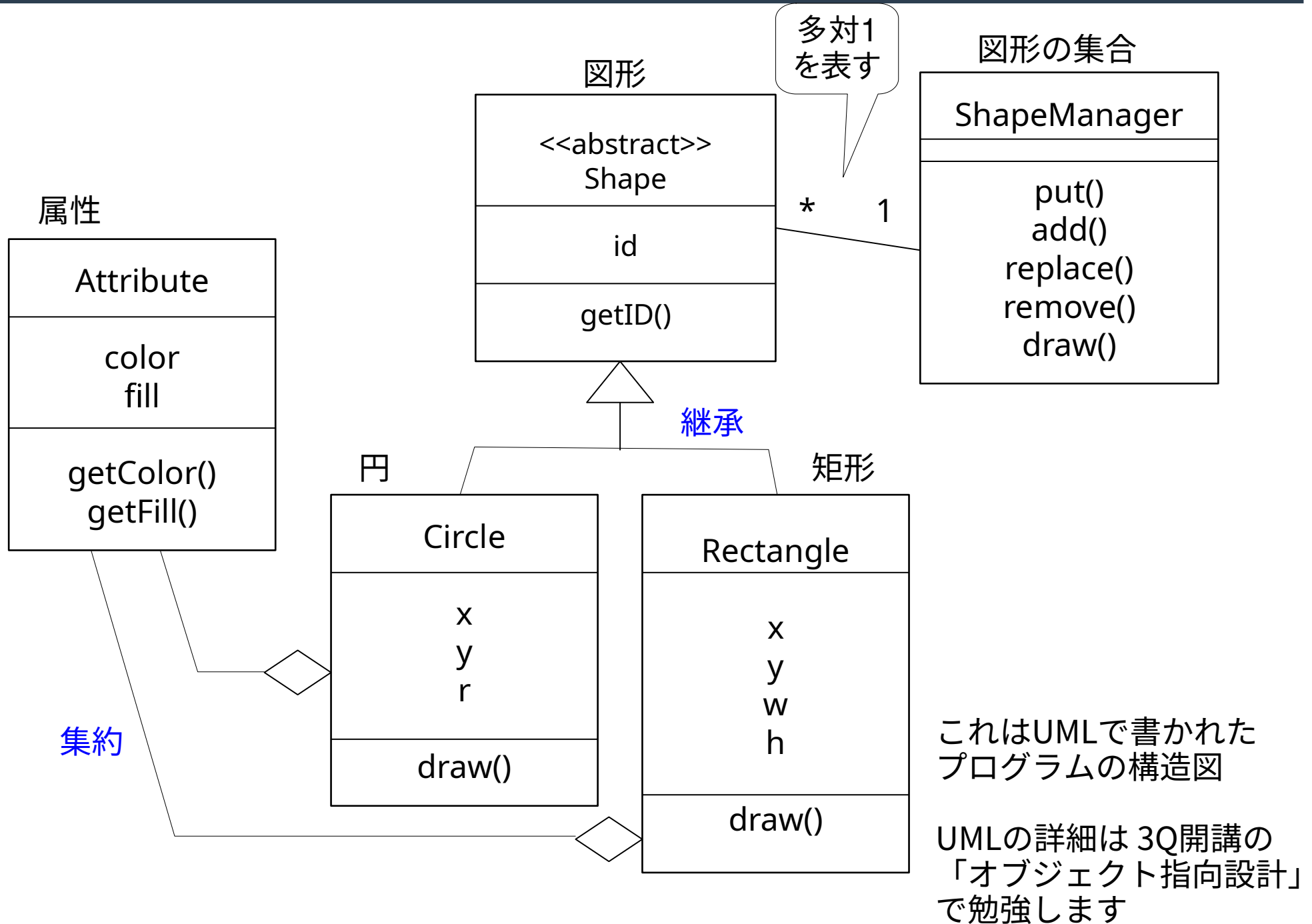
- メインプログラム (para パッケージ)
- 図形 (para.graphic.shape パッケージ)
- 出力装置 (para.graphic.target パッケージ)
- 構文解析器 (para.graphic.parse パッケージ)



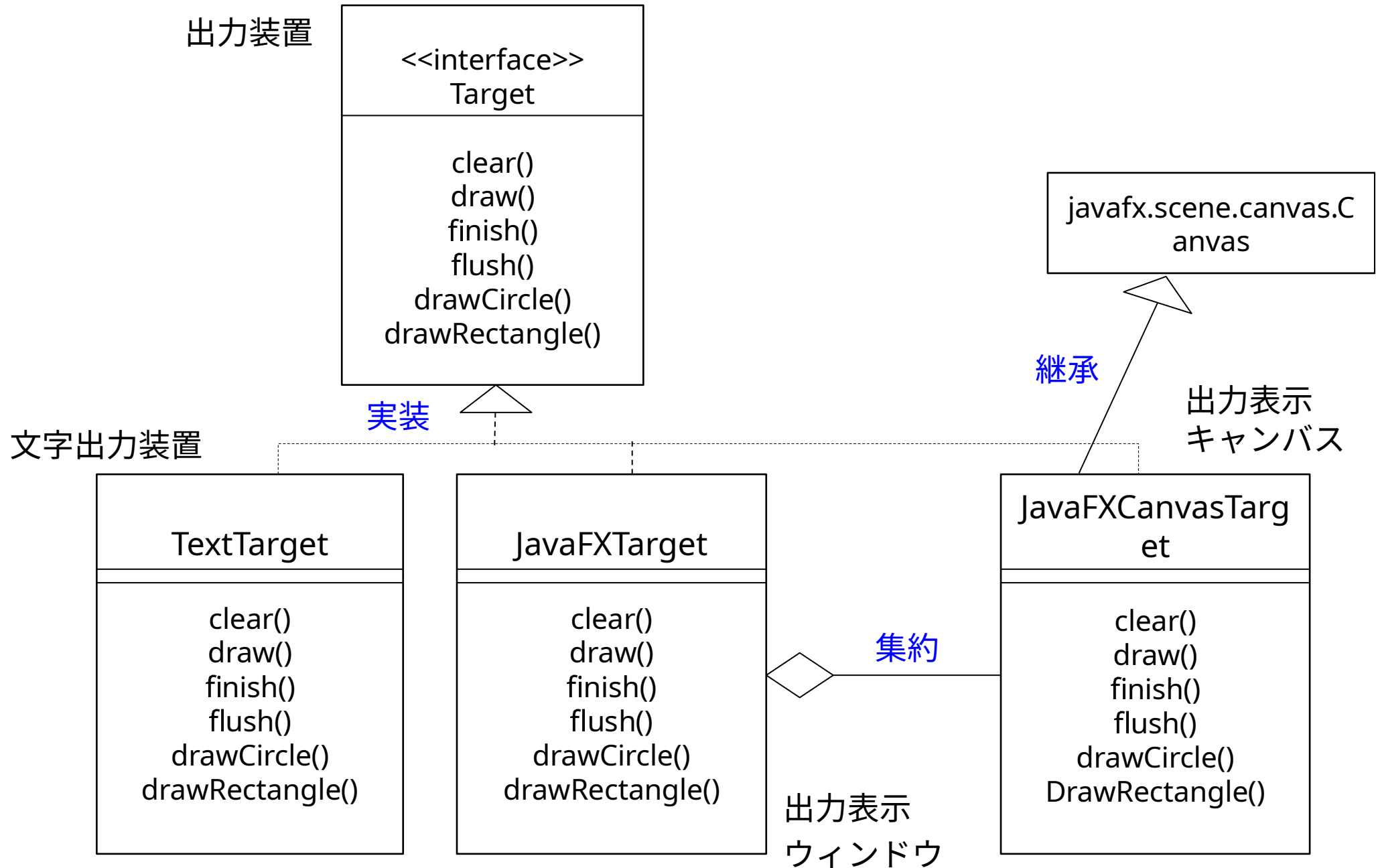
● 基本的流れ

- メインプログラム内で図形オブジェクトを生成し、出力装置に対して出力する
- 命令が書かれた文字列から構文解析器で図形オブジェクトを生成することもある

課題 2 以降のプログラムの説明 (2) 図形



課題 2 以降のプログラムの説明 (3) 出力装置

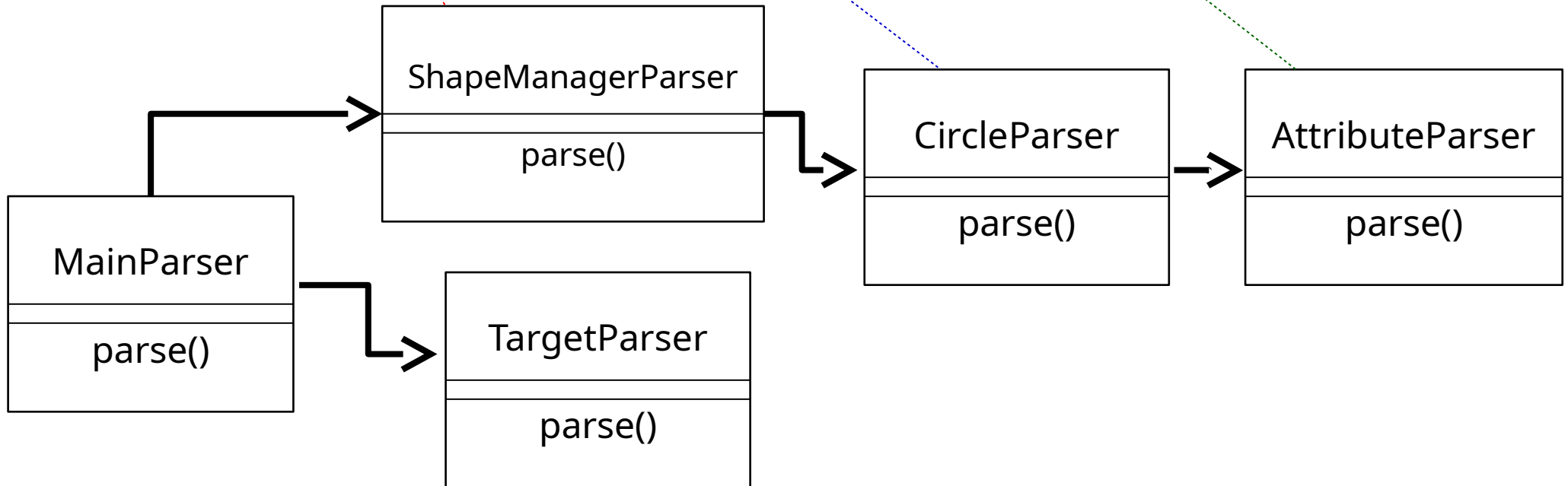


課題 2 以降のプログラムの説明 (4) 構文解析器

● 命令文を解析しながら対応するオブジェクトを生成

命令文：

shape 0 Circle 10 10 40 Attribute 100 40 60 true



課題 2

- **Main00, Main01, Main02, Main03 というデモプログラムをコンパイル・実行する**

Main0?.java の中で使われている Target 型変数の指すインスタンスを TextTarget 型と JavaFXTarget 型それぞれにして実行してみる

- **javadoc コマンドを実行して HTML 文書を生成し，ブラウザで閲覧する**

Mac OS X では `open HTML ファイル名` とすればブラウザが起動する
各クラスのパッケージ名などを確認する

ブラウザのエンコーディングの設定は UTF-8 にする

コンパイルエラーが起こる場合は展開直後にトップディレクトリで
一度 `make clean` とタイプする

課題 3

- 新たな図形として三角形を追加しなさい
- 具体的には：
 - Triangle クラスを追加する
 - Triangle クラスの中には javadoc 用のコメントも記入する
 - 三角形を指定するための命令文を導入し，その命令文を解析できるよう構文解析プログラムを適宜変更する
 - Main02.java で円の代わりに三角形を用いた表示が行われるように変更する

※ **Java Platform Standard Edition の標準 API, JavaFX API を適宜利用すること**

<https://docs.oracle.com/javase/jp/17/docs/api/overview-summary.html>

<https://openjfx.io/javadoc/18/index.html>

課題 4

- package parser の動作を次の文字列 data を解析する場合を例にとり説明しなさい
- 特にこの実装方法が if-else 文を並べた実装方法と比較して実装する上で優れていると気づいた点について述べなさい。ただし、オブジェクト指向言語が持つ大事な特徴に関係付けて回答すること

```
String data =
```

```
"shape 0 Circle 10 10 40 Attribute Color 100 40 60 Fill true\n"+
```

```
"target draw\n"+
```

```
"target flush\n";
```

```
MainParser mp = new MainParser(...);
```

```
mp.parse(data);
```

課題 5

- ダウンロード直後の状態では、`para.graphic.parser` パッケージにある構文解析用のクラスは全て `public` 修飾子がついている
すなわち、`para.graphic.parser` の外にあるクラス（以下、外部クラス）から全て参照可能である
- パッケージの外から見える必要のないクラスと見える必要のあるクラスを分け、それぞれに適したアクセス修飾を付けなさい
- なぜそのようにアクセス修飾を付けのか、その理由を説明しなさい

javaのアクセス修飾は次の4種類

`public`、`protected`、`private`、無修飾

hint スコープルールの質問です

課題 6

- 余力のある人は、
`para.graphic.parser.ImageParser` の仕組みを説明
しなさい

**hint: ImageParser 内で使用されているクラスの
Javadoc を読んで各メソッドが行う処理を調べましょう**

追加点が手に入ります

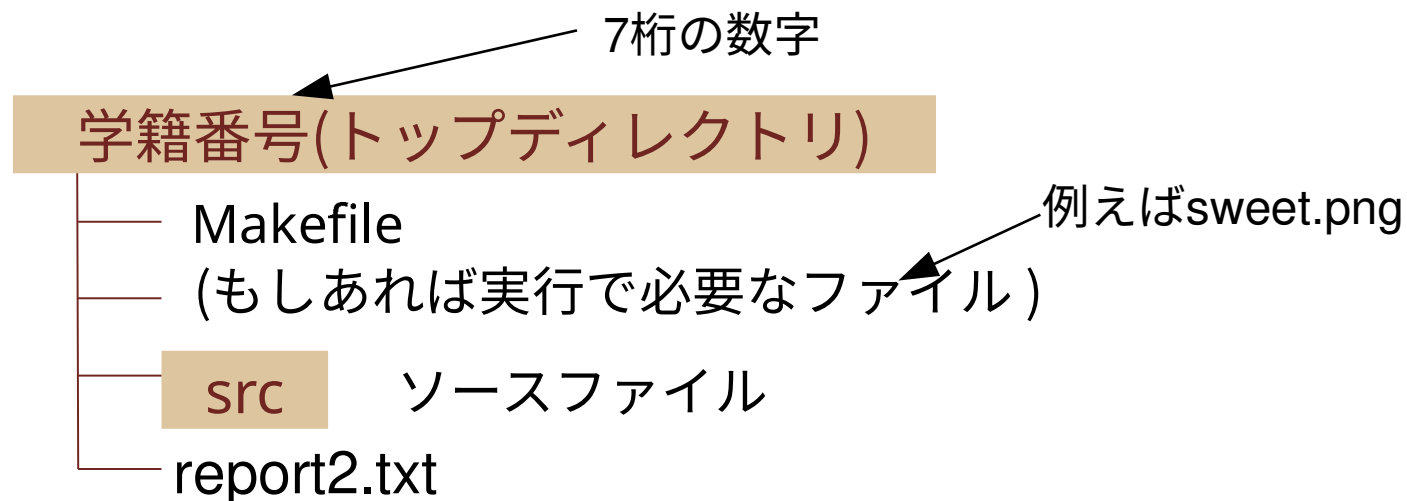
回答しましょう！！

提出方法 (1 of 3)

- para2.zip を展開したディレクトリ構造を保ったまま、課題 1,3,5 の変更作業を行う
- 課題 1,3,5 で変更した自分変更したファイルの先頭には自分の名前と学籍番号を書いておく

プログラムの場合はコメント内に書く

- 課題 1,4,5,6 の回答文、工夫点および感想を書いた report2.txt を用意する (雛形は課題のウェブページ)



回答プログラムとレポートの作成終了後、次のようにファイルを配置したディレクトリを作成

次ページに具体的な作業手順あり

提出方法 (2 of 3)

- 提出用ディレクトリを作成する

```
mkdir dir
```

学籍番号から7桁の数字にすること

- ソースファイルのディレクトリのコピーを作る

```
cp -R Para2/src dir
```

- *dir* に report2.txt もコピーする

```
cp report2.txt dir
```

- *dir* に課題2で必要なその他のデータがあればコピーする

```
cp Para2/otherfiles dir
```

例えばsweet.png

- 次のコマンドを実行する

```
zip ex2-2012345.zip -r dir
```

学籍番号に対応する7桁の数字にすること

- *dir* 以下の内容が圧縮され、**ex2-2012345.zip**が作られます

圧縮後に内容を“unzip ex2-2012345.zip”で確認すると提出ミスを防げて安全

提出方法 (3 of 3)

- 作成した zip ファイルを T2SCHOLA にアップロードする
- 締め切り

6 月 30 日 (木)

23:59 (JST)