

並列プログラミング Parallel Programming

2022 2Q

演習 第3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

- java の復習
- データの参照と実体の関係の理解
- Multi Thread Programming を経験

● 題材

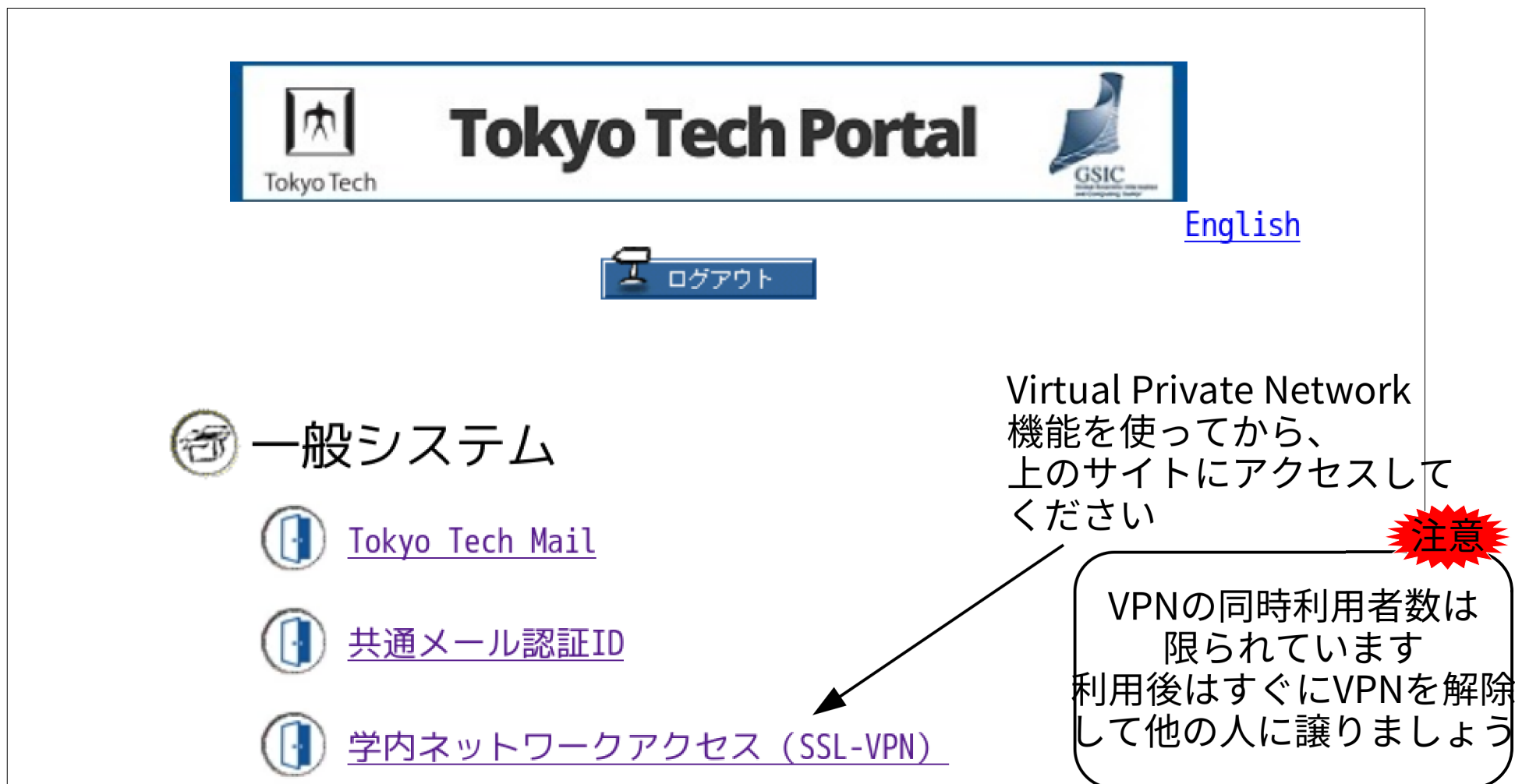
- 図形表示プログラム
- ゲームプログラム

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. 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. To the right of 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 (1)

- ダウンロードした **para3.zip** を展開する

```
unzip para3.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

Para2

Para3

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)

- Para3 以下の `._*` ファイルを再帰的に消去

```
cd Para3  
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

今回の演習ではトップディレクトリで

Main04.java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/Main04.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

今回の演習ではトップディレクトリにて

```
java --module-path ../javafx-sdk-18.0.1/lib --add-modules  
javafx.controls,javafx.swing -cp bin:lib/*: para.Main04
```

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.Main0? と
para.Game01 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

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、Para3 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 para.game 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)

- 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 Main04	Main04 をコンパイルして、実行
make Main05	Main05 をコンパイルして、実行
make Main06	Main06 をコンパイルして、実行
make Game01	Game01 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衣小される
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
```

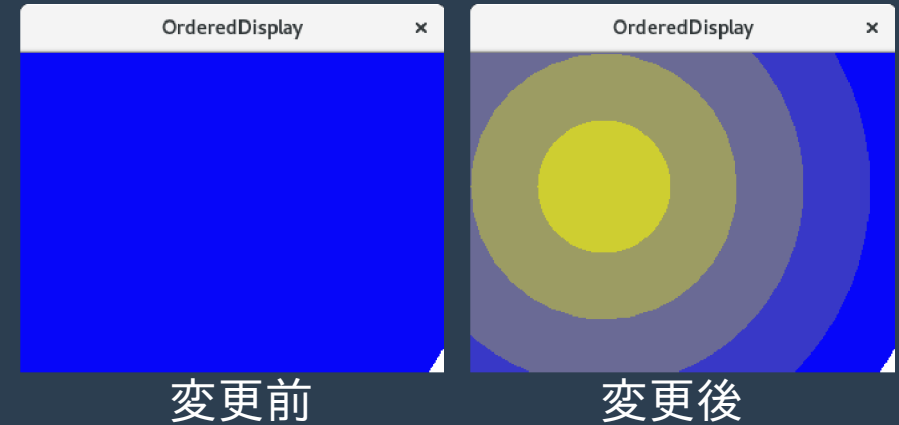
5行目
定数 JAVABINにjavacコマンドがある
ディレクトリを絶対パスで設定する必要
がある場合もある

課題 0

- **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 cleanall` とタイプする

課題 1

「Target.draw(ShapeManager)を呼び出した時 ShapeManagerへの登録順ではなく、Shapeのidの大きいものから先に図形を描画する」この仕様を満たすようShapeManagerを継承したOrderedShapeManagerを定義し para.Main04の中のShapeManagerのインスタンスをそれに置き換えよう



1.1) ShapeManagerを継承したOrderedShapeManagerをpara.graphic.shapeパッケージのクラスとして定義せよ。ただし、OrderedShapeManagerはShapeManagerを継承して定義すること。また、ShapeManagerクラスの定義は一切変えないこと。

hint ShapeManagerのメンバ変数dataはAbstractCollection<Shape>型である。その参照インスタンスは値をソートして格納するのではないArrayList<Shape>である。AbstractCollectionを継承したクラスにはソートして格納するものがないかを調べる

1.2) 以下の文の2つの空欄に当てはまる語をそれぞれ下の候補から選び答えよ

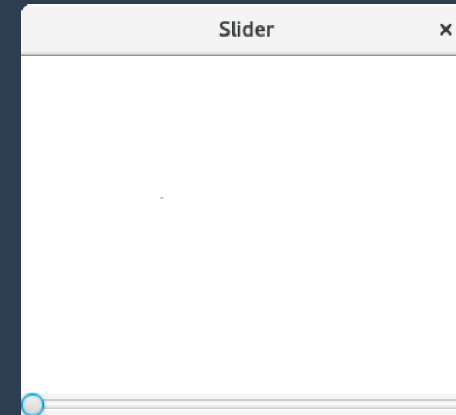
ShapeManagerのメンバ変数dataはクラスの外部から直接参照することができず、必ずメソッドを経由して値の読み書きを行うようになっている。このようにあるデータを直接参照できないように隠蔽することを「データを A する」と言う。

隠蔽されたデータはそれを包含するクラスのインスタンスを鍵としてその読み書きについて同期を取る仕組みを B と呼ぶ。java言語ではメソッドやブロックにsynchronizedを付けることでそれを実装できる。

1)抽象化 2)カプセル化 3)メンバ化 4)概念化 5)金庫 6)モニタ 7)指揮者 8)シングルブリッジ

課題 2

para.Main05はスライダを動かすと表示される図形が替わるプログラムである。ソースコードsrc/para/Main05.java中のvolatileの必要性を説明せよ

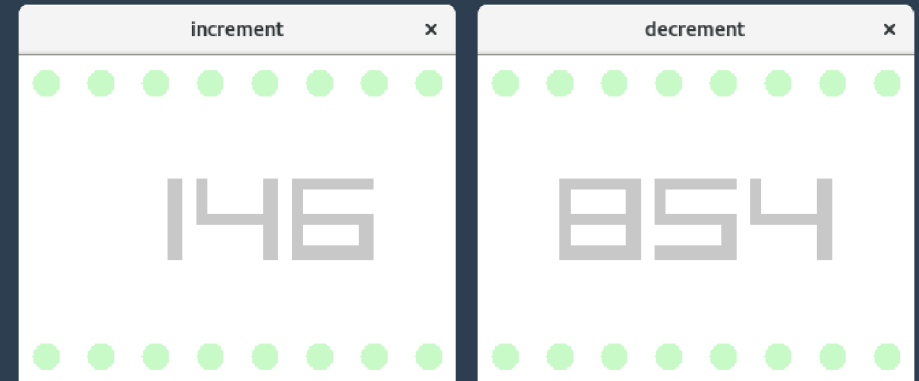


2.1) para.Main05の起動直後からユーザ操作中までにメンバ変数valueに直接関連した処理を行うスレッド名を列挙せよ。またそれぞれの役割について簡潔に述べよ。

2.2) volatile修飾を変数valueに付けない場合、プログラムの挙動に不具合が生じる可能性がある。その不具合が生じた際の、ユーザの視点からのプログラムの挙動を説明せよ。またその不具合の原因について講義で説明されたプログラムを処理する仕組みに基づき説明せよ。

課題 3

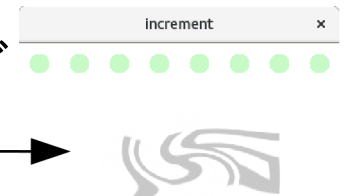
「起動すると開かれる2つの窓の中の数値が、2つのスレッドにより一方の窓内では000から999を循環するように1ずつ増加、もう一方の窓内では1ずつ減少する。」
この仕様のようpara.Main06を修正しよう



3.1) 配布されたpara.Main06の挙動は上記の仕様のようには動かない。どのような挙動であるかを説明し、その推測される原因も説明せよ。説明は読み手が問題点を明確に把握しやすく書かれていること。

hint 「数値の表示がグチャグチャに変化している」は曖昧度が高すぎ

上記の説明から解釈できてしまう挙動の一例



3.2) src/para/Main06.javaの23行目で変数sm2の参照先を

```
sm2 = sm1.duplicate();
```

のように変更する。このメソッドduplicateが行う処理はdeep copyであるかshallow copyであるかを答えよ。この変更によりプログラムはスレッドセーフとなったのであろうか。スレッドセーフであると考えれば、その理由をスレッドセーフを保証する3条件に基づき説明せよ。スレッドセーフでないと考えれば、必要な修正をソースコードに加えて、その修正理由を説明せよ。なお修正行は

src/para/Main06.java:?? 行目から??行目

のようにレポートに記載せよ

課題 4

para.Game01は「モグラ叩き」ゲームの雛形である。楽しめるようにゲーム性を高めよ。

モグラ叩きゲームを知らない人は

https://www.youtube.com/watch?v=6_qeXqvgFdg

<https://www.youtube.com/watch?v=uB4eRNWqa6A>

叩く操作はマウスクリック

4.1) スタートボタンが押された後の主要なスレッドを2つ挙げ、それぞれの役割について説明せよ。

4.2) para.game.GameFrame のメンバ変数xyはvolatileではなく、synchronizedメソッドにより同期が取られている。そのように同期を行わなかった場合、こういった不正確なデータの処理が起きるかを例を挙げて説明せよ。また、synchronizedメソッドによる同期の代わりにxyにvolatile修飾を付けることによるマルチスレッドへの対応では不十分であることの理由を説明せよ。

4.3) para.game.GameFrameのメンバ変数leftonやrightonは、volatile修飾によりマルチスレッドへの対応が十分であり、synchronizedメソッドやsynchronizedブロックによって同期を取る必要はない。なぜvolatile修飾を付けるだけでマルチスレッドへの対応が十分なのかを説明せよ。

課題 4

para.Game01は「モグラ叩き」ゲームの雛形である。楽しめるようにゲーム性を高めよ。

モグラ叩きゲームを知らない人は

https://www.youtube.com/watch?v=6_qeXqvgFdg

<https://www.youtube.com/watch?v=uB4eRNWqa6A>

叩く操作はマウスクリック

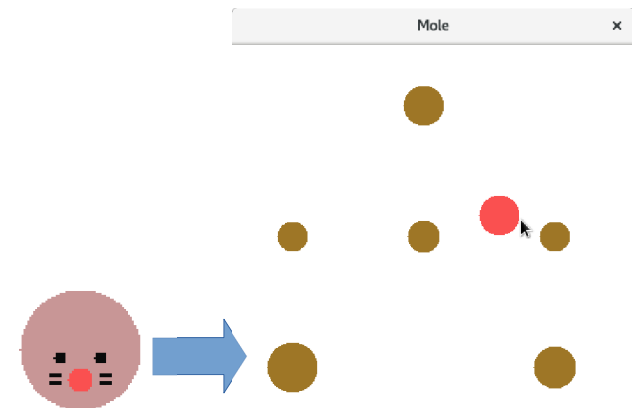
4.4)ゲーム性を高めるため以下を実装せよ

- 得点ルールを定義しその得点を刻々表示する
- 課題2で表示される図形を利用する
- ゲームの終了条件を定める

また実装部分が分かるように説明せよ

4.5)余力ある人は、更なるゲーム性の向上のための実装を追加し、その実装部分について説明せよ

昨年も多くの人が回答してくれています!!



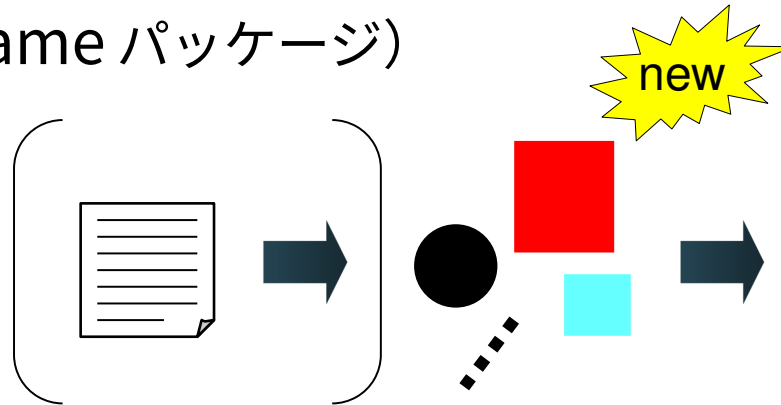
146 →



課題のプログラムの説明 (1) 概要

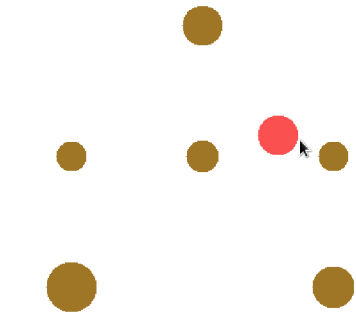
● プログラムは5つのパッケージから成る

- 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)
- ゲーム制作用 (para.game パッケージ)

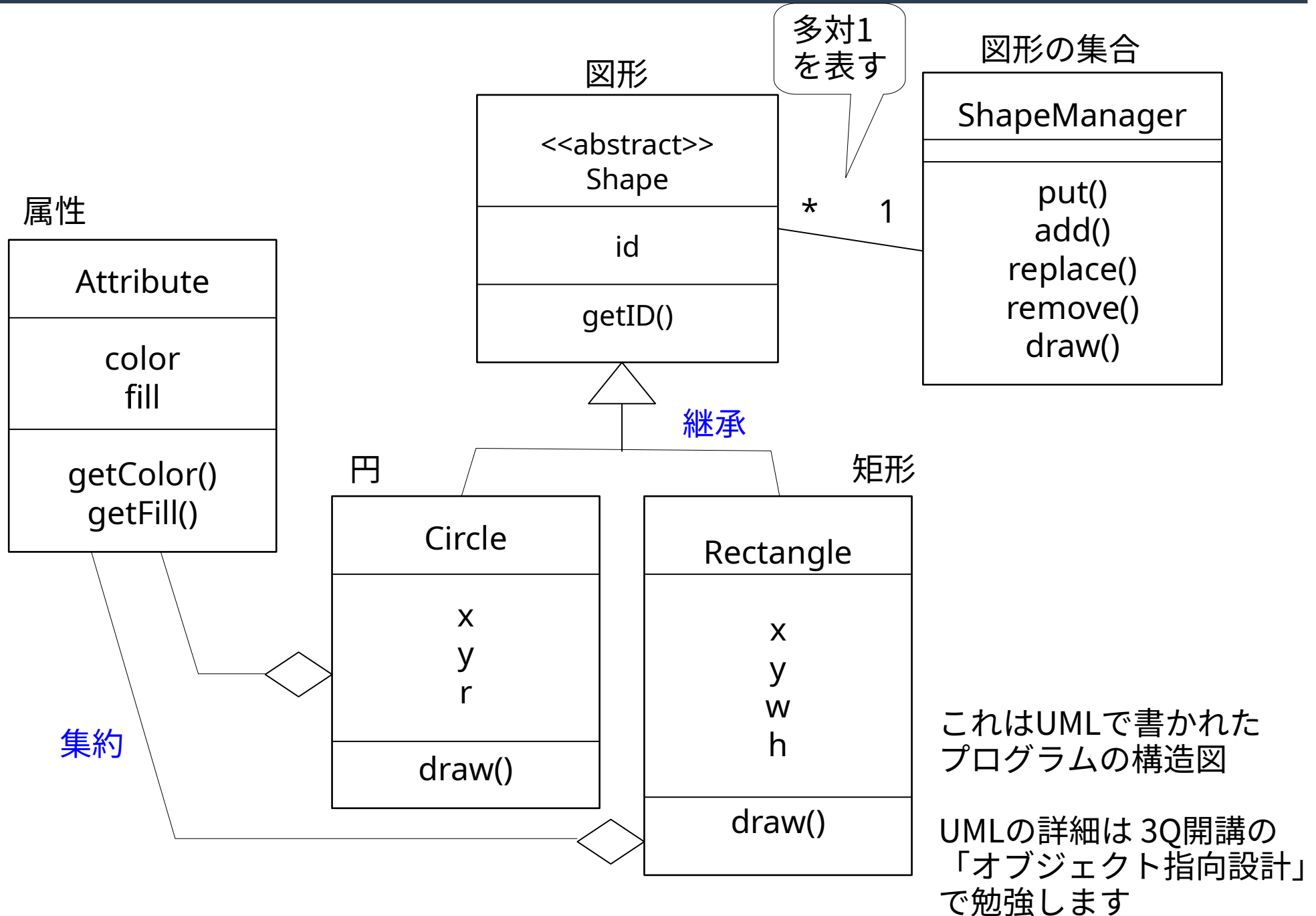


● 基本的流れ

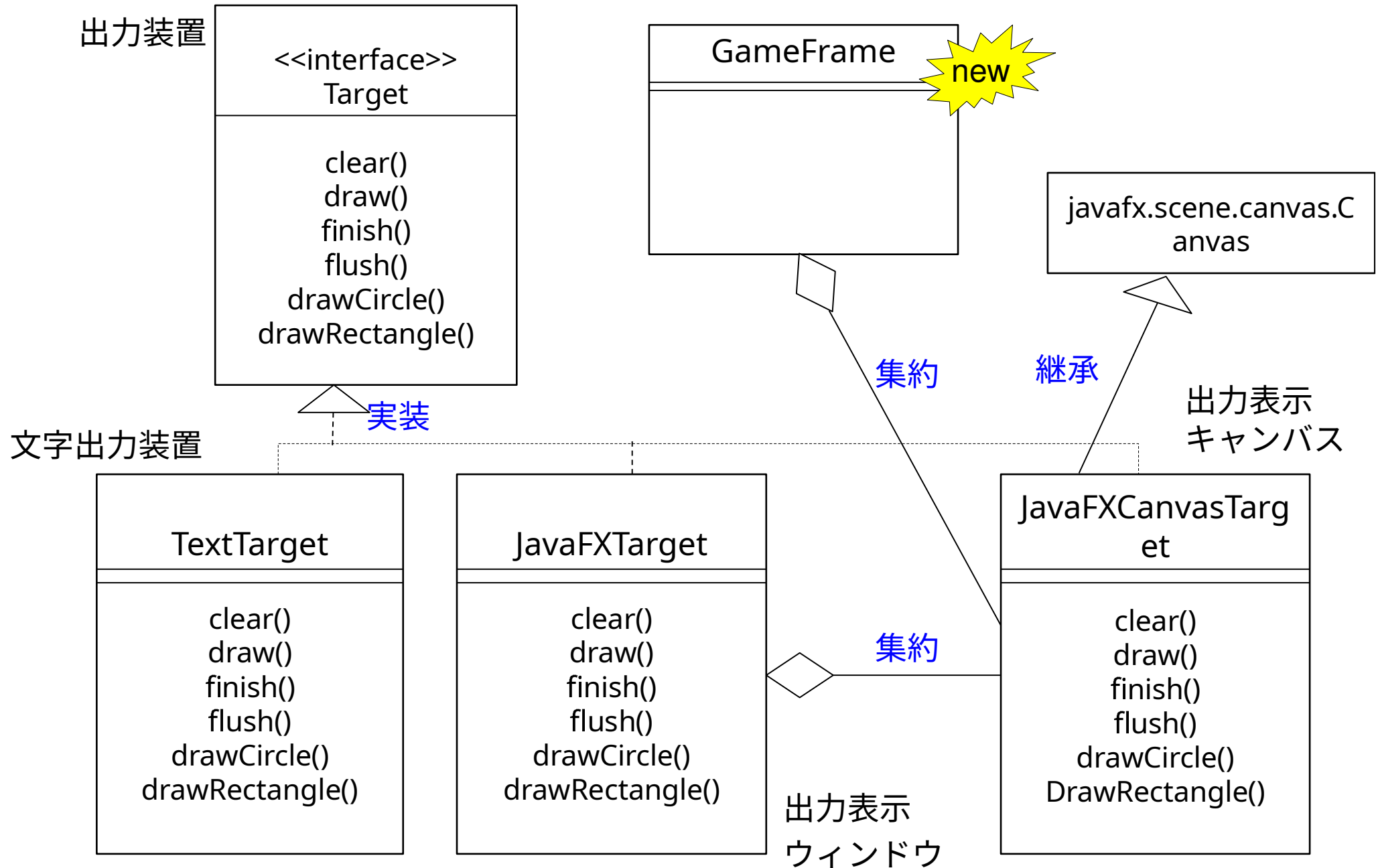
- メインプログラム内で図形オブジェクトを生成し、出力装置に対して出力する
- 命令が書かれた文字列から構文解析器で図形オブジェクトを生成することもある



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

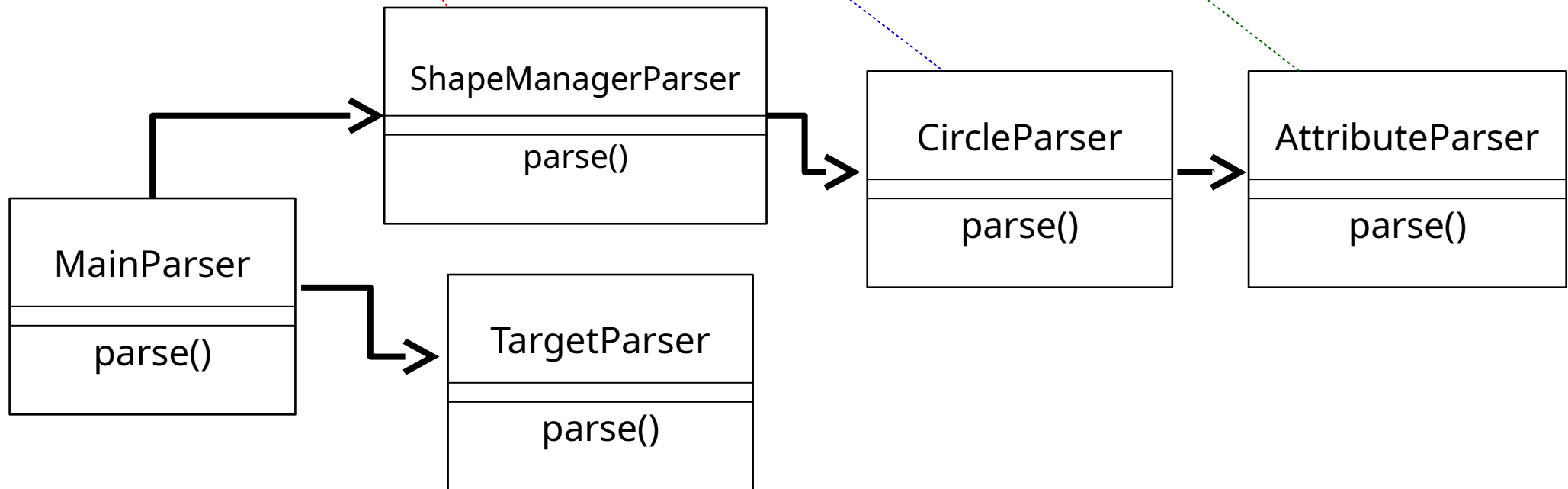


課題 2 以降のプログラムの説明 (4) 構文解析器

● 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



提出方法 (1 of 3)

- para3.zip を展開したディレクトリ構造を保ったまま、課題の変更作業を行う
- 各課題で自分が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 から 4 の回答文、工夫点および感想を書いた report3.txt を用意する (雛形は課題のウェブページ)

学籍番号(トップディレクトリ)

7桁の数字

Makefile

(もしあれば実行に必要なファイル)

src

ソースファイル

report3.txt

例えばsweet.png

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

学籍番号から7桁の数字にすること

```
mkdir dir
```

今回はPara3

● ソースファイルのディレクトリのコピーを作る

```
cp -R トップディレクトリ/src dir
```

● **dir** に Makefile report3.txt もコピーする

```
cp トップディレクトリ/Makefile トップディレクトリ/report3.txt dir
```

● **dir** に課題4に必要なその他のデータがあればコピーする

```
cp トップディレクトリ/otherfiles dir
```

例えばsweet.png

● 次のコマンドを実行する

学籍番号に対応する7桁の数字にすること

```
zip ex3-2012345.zip -r dir
```

● **dir** 以下の内容が圧縮され、**ex3-2012345.zip** が作られます

- 圧縮後に内容を“unzip ex3-2012345.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

7 月 7 日 (木) 23:59 (JST)