並列プログラミング
Parallel Programming

2024 2Q

演習　第5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

演習課題概要

- **目的**
  - **java Stream** の並列計算を体験
  - **java** の動態の観察 **visualvm**

- **題材**
  - 映像の減色処理を行うプログラム
  - 衝突判定を並列で行う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

次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/

準備（0）

- プログラムの処理を観察するためのソフトウェアのインストール
- visualvm のインストール
- VisualVM https://visualvm.github.io/download.html
- linux,windows では .zip をダウンロードしてインストール

  macOS では macOS Application Bundle .dmg をインストール

準備（0）

- **プログラムの処理を観察するためのソフトウェアのインストール**

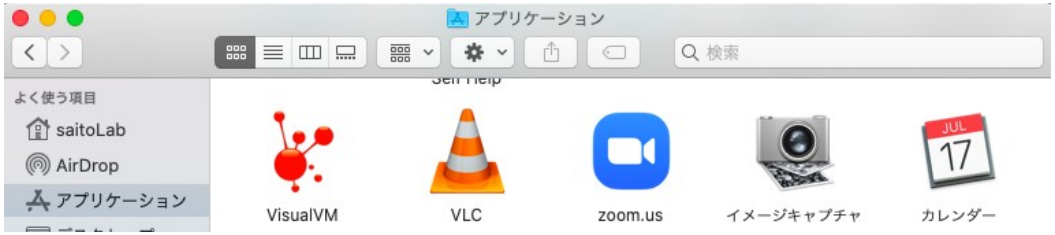
	インストール方法	起動方法
Linux	visualvm_218.zipを ホームディレクトリ/parawork に展開	**cd $HOME/parawork/visualvm_218/bin** **./visualvm --jdkhome JAVA_HOME**

JAVA_HOME は */usr*/lib/jvm/java-21-openjdk-amd64　などjavaのインストール場所に置換え

Windows	visualvm_218.zipを ホームディレクトリ/parawork に展開	**cd $HOME/parawork/visualvm_218/bin** **./visualvm.exe --jdkhome JAVA_HOME**

JAVA_HOME は C:\Program Files\Eclipse Adoptium\jdk-21.0.3.9-hotspot
などjavaのインストール場所に置換え

macOS	インストーラを起動して アプリケーションとして インストール



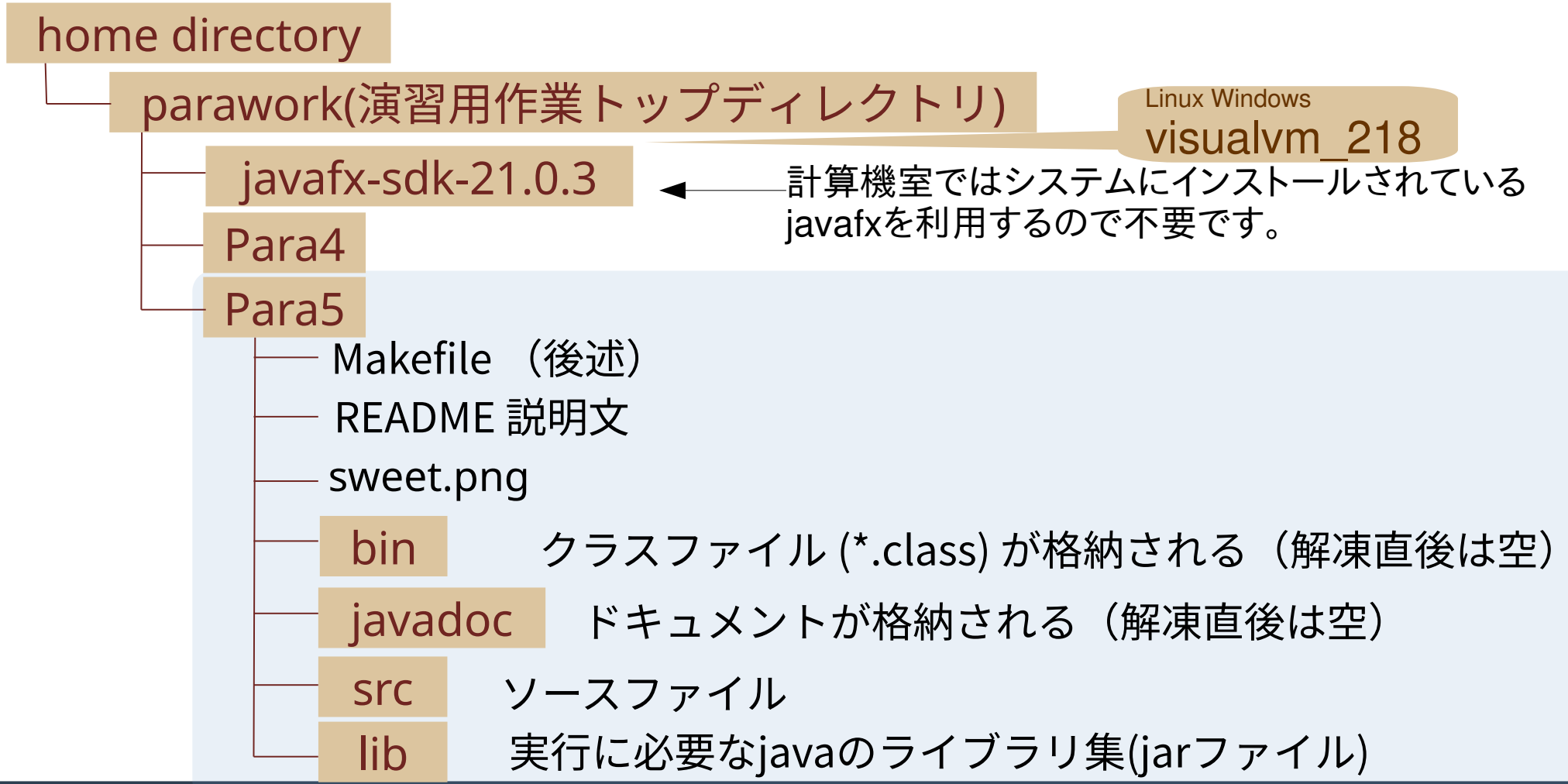
クリック

準備（1）

- ダウンロードした para5.zip を展開する

```
unzip para5.zip
```

- 解凍後のディレクトリ



準備 (2)

- **Para5 以下の ._* ファイルを再帰的に消去**

```
cd Para5
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) 計算機室の PC で作業する場合

- **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**

Main09.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

今回の演習ではトップディレクトリで

```
javac -d bin -encoding UTF-8 --module-path /Library/Java/JavaFX/javafx-sdk-21.0.2/lib --add-modules javafx.controls,javafx.swing -sourcepath src -classpath "lib/*:" src/para/Main09.java
```

実際は一行で書く

として下さい

`-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/21/docs/specs/man/javac.html でその他のオプションを確認すること

準備 (3) 自分の PC で作業をする場合

- **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

```
javac -d bin -encoding UTF-8 --module-path ../javafx-sdk-21.0.3/lib
--add-modules javafx.controls,javafx.swing -sourcepath src -classpath
"lib/*:" src/para/Main09.java
```

として下さい

Main09.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

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/21/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) 計算機室の PC で作業する場合

- **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

  実際は一行で書く

```
java --module-path /Library/Java/JavaFX/javafx-sdk-21.0.2/lib --add-modules javafx.controls,javafx.swing  -cp "bin:lib/*:" para.Main09
```

  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.Main??** と **para.Game03** 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/21/docs/specs/man/java.html
でその他のオプションを確認すること

準備 (4) 自分の PC で作業する場合

- **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-21.0.3/lib --add-modules
javafx.controls,javafx.swing  -cp "bin:lib/*:" para.Main09
```

実際は一行で書く

Windowsでは : を ; に置換

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.Main??** と **para.Game03** 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/21/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) 計算機室の PC で作業をする場合の設定

- 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、 Para5 直下

```
javadoc -html5 -charset utf-8 -encoding UTF-8 -d javadoc -sourcepath src
--module-path /Library/Java/JavaFX/javafx-sdk-21.0.2/lib --add-modules
javafx.controls,javafx.swing
-link https://docs.oracle.com/javase/jp/21/docs/api
-link https://openjfx.io/javadoc/21
-package para.game para para.graphic.shape para.graphic.target
para.graphic.parser para.graphic.camera
```

外部javadocのURIリンクを作成するには必要

パッケージ名

として下さい

※ https://docs.oracle.com/javase/jp/21/docs/specs/man/javadoc.html
でその他のオプションを確認すること

準備 (5) 自分の PC で作業をする場合の設定

- 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;
```

今回の演習では、 Para5 直下

```
javadoc -html5 -charset utf-8 -encoding UTF-8 -d javadoc -sourcepath src
--module-path ../javafx-sdk-21.0.3/lib --add-modules
javafx.controls,javafx.swing
-link https://docs.oracle.com/javase/jp/21/docs/api
-link https://openjfx.io/javadoc/21
-package para.game para para.graphic.shape para.graphic.target
para.graphic.parser para.graphic.camera
```

パッケージ名

として下さい

HTML文書の出力先ディレクトリ

実際は一行で書く

外部javadocのURIリンクを作成するには必要

※ https://docs.oracle.com/javase/jp/21/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 Main09`	Main09 をコンパイルして、実行
`make Main10`	Main10 をコンパイルして、実行
`make Main11 PARAMETER=SINGLE`	Main11 をコンパイルして、引数SINGLEを与えて実行
`make Main11 PARAMETER=PARALLEL`	Main11をコンパイルして、引数PARALLELを与えて実行
`make Game03`	Game03 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

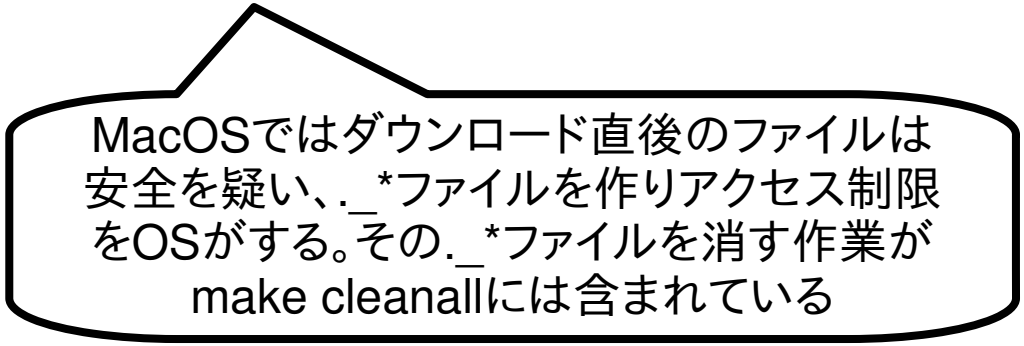
ソースコードを更新してもmakeが感知しない場合があるので、コードを書き換えても結果に変化がない場合は、一度 make clean して再コンパイルして下さい

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

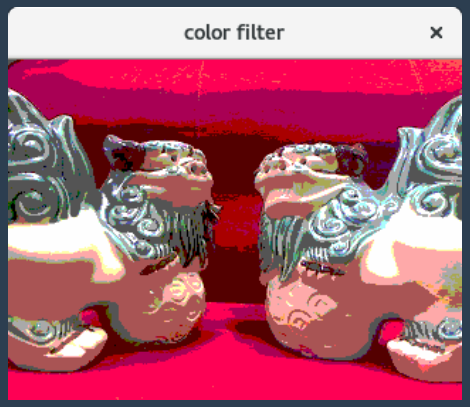
課題 0

- 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 とタイプすると解決する場合がある

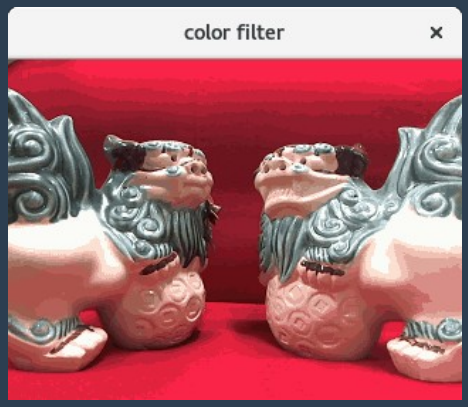


課題 1

para.Main09とpara.Main10はカメラ映像を減色して表示するプログラムである。Main09の方が処理が単純で高速であるが、Main10の方が減色しても元の色との違いが少なく画質は良い。Main10の減色処理を並列化し、処理速度を改善せよ



Main09



Main10

1.1)減色する前のカメラ画像では、個々の画素の色には表現可能な全 A 色中の1色が用いられている。Main09とMain10では各画素はそれぞれ B 色中と C 色中の1色が用いられる。空欄に入る数を答えよ。

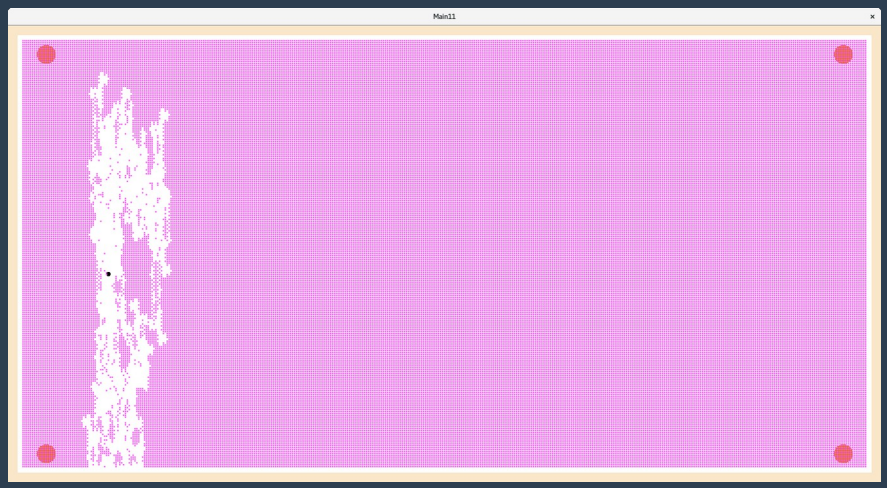
1.2) Main10が使用しているpara.TargetColorFilter2 のメソッドprocessの内部には、Streamを使った処理が3箇所ある。これらをParallelStreamに変換して並列化をすると並列化前とは異なった結果になってしまう。単純にParallelStreamに変換して問題となるStreamは何番目か。問題が発生するStreamが複数あると考える場合はそれら全てを挙げよ。

1.3) 1.2での単純な並列化では並列化前と異なる結果を生じてしまう原因を説明せよ。

1.4) 1.3の原因を取り除き、全てのStreamの処理を並列化対応させて、ParallelStreamに変換し、Main10の高速化を実現せよ。変更した点を説明せよ。

課題 2

玉とブロックが衝突するとブロックは消えて、玉は完全弾性で跳ね返る運動を続けるプログラムpara.Main11を実行して並列処理が計算時間を短縮するために有効であることを確認し、さらに計算時間を短縮するための改善を施せ



2.1) para.Main11は100単位時間分の玉の運動を計算すると終了する。起動時に引数として「SINGLE」を与えると並列処理なしで演算を行い、「PARALLEL」を与えると並列処理ありで演算が行われる。出力される時間は何を計測した時間であるかを説明し、どの程度高速化されたかを数値で答えよ。なお、計測するにあたり、計測条件や計測方法として注意した点も答えよ。

hint マルチタスクOSでは他の起動中のプロセスにより計測時間にばらつきが生じる。

2.2) プログラムの一部分だけの演算時間を調べるためのソフトウェアであるvisualvmを使って、より詳細に演算時間を調べよ。

visualvmの起動方法は、準備（0）のスライドを参照

（つづく）

課題 2

玉とブロックが衝突するとブロックは消えて、玉は完全弾性で跳ね返る運動を続けるプログラムpara.Main11を実行して並列処理が計算時間を短縮するために有効であることを確認し、さらに計算時間を短縮するための改善を施せ

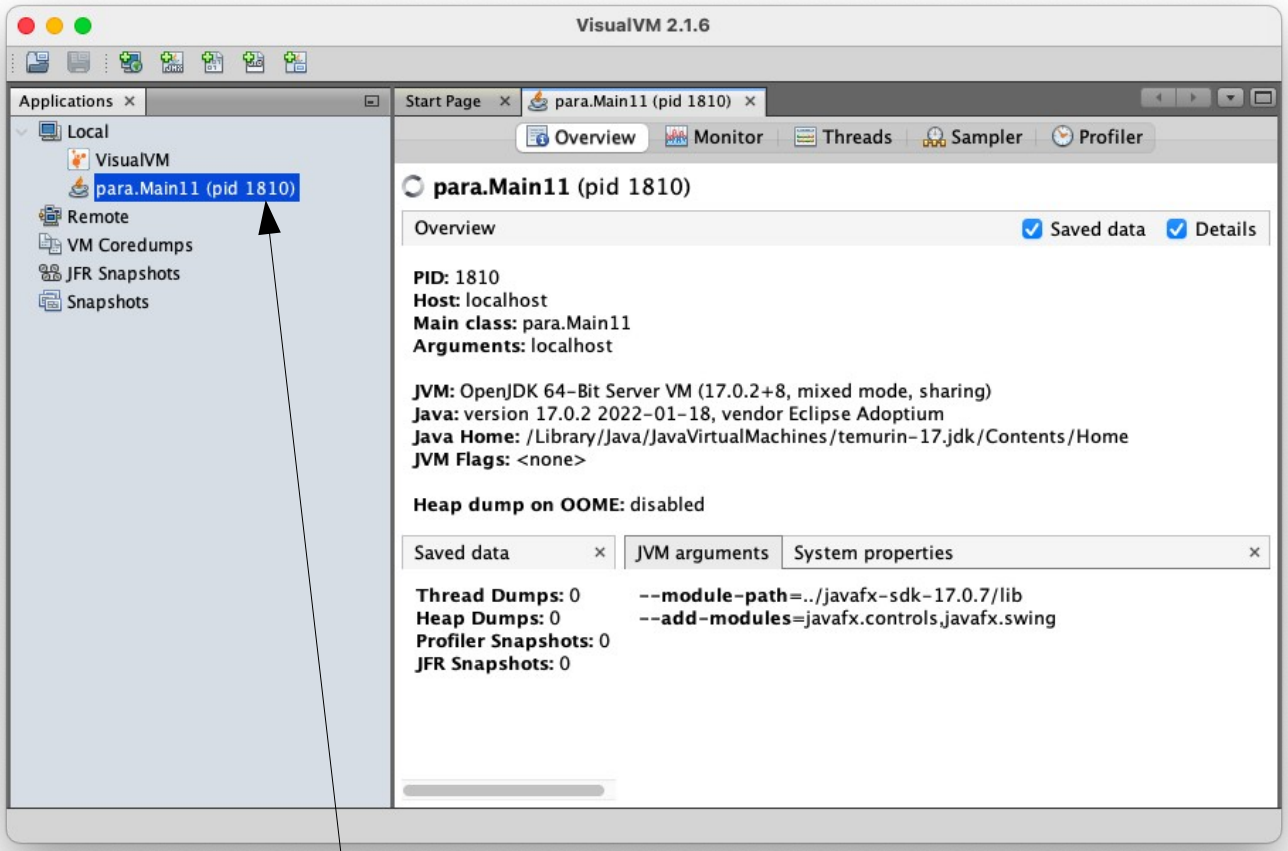
衝突判定部分は並列化により何倍速くなったかをvisualvmを用いて調査せよ。visualvmをどのように利用したのかを説明し、数値を交えて高速化の調査結果を答えよ。
hint visualvmで得られる数値の意味を調べてから回答すること

2.3) 衝突判定部分は並列化による改善法が実装されているが、それ以外の部分でプログラム全体を高速化するにあたり、効果的な改善対象はどこであろうか。visualvmを用いて、改善対象を選び、その部分を高速にするようプログラムを改良せよ。改良部分のコードがどこであるかと、どのような方針で改良したかを説明せよ。

2.4) 2.3で改良した部分のみに関して、改良前と比べ、処理時間は何％になったかをvisualvm を用いて調べて回答せよ。ただし処理時間の定義にはこの場合にもっとも適すると考えるものを選び、その定義も説明せよ。

課題 2　補足説明 visualvm

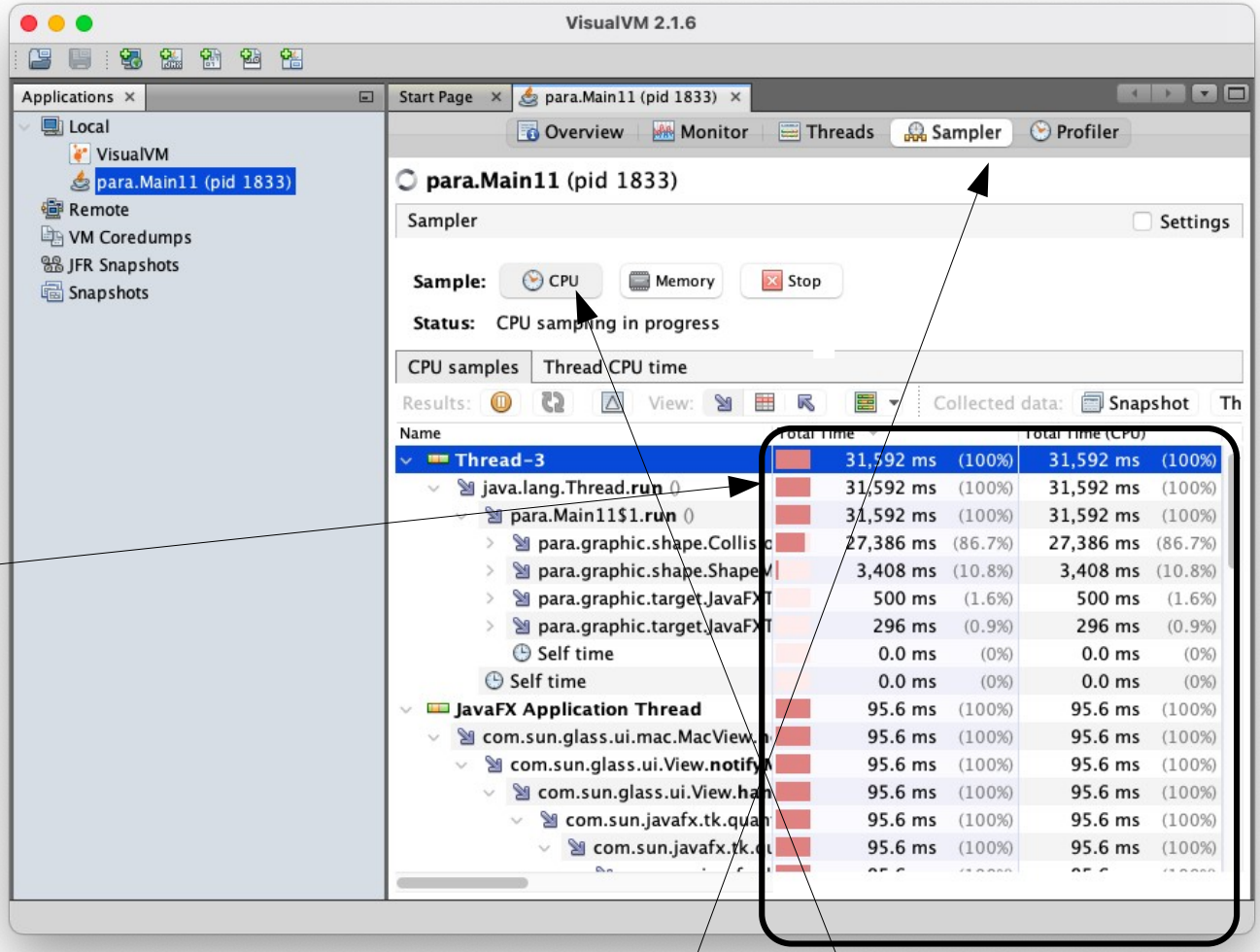
- Step1
  visualvmを起動
- Step2
  Main11を起動



Main11 を走らせると、それを動かすjava VMが現れる
ダブルクリックでvisualvmとそれとの接続を開始する

課題 2　補足説明 visualvm

- Step1
  visualvmを起動
- Step2
  Main11を起動
- Step3
  Main11の観察開始



色々な測定値が表示される

サンプラをClick してから、CPUをクリックして
CPUサンプルを開始

課題 3

para.Game03は「ブロック崩し」ゲームの雛形である。楽しめるようにゲーム性を高めよ。

ユーザの操作はキーボードのFキー（左）とJキー（右）

3.1)ゲーム性を高めるため以下を実装せよ

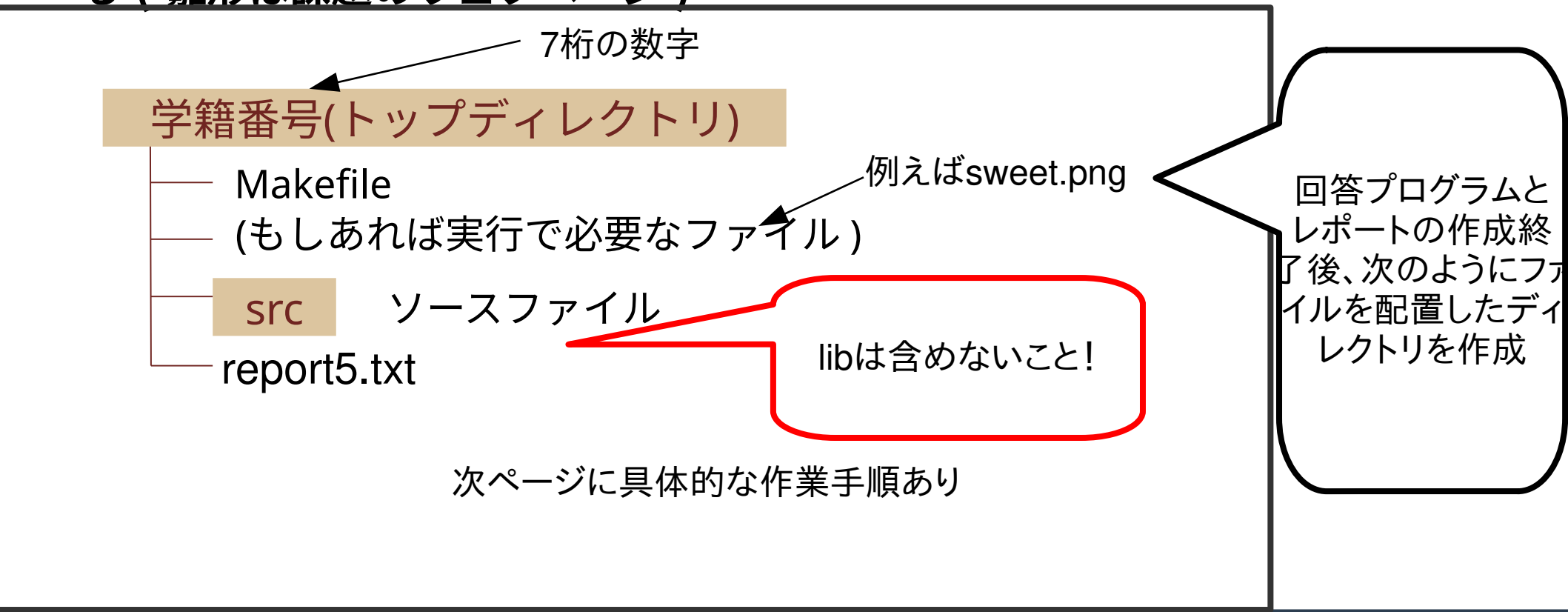
- 得点ルールを定義しその得点を刻々表示する
- ゲームの終了条件を定める
- ゲームオーバーの後、スタートボタンを押すと新しくゲームを開始できる
- スタートボタンの左にある数値選択機能を利用して、ゲーム開始時に難易度を設定できるようにする

上記の実装部分がコード中のどこか分かるように説明せよ

3.2)さらにゲーム性を高める方法を1つ以上考え、それを実装せよ
どのような方法であるかを説明し、その実装部分がコード中のどこか分かるように説明せよ

提出方法 (1 of 3)

- **para5.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 3 の回答文、工夫点および感想を書いた report5.txt を用意する (雛形は課題のウェブページ)**



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
```

- **ソースファイルのディレクトリのコピーを作る**

今回はPara5

```
cp -R トップディレクトリ/src   dir
```

- **dir に Makefile report5.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/report5.txt dir
```

- **dir に課題 3 で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 ex5-2212345.zip -r dir
```

lib ディレクトリは絶対含めない!!

- **dir 以下の内容が圧縮され、 ex5-2212345.zip が作られます**

圧縮後に内容を“ **unzip ex5-2212345.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 18 日（木）　23:59 (JST)