エルゴノミクスコンピューティング実習

01 Processingの概要

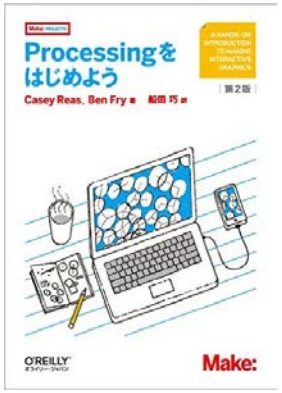
人間システム工学科 井村 誠孝
m.imura@kwansei.ac.jp

Processingの起動

- スタートボタン▷3. マルチメディア▷processing により行う．統合開発環境のPDEが起動する．
- アイコン一番左端のボタン(右向き矢印) でプログラムが実行される．
- Processing に関連する情報は，本家のウェブサイトが詳しいが，英語である．ただそれほど難しくはないしサンプルを見れば意味がわかるので，英語を参照する練習としてもよい題材であろう．
  - リファレンス: https://processing.org/reference/
  - ライブラリ: https://processing.org/reference/libraries/
  - サンプル: https://processing.org/examples/
- PDEのメニューFile ▷Examples... を開くと，大量のサンプルプログラムがすぐ実行できるようになっている．適宜実行してみると楽しい．

実習中，referenceは適宜参照する

- Processingに関する全てを逐一解説することはしません.
- https://processing.org/reference/ をブックマークしておき，適宜参照してください.
  - ウェブの日本語情報を検索するよりも，ここから適切な命令を探す方が，情報が正確で速い.
- 日本語の情報が必要な人は，書籍「Processingをはじめよう 第2版」がよいでしょう.



Processingの出自

- Processingのベース(コア部分およびライブラリ)はJava.
- JavaはCおよびC++から影響を受けている.
- C(プログラミング実習I), Java(プログラミング実習II)を習得していれば, ソースコードの記述は難しくない.

プログラミング言語としてのProcessing

- 変数と演算
- 制御構文
- 配列変数
- 文字列
- ファイル入出力

> リファレンスを参照しながら確認します

変数の型: Java由来

名前	種類	サイズ	値の範囲
boolean	論理型	1ビット	true / false
byte	整数型	8ビット	-128 から 127
char	整数型	16ビット	0 から 65535
int	整数型	32ビット	-2147483648 から 2147483647
float	浮動小数点型	32ビット	-3.40282347E+38 から 3.40282347E+38
color	整数型	32ビット	16777216色+透明度

Processingのライブラリは基本的に浮動小数点型としてfloatを使う.
color型は色の情報を保持するProcessing特有の型
C言語で育った人にとっては，byteとcharの違いに注意.

型の変換

- 精度が落ちる代入を行う場合，C言語のような暗黙の型変換(例: float型をint型に代入するときに，明示しなくても整数に丸め込まれる)は行われない.
  - 精度が上がる場合(例: int型をfloat型に代入)は自動的に変換される.
- 明示的型変換関数の使用を推奨.
  - `boolean()`, `byte()`, `char()`, `float()`, `int()`, `str()`
  - キャストも使用できる.

1 / 3 はProcessingでも 0

- 整数型と整数型の演算結果は，整数型になる．
- 適切に型変換することが必要．

×NG

```
int a, b;
float c;
a = 1;
b = 3;
c = a / b;
println(c);
```

○OK

```
int a, b;
float c;
a = 1;
b = 3;
c = a / float(b);
println(c);
```

制御構文

- CやJavaと同じなので省略
- 条件分岐: if文, switch文
- 繰り返し: for文, while文

配列変数

- 宣言は 型名[] で行う.
- 宣言だけではメモリが確保されない. 作成を行う.
- 配列のデータを管理するために役立つ関数については, リファレンスを参照すること.

配列の使い方を示すだけの, 全く面白くない例:
10要素の配列を宣言し, 要素番号の二乗を計算して格納

```
int[] x;
x = new int[10];
for (int i = 0; i < x.length; i++) {
  x[i] = i * i;
}
println(x);
```

配列変数の宣言と初期化いろいろ

- 宣言と作成と値の割り当てを別個に行う

```
int[] x;
x = new int[10];
x[0] = 0;
x[1] = 1;
:
```

- 宣言と作成を同時に行う

```
int[] x = new int[10];
x[0] = 0;
x[1] = 1;
:
```

- 宣言と作成と割り当てを同時に行う

```
int[] x = {0, 1, 4, ..., 81};
```

文字列

- 文字列はStringクラスを用いる.
  - Stringクラスのメソッドやコンストラクタについては,リファレンスを参照すること.
  - 文字列は書き換え不可．Stringの一部を書き換えることはできない.
  - 数値から文字列への変換は，str()関数か，Stringクラスのメソッド String.valueOf()を用いる.
    - フォーマット付きの変換がしたければString.format()を用いる.

文字列の結合などの演算も可能

```
String s1 = "Alan";
String s2 = "Turing";
String s3 = s1 + " " + s2;
println(s3);
```

ファイル入出力

- テキストファイルを行ごとにStringクラスの配列に読み込む方法

```
String[] lines = loadStrings("data.txt");
```

- Stringクラスの配列をテキストファイルに書き出す方法

```
String[] lines = {"sample", "text"};
saveStrings("data.txt", lines);
```

- プログラム実行中に少しずつ書き出したい場合は，PrintWriterクラスを用いる．
  - 詳細はリファレンス参照