

プロジェクト

パッケージ

Javaファイル

クラス

```
メソッド () {}  
ステートメント ;  
// コメント
```

Javaファイル

クラス

```
メソッド () {}  
ステートメント ;  
// コメント
```

クラス

モノ（インスタンス）を作るための
設計図。

インスタンス変数

インスタンスの状態や情報を表す。

インスタンスメソッド

インスタンスの行動を表す。

コンストラクタ

インスタンスを作るときに呼び出される
メソッド。クラスと同じ名前。

変数

データの入れ物。

代入演算子

=のこと。
変数にデータを入れるときに使う。

型

変数の中に入っているデータの種類を示すもの。

基本型

基本的なデータの型。全部で8つある。
int、boolean、floatなど

参照型

インスタンスのデータ型。
参照型の型は全てクラス。

byte 整数 (-128~127)

boolean true / false

short 整数 (-32768~32767)

char 一文字のUnicode文字

int 整数 (-2147483648~
2147483647)

float 32ビット単精度浮動小数点数
(要するに小数)

long 整数
(9,223,372,036,854,775,808~
9,223,372,036,854,775,807)

double 64ビット倍精度浮動小数点数
(要するに桁が大きい小数)

メソッド

コードをタスクごとにまとめたもの。

引数

メソッド内の処理に必要なデータ。

戻り値

メソッドが処理した結果のデータ。

mainメソッド

Javaプログラムの起動をするメソッド。

【想定通りに動かない時は……】

①実行時にエラーが起きる場合

エラー内容を確認してみよう！

②Javaの文法でエラーが起きている場合

コードをパーツごとにコメントアウトしてみよう！

③エラーは出ないけど結果が想定と違っている場合

デバッグ機能を使って、動きを細かく確認しよう！

if文

条件によって処理を分岐させる。elseやelse ifによって細かく条件を分けることができる。

switch式

変数の値によって処理を分岐させる。enumとの相性がいい。

比較演算子

値の比較に使う記号。==、!=、>など。

論理演算子

複数の条件を組み合わせるための記号。&&（かつ）、||（または）の二種類。

論理否定演算子

条件を反転させる記号。「!」のこと。

配列

複数のデータをまとめて管理する変数。
後から長さを変えることができない。

インデックス

配列の部屋番号。0から始まる。

ArrayList

便利な機能がたくさんついた配列。
参照型しか格納できない。

List.of()

ArrayListを作るためのメソッド。
List.of()を使って作られたArrayListは、
変更ができない。

【算術演算子】

+ : 足す **$10 + 5 \rightarrow 15$**

- : 引く **$10 - 5 \rightarrow 5$**

*** : 掛ける** **$10 * 5 \rightarrow 50$**

/ : 割る **$10 / 5 \rightarrow 2$**

% : 余り **$10 \% 5 \rightarrow 0$**

++ : インクリメント (1 増やす)

num++; // numに1を足す

-- : デクリメント (1 減らす)

num--; // numから1を引く

【代入演算子】

`+=` : 左辺に右辺の値を足す

`-=` : 左辺から右辺の値を引く

`*=` : 左辺に右辺の値を掛ける

`/=` : 左辺を右辺の値で割る

for文

インデックスを使って繰り返し処理を行う。
初期化式、条件式、変化式の三つを使って、
ループをコントロールする。

拡張for文

配列の扱いに特化したfor文。
配列の要素を一つずつ取り出す。

while文

条件を使ってループをコントロールする。
条件がtrueの間、繰り返し処理を行う。

continue;

次のループに進む。

break;

ループをやめる。

static

インスタンスを作らずに変数やメソッドを使えるようにする。

static変数

全てのインスタンスで同じ値を共有したい場合に使う。

staticメソッド

いつ誰がやっても同じ処理になるような汎用的なメソッドはstaticにする。

定数

変更できないデータ。
static finalが付く。名前は大文字のみ。

継承

あるクラスを拡張して、似たような別クラスを作ること。書き方は「extends 親クラス」

オーバーライド

親クラスのメソッドを上書きすること。

IS-Aテスト

クラスAがクラスBを継承できるかをチェックするテスト。「AはBです」が成立すれば継承できる。

Objectクラス

全てのクラスの大元のクラス。
全てのクラスはObjectクラスを継承している。

多態性 (ポリモーフィズム)

- ・ 子クラスは、親クラス型として扱うことができるという性質。
- ・ 同じメソッドの処理を、子クラスごとに変えることができる性質。

ジェネリクス

どんなデータでも汎用的に使える機能。
宣言時に<>を使ってデータを指定する。

抽象クラス

継承される前提のクラス。
子クラスを管理したり、コードに統一性を
持たせることができる。

抽象メソッド

空のメソッド。オーバーライドされる前提の
メソッド。子クラスは
オーバーライドしないとエラーになる。

ラムダ式

無名のメソッドを簡単につくることができる機能。

関数型 インターフェース

ラムダを定義するための型のようなもの。

メソッド参照

ラムダをさらに短くした書き方。
「クラス名::メソッド名」と書く。

関数型 インターフェース	引数	戻り値	実行するメソッド
Supplier	なし	あり	get()
Consumer	一つ	なし	accept()
Function	一つ	あり	apply()
Predicate	一つ	あり (boolean型)	test()

Enum (列挙型)

定数をグループ化する機能を持つクラス。
選択肢が限られている定数を作るときに
使われる。