

【Java】入門

環境編

■環境構築（Linux）

- ▶ ☆ インストール (on Debian)

基礎文法編

■初歩的注意

- ▶ ※ 大文字と小文字を区別する言語である。
- ▶ ※ 変数に値が未設定の状態で変数の値を取得しようとするとエラーになる。ただし、要素数を決めた配列変数については、はじめから各要素に何らかの値が設定されている。
- ▶ ※ ガベージコレクション (GC) が常に働いている。

■データ型の種類

- 整数 byte short int long 300000L -2000001 (longの例)
- 小数 float double 30.5F -20.5f (floatの例)
- 真偽値 boolean true false
- 文字 char 'a' '亜'
- 文字列 String "Hello" "やあ"

■基礎

- ▶ ☆ 定型文
- ▶ コメントのしかた
- ▶ 変数を定義
- ▶ 定数を定義
- ▶ 変数の参照を切る

■標準入出力

- ▶ 1つの整数の入力受付
- ▶ 1行Stringの入力受付
- ▶ 出力

【Java】入門

環境編

■環境構築（Linux）

- ▶ ☆ インストール (on Debian)

基礎文法編

■初歩的注意

- ▶ ※ 大文字と小文字を区別する言語である。
- ▶ ※ 変数に値が未設定の状態で変数の値を取得しようとするとエラーになる。ただし、要素数を決めた配列変数については、はじめから各要素に何らかの値が設定されている。
- ▶ ※ ガベージコレクション (GC) が常に働いている。

■データ型の種類

- 整数 byte short int long 300000L -2000001 (longの例)
- 小数 float double 30.5F -20.5f (floatの例)
- 真偽値 boolean true false
- 文字 char 'a' '亜'
- 文字列 String "Hello" "やあ"

■基礎

- ▶ ☆ 定型文
- ▶ コメントのしかた // で行末まで、あるいは /* */ で囲めば改行可能。
- ▶ 変数を定義 型名 hoge; か 型名 hoge = 値;
- ▶ 定数を定義 final 型名 HOGE = 値;
- ▶ 変数の参照を切る 変数名 = null

■標準入出力

- ▶ 1つの整数の入力受付 new java.util.Scanner(System.in).nextInt()
- ▶ 1行Stringの入力受付 new java.util.Scanner(System.in).nextLine()
- ▶ 出力 System.out.println(式);

■条件分岐	
▶ 条件分岐	
▶ 比較演算子	
▶ 論理演算子	
▶ 2 股分岐の略記	
▶ switch文	
■繰り返し処理	
▶ for文	
▶ while文	
▶ do-while文	
▶ 中断し、次へ・脱出	
▶ 意図的に無限ループ	
▶ for each	
■例外処理	
▶ 強制終了	
▶ 例外を投げる	
▶ 例外を受け取って処理	
▶ 例外の発生・非発生によらずある処理を実行	
■文字列	
▶ 特殊な文字を表現	
▶ 文字列の結合	
▶ 数値への変換	
▶ 文字数	
■数値	
▶ 2 8 16進数を表現	
▶ 数値の強制的な型変換	

■条件分岐	
▶ 条件分岐	if else if else
▶ 比較演算子	< <= > >= == != ※ただし文字列では 変数.equals("文字列")
▶ 論理演算子	&& ！ !()
▶ 2 股分岐の略記	条件式 = ? 真での値 : 偽での値
▶ switch文	switch (式) { case 値: 処理; break; default: 処理; }
■繰り返し処理	
▶ for文	for (int i = 0; i < 10; i++) { 処理 }
▶ while文	while (条件式) { 処理; 条件に関する処理; }
▶ do-while文	do { 処理; 条件の処理; } while (条件式); ※一度は必ず実行
▶ 中断し、次へ・脱出	continue; ・ break;
▶ 意図的に無限ループ	while (true) { 処理 } もしくは for (;;) { 処理 }
▶ for each	for (要素の型 好きな変数 : 配列) { 処理 }
■例外処理	
▶ 強制終了	exit; ← ??? 疑わしい System.exit(0); では ?
▶ 例外を投げる	try { … throw new 例外クラス名(引数あるかも); … } ※当然、throw～;部分を書かなくても、try等が起こればおのずと例外が投げられる。
▶ 例外を受け取って処理	catch (例外クラス名 \$e) { 何らかの処理※; exit; } ※ \$e->メソッド を使うことが多いだろう
▶ 例外の発生・非発生によらずある処理を実行	finally { 処理 } ※ catch のなかの exit; は消しておく！！
■文字列	
▶ 特殊な文字を表現	\" \' \\ \n
▶ 文字列の結合	+ ※代入演算子 += 使えます ※数値型との結合可能
▶ 数値への変換	Integer.parseInt("str")
▶ 文字数	str.length()
■数値	
▶ 2 8 16進数を表現	数値の先頭に 0b 0 0x をつける
▶ 数値の強制的な型変換	(型名) 数値 例) int age = (int) 3.2 ※数値どうしのみ

▶ 数値の自動的な型変換	
▶ カンマをつけない！	
▶ 算術演算子	
▶ 算術代入演算子	
▶ インクリメント・デクリメント演算子	
▶ 最大値・最小値	
▶ 0以上 n 未満の乱数	

■配列

▶ 配列を定義	
▶ 要素数	
▶ 要素の値を参照	
▶ 2次元配列	

■メソッド

▶ メソを定義	
▶ ※ 配列型やクラス型で引数を渡す場合、参照渡しになることに注意。	
▶ 戻り値を返す	
▶ 戻り値がない場合	
▶ メソを呼び出し	
▶ ※ <code>パケ.クラ</code> は完全限定クラス名、FQCNと呼ばれる。	
▶ FQCN省いて呼び出し	
▶ ※ <code>return</code> 文のあとに処理を書くと エラー になる。	
▶ ※ 仮引数の個数や型が異なれば、同じ名前のメソを複数作れる（＝オーバーロード）。	

■クラス

▶ ※ Javaのソースファイルの名前は、その内部で定義しているクラ名を用いて クラ名.java にしなければならない。	
▶ クラを定義	

▶ 数値の自動的な型変換	代入時；より大きな型になら代入OK ※int型だけ例外 演算時；より大きな型に統一されて演算
▶ カンマをつけない！	数値の自由な箇所に <code>_</code> はつけられる 例) <code>2_000_000</code>
▶ 算術演算子	<code>+</code> <code>-</code> <code>*</code> <code>/</code> <code>%</code> ※累乗は <code>Math.pow(底, 指数)</code> を使う
▶ 算術代入演算子	<code>=</code> <code>+=</code> <code>-=</code> <code>*=</code> <code>/=</code> <code>%=</code> ※多重代入 <code>a = b = 3</code> できるよ！
▶ インクリメント・デクリメント演算子	<code>a++</code> <code>++a</code> <code>a--</code> <code>--a</code> ※極力ほかの演算子と併用せず単独で
▶ 最大値・最小値	<code>Math.max(a, b)</code> <code>Math.min(a, b)</code> ※2つの数しか比較できない
▶ 0以上 n 未満の乱数	<code>new java.util.Random().nextInt(n)</code>

■配列

▶ 配列を定義	<ul style="list-style-type: none"> ・要素の型[] 配列名; 配列名 = new 要素の型[要素数]; ・要素の型[] 配列名 = new 要素の型[要素数]; ・要素の型[] 配列名 = new 要素の型[] {値1, 値2, ...}; ・要素の型[] 配列名 = {値1, 値2, ...};
▶ 要素数	配列.length
▶ 要素の値を参照	配列[n]
▶ 2次元配列	要素の型[][] 配列名 = new 要素の型[行数][列数];

■メソッド

▶ メソを定義	<code>public static 戻り値の型 helloWorld(String p1, int[] p2, ...) { .. }</code>
▶ ※ 配列型やクラス型で引数を渡す場合、参照渡しになることに注意。	
▶ 戻り値を返す	<code>return 値;</code>
▶ 戻り値がない場合	戻り値の型を <code>void</code> に
▶ メソを呼び出し	<code>メソ()</code> <code>クラ.メソ()</code> <code>パケ.クラ.メソ()</code>
▶ ※ <code>パケ.クラ</code> は完全限定クラス名、FQCNと呼ばれる。	
▶ FQCN省いて呼び出し	<code>import パケ.クラ</code> を冒頭に書けば <code>クラ.メソ()</code> と書け、 <code>import パケ.クラ.*</code> を冒頭に書けば <code>メソ()</code> と書ける。
▶ ※ <code>return</code> 文のあとに処理を書くと エラー になる。	
▶ ※ 仮引数の個数や型が異なれば、同じ名前のメソを複数作れる（＝オーバーロード）。	

■クラス

▶ ※ Javaのソースファイルの名前は、その内部で定義しているクラ名を用いて クラ名.java にしなければならない。	
▶ クラを定義	<code>public class Hoge { .. }</code>

- ▶ クラをパケに属させる
- ▶ ※ パケ名として `hoge.baa` のように `.` を使うこともあるが、**パケに親子関係 (階層関係) はない。**

- ▶ クラをパケに属させる `package` パケ; をクラのソースファイルの1行目を書く
- ▶ ※ パケ名として `hoge.baa` のように `.` を使うこともあるが、**パケに親子関係 (階層関係) はない。**