0-9章（復習）

講義目標

- プログラミングの基礎の習得
  - 問題解決、アルゴリズム、データ構造、デバッグ、 性能実測など
- プログラミング言語の基本的な枠組みの習得
  - データ、制御、抽象など
- 教科書
  - 結城浩、”Java言語プログラミングレッスン(上)(下)”、ソフトバンククリエイティブ、2007.

目次

0章: JAVA言語の見晴らし台

- Java言語: プログラミング言語の一種
  - プログラム = 機械を動かすための指示
  - OO、機種依存性低い、静的型付け、並行性、GC
- Javaプログラミングの流れ
  1. エディタでソースファイル X.java の記述
  2. javac X.java でコンパイルし X.class 生成
  3. java X で実行
  4. 必要に応じてデバッグ → 1. へ
- Javaプログラムの実行環境

1章: JAVAでこんにちは

- クラス(class)
  - Javaプログラムを構成する要素の単位
  - クラス名は慣習的に大文字で始める
  - トップのクラスXを定義するソースファイル名: X.java
- mainメソッド
  - プログラムの開始点(エントリーポイント)
- System.out.println
  - 文字列を表示して改行
  - 文字列は二重引用符“ “でくくる
  - 二重引用符そのものは ¥”と書く
- 演習:Study.java

```
public class Study {
 public static void main(String[] args) {
    System.out.println("こんにちは。");
    System.out.println("私は ¥"Java¥" を勉強している。");
 }
}
```

2章: 計算をやってみよう

- 加減乗除・剰余(算術演算子)+ - * / %
  - 整数の除算において小数部分は切り捨て
  - 括弧 ( ) による優先順位変更
  - オーバーフローの可能性に留意
- 整数の定数
  - int型 32ビット: -2147483648〜2147483647
  - long型 64ビット: 末尾に L
    - 9223372036854775808L〜9223372036854775807L
- 文字列の連結: +
- 演習 Calc.java

```
public class Calc {
 public static void main(String[] args) {
    System.out.println("(7+8)/7=" + ((7 + 8) / 7));
    System.out.println("(7+8)%7=" + ((7 + 8) % 7));
    System.out.println("100000*100000=" + (100000L * 100000L) );
 }
}
```

3章: 変数と型: 使い方

- 変数: 値を入れておく箱
    1. 宣言: 型 変数名;
    2. 代入: 変数名 = 値;
        1. 初期化(宣言と代入を同時): 型 変数名 = 値;
    3. 参照(代入や初期化の後で): 変数名
- 変数の型: 誤りを防ぐための箱の種類
    - 基本型(primitive type): 論理値 `boolean`、 整数 `char`, `byte`, `short`, `int`, `long`、 浮動小数点数 `float`, `double`
    - 参照型(reference type): クラス、インタフェース、配列、列挙型
- 演習: `VarInt.java`

```java
public class VarInt {
 public static void main(String[] args) {
    int x = 7 + 8;
    int y = 2;
    y = x / y;
    System.out.println(y);
 } }
```

3章: 変数と型: 基本型の種類 (1)

基本型	種類	サイズ	値の範囲	定数
boolean	論理	(1bit)	true, false	true, false
char	整数/文字	符号なし16bit	0〜$2_{16}-1$	‘x’ 0
byte	整数	符号付き8bit	-2_{7}〜$2_{7}-1$	0
short	整数	符号付き16bit	-2_{15}〜$2_{15}-1$	0
int	整数	符号付き32bit	-2_{31}〜$2_{31}-1$	0
long	整数	符号付き64bit	-2_{63}〜$2_{63}-1$	0L
float	浮動 小数点数	単精度32bit	1.401298E-045〜 3.402823E+038	0.0F 314E-2F
double	浮動 小数点数	倍精度64bit	4.9406564584124654E-324〜 1.7976931348623157E+308	0.0 314E-2

3章: 変数と型: 基本型の種類 (2)

- 演習: VarDouble.java

```
public class VarDouble {
 public static void main(String[] args) {
    double x = 7 + 8;
    double y = 2.0;
    y = x / y;
    System.out.println(y + 314E-2);
 }
}
```

3章: 変数と型: キーボードからの文字列入力

- 標準入力からの文字列行入力
  1. System.inをラッピングしてBufferedReaderの作成
  2. BufferedRearderに対してreadLine
- コメント: プログラマのための説明文

```
// 一行コメント
/* 通常コメント */
/** ドキュメンテーションコメント */
```

- 演習: KeyboardInput.java

```
import java.io.*;
public class KeyboardInput {
 public static void main(String[] args) {
    BufferedReader reader = new BufferedReader(new InputStreamReader
    try {
       String line = reader.readLine(); // 文字列の読み込み
       int number = Integer.parseInt(line); // 整数への変換
       System.out.println("入力された整数: " + number);
    } catch (Exception e) {
       System.out.println(e);
    } } }
```

4章: IF文、比較・論理演算子

- if文: 分岐、二者択一(の繰り返し)

```
if (条件式) { 条件成立時 }
if (条件式) { 条件成立時 } else { 不成立時 }
if (条件式A) { A成立時 } else if(条件式B) { Aが不成立でB成立時 }
```

- 条件式: boolean 型、値は true または false
  - 比較演算子: == , != , >= , <= , > , <
  - 短絡論理演算子: 論理積(かつ)&&, 論理和(または) ||
- 演習: IfStatement.java

```
public class IfStatement {
 public static void main(String[] args) {
    int x = 60;
    if(x >= 0 && x <= 100) {
        if(x >= 80) {
           System.out.println("合格");
        } else {
           System.out.println("不合格");
        }
    } else {
        System.out.println("エラー");
    } } }
```

5章: SWITCH文

- switch文
  - 多数から選択
  - 分岐数が多ければ高速可能性
  - 整数式の型は char, byte, short, int のみ
  - breakがなければ以降もcaseを無視して続行

```
switch(整数式) {
case 定数式1:
 // 評価結果が定数式1と等しい場合
 break; // switch文の終了
 ...
default:
 // 全caseが満たされない場合
 break;
}
```

- 演習: SwitchStatement.ja

```
public class SwitchStatemer
 public static void main(St
    char answer = 'n';
    switch(answer) {
       case 'y':
          System.out.print
          break;
       case 'n':
          System.out.print
          break;
       default:
          System.out.print
          break;
    }
 }
}
```

6章: FOR文、変数のスコープ

- for文: 繰り返し、ループ

```
for (初期化; 条件式; 次の一歩) { 繰り返す処理 }
```

  - 初期化: 最初に一度のみ
  - 条件式: 繰り返しの条件
  - 次の一歩: 繰返しを進める処理
- 変数の有効範囲(スコープ)
  - 宣言したブロック内のみ、通常{...}内
- 演習: ForStatement.java

```java
public class ForStatement {
 public static void main(String[] args) {
    for (int i = 0; i < 10; i++) {
       System.out.print(i + ":"); //改行なし文字列表示
       for (int j = 0; j < i; j++) {
          System.out.print("*");
       }
       System.out.println("");
    }
 }
}
```

7章: WHILE文とSTRING型: WHILE

- while文: 0回以上の繰り返し、初期化・次の一歩の部分なし

```
while (条件式) { 繰り返す処理 }
```

- do-while文: 1回以上の繰り返し

```
do { 繰り返す処理 } while (条件式);
```

- for, while, do-while共通
  - break で繰返しを抜ける。
  - continueで繰返し中の以降をスキップし、次の繰り返し。
- 演習: WhileStatement.java

```
import java.io.*;
public class WhileStatement {
 public static void main(String[] args) {
    BufferedReader reader = new BufferedReader(new InputStreamReader
    try {
       String line;
       while ((line = reader.readLine()) != null) {
          System.out.println(line);
       }
    } catch (Exception e) {
       System.out.println(e);
    } } }
```

7章: WHILE文とSTRING型: STRING

- Stringクラス: 書き換え不可の文字列
  - 文字の置換: replace、 部分文字列: substring
  - 小/大文字化: toLowerCase / toUpperCase
  - 全て「新しい」文字列オブジェクトを返す
- StringBufferクラス:書き換え可能な文字列

7章: WHILE文とSTRING型: STRING (2)

- 演習: WhileAndString.java

```
import java.io.*;
public class WhileStatement {
 public static void main(String[] args) {
    BufferedReader reader = new BufferedReader(new InputStreamReader
    try {
       String line;
       while ((line = reader.readLine()) != null) {
          s = line.toLowerCase();
          System.out.println(s);
       }
    } catch (Exception e) {
       System.out.println(e);
    }
 }
}
```

8章: メソッド

- メソッド(method): 処理のまとまり

```
戻り値の型 メソッド名 (引数列) {
    処理
    return 戻り値;
}
```

  - 戻り値が不要な場合の戻り値の型: void
  - public: どこからでも利用可
  - static: インスタンスを作らず利用可
- 演習: Method.java

```java
public class Method {
 public static void main(String[] args) {
    System.out.println(getPower(2, 8));
 }
 public static int getPower(int base, int exponent) {
    int power = 1;
    for (int i = 0; i < exponent; i++) {
       power = power * base;
    }
    return power;
 }
}
```

9章: 配列

- 配列(array): 同種変数の番号付られた並び
    1. 宣言: 要素型[] 変数名;
    2. 確保: 変数名 = new 要素型[個数];
    3. 代入: 変数名[添字] = 要素;
    4. 初期化(宣言、確保、代入を同時)
        - 要素型[] 変数名 = { 0番目要素, . . . , N番目要素 } ;
        - (途中で)変数名 = new 要素型[]{ 0番目要素, . . . , N番目要素 } ;
    5. 参照(代入や初期化の後で): 配列名[添字]
- 添字(index): 配列中の番号、0番スタート
- 配列の要素数: 配列名.length

9章: 配列 (2)

- 演習: Array.java

```java
public class Array {
 public static void main(String[] args) {
    int[] values = new int[]{ 62, 90, 75 };
    int sum = 0;
    for (int i = 0; i < values.length; i++) {
       sum += values[i];
    }
    System.out.println(sum);
 }
}
```

9章: 配列: 二次元配列

- 二次元配列: 配列の配列
  - 内部の配列の要素数は一定でなくてよい
  - 三次元以上の多次元も可能
- 演習: Array2.java

```
public class Array2 {
 public static void main(String[] args) {
    int[][] vs = new int[][]{
       { 62, 90, 75 },
       { 100, 0, 50 },
       { 30, 40 }
    };
    int sum = 0;
    for (int i = 0; i < vs.length; i++) {
       for (int j = 0; j < vs[i].length; j++){
          sum += vs[i][j];
       }
    }
    System.out.println(sum );
 }
}
```

0-9章のまとめ（復習）

- Javaの仕組み
  - 仮想マシン、クラス、ソースファイル、クラスファイル
  - mainメソッド
- 演算子
  - 算術演算子: +, -, *, /, %,
  - 比較演算子: ==, !=, <, >, >=, <=
- 変数と型
  - 基本型(primitive type): boolean, char, …
  - 参照型(reference type): String, 配列, …
- 制御構文
  - 条件分岐: if, switch
  - 繰り返し: for, while, do-while
- メソッド
- 配列