情報処理概論
(Basic Theory of Information Processing)

第２回：Javaプログラムの作成
(eclipseのインストール)

概要

- Hello World!
- eclipse
- 各種Javaプログラム
- 演習

3　Javaプログラムの作成

3.1　Hello World!

- Hello World と表示するだけ。
- プログラムの実行環境が正しいかどうか確かめることができる。

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

- http://www.oracle.com/technetwork/java/javase/downloads/index.html から，Java Platform (JDK) 8u(数字) をダウンロードして，インストールする。
- 環境変数Pathに実行ファイルのあるフォルダを加える。

コンパイル(HelloWorld.classというファイルができる．)

```
javac HelloWorld.java
```

実行

```
java HelloWorld
```

3.2 Eclipse

- IBMが開発した**統合開発環境 (IDE, Integrated Development Environment)**
  - ブログラムの作成・管理
  - ブログラムの実行・デバック (間違い(バグ，虫)を取り除くこと)
- Eclipse Public License
- Javaで書かれている。
- eclipseが扱うことができる言語：

  Java, FORTRAN, C, C++, perl, PHP, ...

  など，多数の言語に対応している。
- プログラムのデバッグが容易になる。
  - コーディィング時にいろいろと示唆する。
  - 実行時に途中で止めることができる。
- 最近の人は，使うのが当たり前。
  昔の人(私など)は，普通にエディタでプログラムを書いて，デバックは計算途中の値をを書き出して動きをチェックした。
- eclipseを使って課題を行う。

3.3 Eclipseのインストール (Windows)

- ○日本語版が良い人：`http://mergedoc.sourceforge.jp`
  において，Eclipse 3.6 Heliosのボタンをクリックし，Full All in One (JRE あり)のJava のDownloadのボタンをクリックしてダウンロードする
  (現在の最新バージョンは4.3であるが，計算機室は3.6)。
- ダウンロードしたファイルを解凍すると，eclipseというフォルダが作成される。
- そのフォルダの中のeclipse.exeをクリックすれば，eclipseが実行される。
- インストールしたeclipseのフォルダに，eclipse.iniというファイルがあるので，それをエディタ(**メモ帳では改行コードの関係で編集できない。**K2Eなどのエディタをインストールする) で，文字コードUTF-8で動くように，

  `-Dfile.encoding=utf-8`

  を最後の行に追加する。
- Linuxの場合は，各ディストリビューションのパッケージに入っているので，パッケージ管理ツールでインストールできる(UTF-8で動く)。
- **Macの学生は各自調べること(使えるようではあるけれど)。**

4　Javaプログラミング

4.1　EclipseでHello World

1. eclipseを立ち上げる
2. 左にあるパッケージ・エクスプローラウィンドウの空いているところで右クリックして,「新規」→「Java プロジェクト」を選択
3. プロジェクト名「HelloWorld」入力して，完了をクリック
4. パッケージ・エクスプローラにプロジェクト HelloWorldが表示されている。
5. 右三角をクリックして，中身を表示する。
6. srcの上で右クリックして,「新規」→「クラス」を選択
7. クラス名「HelloWorld」を入力して，完了をクリック
8. 右側に「HelloWorld.java」をエディットするためのウィンドウが開く。
9. 先のプログラムを入力する。
10. ツールバーの緑の円の中に右三角形があるアイコンをクリックする。
11. 実行形式を聞かれたら，Java アプリケーションを選択する。
12. 下の「コンソール」というウインドウに，HelloWorldが表示される。

4.2　細かいところは無理だけど，HelloWorld.javaをざっと見てみると

```
public class HelloWorld {
  public static void main(String[] args) {
    System.out.println("Hello World");
  }
}
```

- プログラムの中では，{ と }ではさんで，一つの固まりを表す。
  プログラムを構造(木構造)化することができる。
- はじめの，class HelloWorld {から，最後の}までで，HelloWorldというクラスを宣言している。
- クラスは，データ構造とそれに対する処理を記述するひとつの「かたまり」である。
- static public void main(String[] args) {から，対応する}までで，mainというメソッドを宣言している。
- メソッドの宣言で，クラスの中で処理を定義している。
- System.out.println("Hello World")によって，他のメソッドを呼び出している。
- System.out.printlnは，コンソールに文字列を表示するためのメソッドである。
- プログラムの中で，文字列を"と"で囲むと，それが文字列のデータとして扱われる。

4.3　コメント

- プログラム中に書く説明文
- 他人がプログラムを読んだときに分かり易いようにする。
- しっかりとコメントを書かないと，あとで自分でも忘れてしまうことが多い。
  (私が歳だからというわけではない。)
- デバッグなどのために，プログラムの文を一時的に無効にするためにも使われる。
  (**コメントアウト**)

コメントの書き方

- //　それ以降の1行がコメントになる.
- /*　から　*/までがコメントになる．複数行のコメントが可能.
- /**　から　*/までがコメントになる．クラスの註釈などに用いる.
  javadoc コマンドを用して，クラスの注釈を自動的に抽出して，
  html文書にすることが可能である。

コメントの例

```
// １行ずつコメントが書けます。

package src2;

/* 複数行にわたって書きたいときには
このようにします。
そうすれば，何行でも書けます。
*/

/*
いいか良く聞け．ノーコメントだ！
system.out.println("(・o・;)")
*/
```

```
/** これは，単なるコメントではなく
 * Javadocで使われます。
 * 自動的にhtmlでまとめたものを，生成してくれます。
 */
public class Comment {
  /**  main メソッドの説明です。
   * @param args 入力パラメター
   */
  public static void main(String args[]) {
    // プログラムの中にも書けます。
    System.out.println("こんにちは，みなさん。");
    /* またコメントです。
     * この最初の*は，見やすいようにエディタが勝手につけたものです。
     */
    System.out.println("それでは，さようなら。");
  }
}
// プログラムの後に書いても大丈夫です。
```

4.4　これから勉強していくこと

- **変数**：データを格納し，取り出すことができる。
- **演算**：データを処理する。
  　四則演算，関数演算，文字列処理
- **制御構造**：条件分岐，ループ文
  - 条件分岐：もし～ならば，～を実行する。
  - ループ文：～が成立する間は，～を繰り返し実行する。
- **メソッド**：
  - 同じような処理(手続き)を何回も行うことがある。
  - 手続きをまとめて記述し，その手続きを使うときは，１つの関数として呼び出す。
- **クラス**：
  - データの構造を定義することができる。
    　テーブル，木
  - 構造化されたデータとそれに対する処理を「ひとかたまり」(オブジェクト)にして扱う。
  - クラスはオブジェクトの設計図。

4.5 System.out.println()

- **出力**しないと，せっかく処理した意味がない。
- `System.out.println(文字列)`で文字列を表示する。
- 数も文字列に変換して，表示できる。

  ```
  System.out.println(10);
  ```

  という文により，

  `10`

  が表示される。
- `+`を使って，文字列を繋げることができる。たとえば，
  `"xは" + 3 + "に等しい"` は，文字列`"xは3に等しい"`と同じ。
- 例(Println.java)，

  ```
  System.out.println("xは" + 3 + "に等しい。");
  ```

  という文により，

  `xは3に等しい`

  が表示される。

4.6　変数

- 変数に値を**代入**することができる。

```
x = 2;
y = x + 3;
```

と書けば，yには5が代入される。

- System.out.println(文字列)と組み合わせて，

```
x = 2;
y = x + 3;
System.out.println("xは" + x + "に等しい。yは" + y + "に等しい。");
```

というプログラムにより，

xは2に等しい。yは5に等しい。

と表示される。

4.7　キーボードからのデータの入力

- キーボードからデータが入力できれば便利である。
  (ファイルからの読み込みとファイルへの書き込みは後で講義する予定。)
- 説明は難しいので，プログラム例だけ示す。
- 文字列の読み込み

```
import java.util.Scanner;

public class KeyInput {
  public static void main(String args[]){
    Scanner stdIn = new Scanner(System.in);
    System.out.println("お名前は？");
    String inStr = stdIn.next();  // 文字の読み込み
    stdIn.close();
    System.out.println(inStr + "さん，はじめまして。");
  }
}
```

- 変数inStrに，キーボードから入力した文字列が格納される。

- 数字を読み込んで加算する。

```
import java.util.Scanner;

public class KeyAdd {
  public static void main(String args[]){
    Scanner stdIn = new Scanner(System.in);
    System.out.println("x + y を計算します, xを入力してください。");
    int x = stdIn.nextInt();  // 数字の読み込み
    System.out.println("yを入力してください。");
    int y = stdIn.nextInt();  // 数字の読み込み
    stdIn.close();
    System.out.println(x + " + " + y + " = " + (x + y));
  }
}
```

- 変数xとyに，キーボードから入力した整数が格納される。

4.8 制御構造

- 条件文 (if 文, else 文)
- ループ文の例：1から10まで加算する。

```
sum = 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10;
```

または,

```
sum = 0;
sum = sum + 1;
sum = sum + 2;
sum = sum + 3;
sum = sum + 4;
sum = sum + 5;
sum = sum + 6;
sum = sum + 7;
sum = sum + 8;
sum = sum + 9;
sum = sum + 10;
```

とすれば，1から10まで加算することができる。

- 1000までならばどうする？　そのためには，

```
sum = sum + x;
```

という文を，xを1から1000まで順番に変えて，実行すればよい。

- そのために，for文を使う。

```
sum = 0;
for (x = 1 ; x <= 1000 ; x = x + 1) {
  sum = sum + x;
}
```

- もう少し複雑な問題として，入力した数が素数かどうか調べてみる。
  - **素数：1とそれ自身でしか割れない数**
  - prime：入力した素数の候補
  - nDivideded：割り切れた回数(これが2ならば素数)
  - diverder：割ってみる数
  - prime % divider：primeをdividerで割った余り。
  - diverderを1からprimeまで変化させ，primeを割ってみる。
  - 割り切れたdiverderの値が2種類ならば，primeは素数

```
public class Prime {
  public static void main(String args[]){
    int nDivided, divider;
    Scanner stdIn = new Scanner(System.in);
    System.out.println("素数かどうか判定する数を入力してください");
    int prime = stdIn.nextInt();  // 数字の読み込み

    nDivided = 0;
    for (divider = 1 ; divider <= prime ; divider = divider + 1) {
      if (prime % divider == 0) {
        System.out.println(prime +"は" + divider + "で割り切れました。");
        nDivided = nDivided + 1;
      }
    }
    if (nDivided == 2) System.out.println(prime + "は，素数です。");
    else System.out.println(prime + "は，素数ではありません。");
  }
}
```

4.9 メソッド

- 何度も利用する処理手続きをまとめて，**関数のように呼び出す**ことができるようにする。
- 素数判定を1つのメソッド`chkPrime(int prime)`とする。
- 素数の場合はメソッドの戻り値として`true`を返し，素数でない場合は`false`を返す。

```
// 引数primeが素数かどうか判定するメソッド
public static boolean chkPrime(int prime) {
  int divider, nDivided;
  nDivided = 0;
  for (divider = 1 ; divider <= prime ; divider = divider + 1) {
    if (prime % divider == 0) {
      nDivided = nDivided + 1;
    }
  }
  if (nDivided == 2) return true;
  return false;
}
```

- chkPrime(int prime)を使って，2から100までの素数を調べるプログラム

```
public class PrimeMethod {

  public static void main(String args[]){
    int prime;
    for (prime = 2 ; prime <= 1000 ; prime = prime + 1) {
      if (chkPrime(prime)) {
         System.out.println(prime + "は，素数です。");
      }
    }
  }

  // 本来はメソッドchkPrime()はここに書く。
}
```

4.10 配列

- 同一種類のデータを並べ，整数インデックスでデータを参照できる。
  a[2]，b[3][8] が変数になる。
- インデックスに変数を代入することができる：a[i] = 3，b[i][j] = 5
- 数学的な例：ベクトル，行列

```
public class IntArray {
  public static void main(String args[]){
    int i, j;
    int a[] = new int[5];
    for (i = 0 ; i <= 4 ; i = i + 1) {
      a[i] = i * i * i;
    }
    for (j = 0 ; j <= 4 ; j = j + 1) {
      System.out.println("a[" + j + "] = " + a[j]);
    }
  }
}
```

- $a = (0, 1, 8, 27, 64)$

4.11 クラスとオブジェクト

- プログラム = データ構造 + アルゴリズム
- 構造をもつデータを処理していく。
- 学生のデータ (StudentData)
- データ構造 (複数の情報の組み合わせ)
  - StdNo：学籍番号
  - name：名前
  - pJapa：国語の点数
  - pMath：数学の点数
  - pEngl：英語の点数
  - pAvel：３科目の平均点
- 処理 (メソッド)
  - print()：データを表示する。
  - calAve()：平均点を表示する。
- クラス (設計図) をもとに，具体的にデータをもつ**オブジェクト**を作成する。
- **オブジェクト指向のプログラミング**では，プログラムの実行は，オブジェクトのデータを出し入れしたり，オブジェクトのメソッドを実行させることによって行われる。

```
public class StudentData {
  // フィールド(オブジェクトのデータ)
  int stdNo;
  String name;
  int pJapa, pMath, pEngl;
  double pAve;

  // コンストラクター（メソッドの一種でオブジェクトを作成する。）
  StudentData(int stdNo, String name, int pJapa, int pMath, int pEngl) {
    this.stdNo = stdNo;
    this.name  = name;
    this.pJapa = pJapa;
    this.pMath = pMath;
    this.pEngl = pEngl;
  }
```

```
  // データを表示するメソッド
  void print() {
    System.out.println(stdNo + " " + name + " 国語" + pJapa + "点, 数学"
                    + pMath + "点, 英語" + pEngl + "点, 平均" + pAve + "点");
  }
  // 平均点を計算するメソッド
  void calAve() {
    pAve = (pJapa + pMath + pEngl) / 3.0;
  }
}
```

- クラスStudentDataの使用例

```
public class StudentEx {
  public static void main(String args[]){
    StudentData yamashita = new StudentData(2, "山下", 80, 90, 70);
    StudentData kawakami = new StudentData(3, "川上", 90, 70, 90);
    yamashita.print();
    kawakami.print();
    yamashita.calAve();
    yamashita.print();
  }
}
```

4.12　演習

- 自分の学籍番号と名前を表示する。
- クラス`KeyInput`と`KeyAdd`を作成する。
- 1から10まで加算してみる(ループを使わない2つの方法)。
- コメントを使ってみる。
- `System.out.println("1 + 2 = " + 1 + 2)`と
  `System.out.println("1 + 2 = " + (1 + 2))`の違いを確かめてみる。

4.13　今日の最後に

- プログラミングは**習うより慣れろ**の部分があるので，自分でプログラムを書いてみることが大切。
- **コンピュータは，知的な鏡**とも言われている。
- プログラミングのスキルは，情報処理分野のプロでなくても，**実験データの整理，Excelのマクロ**などによる文系的なデータ処理にも使える。