「オブジェクト指向プログラミング」追加補足

Last Compiled by LaTeX on 2016/04/19 at 11:59:00

全般

- レポートの書式は特にありません。Word などで作っても LaTeX で作っても良いです。
- Eclipse ではソースコードをドラッグして選択して、右クリック → ソース → フォーマットを選ぶと、文法に従って、見やすいようにソースコードを整形してくれます。インデントが崩れたら試してみてください。
- Eclipse でソースファイル (.java) は画面左の「パッケージエクスプローラー」のプロジェクト名の下の「src」というところに置く必要があります。「main 関数がみつからない」のようなエラーが出る場合、確認してみてください。
- **変数のスコープ**：以下のように変数の宣言文 (int i) の位置によって、変数が参照できる範囲 (変数のスコープ) が決まります。以下の例では変数 i も j も for 文の外では参照できません。

```
for(int i=0; i<array1.length; i++){ // for 文の中で変数 i が宣言 (int i) されている
    int j = i+1; // 変数 j の宣言文
    System.out.println(i);
    System.out.println(j);
}
System.out.println(i); // エラー
System.out.println(j); // エラー
```

- `System.out.println` は引数が一つだけのメソッドです。
- **注意！** p.18 `System.in` にひもづいた `java.util.Scanner` の close

```
Scanner s1 = new Scanner(System.in);
int i = s1.nextInt();
s1.close();
Scanner s2 = new Scanner(System.in);
int j = s2.nextInt();
s2.close();
```

は一見問題なさそうに見えますが、エラーになります。s1 の close() の段階で、s1 にひも付いている System.in も close() されてしまうので、標準入力が閉じてしまうためです。これに対する最も簡単な解決法は「System.in にひも付いた Scanner はもう標準入力が要らないのでない限り close しない」というものです。(しかも System.in, System.out, System.err は reopen できない) もっとちゃんとやるには、

- System.in を保護するラッパークラス (そのクラスの close が呼ばれても System.in の close が呼ばれないように明示的に書く) を作成し、Scanner はそれにひも付ける。
- System.in にひも付ける Scanner を複数作らないような設計とする。

などが考えられますが煩雑なので演習でこの問題に出会ったら close しない方針で構いません。Java ではその Scanner を保持するオブジェクトがどこからも参照できなくなったら自動的にこうしたオブジェクトは片付けられる仕組みがあります。

ヒント

- p.22 ArrayList の要素に対するループの書き方 4 種類:

```
ArrayList<Float> a = new ArrayList<>();
Scanner scanner = new Scanner("0.12 0.23 -1.5");
while(scanner.hasNextFloat()){
    a.add(scanner.nextFloat());
}
// (1) 拡張 for
for(float x : a){
    System.out.println(x);
}
// (2) 標準 for
for(int i=0; i<a.size(); i++){
    System.out.println(a.get(i));
}
// (3) Iterator
Iterator<Float> it = a.iterator();
while(it.hasNext()){
    System.out.println(it.next());
}
// (4) forEach ラムダ式！
a.forEach( (x)->{System.out.println(x);} );
```

- p.37 課題 1.9 「Judge」クラスの作り方 (1):
  - 課題をはじめる前にまず p.35-36 の「カプセル化、アクセサ、Getter/Setter」を読みましょう。Setter が何をするかわかりましたか? わかったら、課題にとりかかりましょう。
  - class は C 言語の構造体みたいにいくつかの変数をパックして持てます。それに加えて、その変数に対する処理操作 (メソッド) もパックされているデータ構造と考えてもよいでしょう。
  - 例えば次の MyFirstClass クラスのインスタンス a は、int, float double, String, Scanner の値を保持しており、a.i, a.f, a.d, a.name, a.s などでアクセスできます (public なので)。このようにいくつかの変数 (i,f,d,name,s) と操作 (run(),setValues(...)) をパックした雛形が MyFirstClass というクラスになります。

```
import java.util.Scanner;
public class MyFirstClass {
    public int i;
    public float f;
    public double d;
    public String name;
    public Scanner s;
    public void setValues(int i, float f, double d){
        this.i = i; this.f = f; this.d = d;
        this.s = new Scanner(System.in);
    }
    public void run(){
        System.out.println(i+' '+f+' '+d);
        System.out.print("Input an integer:");
        System.out.println(this.s.nextInt());
    }
    public static void main(String[] args){
        MyFirstClass a = new MyFirstClass();
        a.setValues(1, 3.3f, 4.56);
        a.run();
    }
}
```

- p.37 課題 1.9 「Judge」クラスの作り方 (2):
  - Judge クラスが指定された main メソッドでどのように「使われているか」を観察して、この部品 (クラス) にどういう動作 (メソッド) が仮定されているかを探ります。
  - Judge クラスはまず、引数として名前の文字列をうけとる「コンストラクタ」を持っている必要があります。これは RandomJankenPlayer と同じなので真似すれば良さそうです。
  - 次に Judge クラスには、setPlayers というメソッドが必要みたいです。このメソッドは何も値を返していないようなので、返り値は void で良いでしょう。引数は 2 つとっていて、RandomJankenPlayer 型の変数を 2 こ受け取るようです。
  - さて、上の setPlayers の入力 (引数) と出力 (返り値) はわかりましたが、中身に何を書けば良いでしょう?[1] よくわからないので、空にしておきます。
  - 次に、Judge クラスには play というメソッドが必要みたいです。引数としては int 型の数字 num「のみ」を受け取っています。課題の実行結果を見ると、どうやらこの数字はじゃんけんの回数で、ユーザーが入力する数字のようです。つまり、play が呼ばれると setPlayers で受け取った二人に num 回のじゃんけんを行わせ、勝敗を計数し、それを指定のかたちで出力すれば良いようです。
  - play では num しか引数にとっていません。ということは、対戦させる二人は setPlayers で受けとって**このクラス自身が保持しておく必要があります**。クラスはフィールドとして変数を保持できたのを思い出しましょう。RandomJankenPlayer は String name を持っていました。Judge には RandomJankenPlayer のフィールドが 2 つ必要で、setPlayers では受け取ったプレイヤーをこのフィールドに格納する setter の動作をさせれば良さそうです。
  - 以上の分析をまとめると、(1)Judge クラスは 3 つのフィールド (String を 1 こと RandomJankenPlayer を 2 こ) を持ち、(2) コンストラクタで String を 1 つ受け取りフィールドに set、(3) setPlayers メソッドで RandomJankenPlayer を 2 つ受け取りフィールドに set、(4) play メソッドで、フィールドに保持した RandomJankenPlayer のインスタンスの showHand を必要な回数呼び出し、num 回対戦させて、勝ち負け判定を行い、それを記録して指定の形の出力、というクラス要件が出てくるでしょう。
- p.38 課題 1.11 「審判はその都度、各プレイヤーに勝敗結果を通知するのみとし、プレイヤーのクラスのインスタンスが自身の勝ち、負け、引き分けの数を保持するようにせよ」の意味
  1. RandomJankenPlayer p1, p2 の勝敗を管理する Judge j は次を行う。
  2. Hand h1 = p1.showHand(); Hand h2 = p2.showHand(); で手を出してもらう。
  3. p1 と p2 のどっちが勝ったか判定する。
  4. 勝敗を通知する。たとえば p1 が勝って、p2 が負けたときは、例えば、p1.isWin(); p2.isLose(); みたいに教える。
  5. なので RandomJankenPlayer には int win, lose, draw のようなフィールドと、isWin(), isLose(), draw() のようなメソッドを持たせ、コンストラクタで win, lose, draw を初期化し、メソッドでフィールド値をインクリメントする。対戦終了後、勝ち数、負け数を取得するため、win, lose, draw には getter も定義しておく。
  6. つまり、RandomJankenPlayer はだいたい次のような感じになります。

[1] 実は名前から「Player」のセッターになっていることが類推できます。

```
public class RandomJankenPlayer {
    String name;
    // 必要なフィールドを追加 (getter/setter も作る)
    int win;
    int lose;
    int draw;
    public RandomJankenPlayer(String name){
        // new 時に呼ぶコンストラクタでフィールド値を初期化
        this.win = 0; this.lose = 0; this.draw = 0;
        this.name = name;
    }
    public Hand showHand(){
        // 中身 省略
    }
    // Judge が勝敗を伝えるとき呼んでもらうメソッド
    public void isWin(){ this.win++; }
    public void isLose(){ this.lose++; }
    public void draw(){ this.draw++; }
      :
    // 以下、省略 (getter/setter も忘れずに作る)
}
```

- try-catch 文によってエラーが表示されないためデバッグがしづらい場合、以下のように catch に printStackTrace を入れてください。どこでエラーが出たかなどのエラーメッセージが表示されるようになります。

```
try{
    ...
}catch(Exception e){
    e.printStackTrace();
}
```

修正・追加

- p.13 課題 1.1 もし、「double[] array2 = array;」とした場合、基本的には array2 を書き換えると array の内容も書き換わります。上の文では「array」が保存されているメモリ上の場所 (アドレス) が「array2」にコピーされるので、「array2」で参照すると結局「array」と同じところを見に行ってしまいます。

- p.17 java.util.Calendar では月に対応する数字がひとつずれています。例えば、1 月は「0」、2 月は「1」などとなります。なので、プログラム中では定数を使ったほうがよいかもしれません。API リファレンスで java.util.Calendar を調べると、最初のほうの「フィールド」に「static int」で定義された「英数大文字」の「定数」がいろいろとあるのがわかります。ここで、例えば「1 月」を参照するには「JANUARY」を見ます (正確には「java.util.Calendar.JANUARY」で参照するか import する)。クリックすると「関連項目」に「定数フィールド値」というのがあり、どういう値で定数が定義されているかみることができます。「定数フィールド値」のリストを見てみると、「JANUARY」の場合「0」となっているのがわかると思います。

- p.19 ダウンロードしたファイル「table.dat」をどこに保存すれば良いか?

  - Eclipse で「table.dat」を使うプロジェクトにドラッグアンドドロップする。たとえば「Kadai1」なら、Eclipse のパッケージエクスプローラーの「Kadai1」の上にドラッグアンドドロップすると「table.dat」で参照することができる。
  - 保存した「table.dat」を右クリック → プロパティの「場所」の情報をつけた完全パスで指定する。例えば、場所が「`/home/staff/takigawa/`デスクトップ」なら

    ```
    Scanner scanner = new Scanner(new File("/home/staff/takigawa/デスクトップ/table.dat"));
    ```

- p.5 「`java -verbose:class HelloWorld | grep Loaded | wc -l`」の最後は「−1」ではなく「-l(ハイフン・エル)」です。

- p.22 Float/float(単精度浮動小数点型) と Double/double(倍精度浮動小数点型) があります。今回の演習ではどちらでも構いませんが、科学技術計算・数値計算の場合、倍精度のほうが精度落ちが少なくてすみます。一方、精度はそんなにいらないから速く処理したい場合は単精度で十分な場合もあります。大文字ではじまる Float や Double はクラス、小文字で始まる float や double は型です。なぜ同じ対象に二つの表現があるのかについては p.23 の「型、ラッパークラス、オートボクシング」を参考にしてください。

- p.22 課題 1.5 scanner2.nextFloat() というメソッドは scanner が持っている文字列から「float(単精度浮動小数点型)」に相当する部分を一つ持ってこい、という意味で、このメソッドが呼ばれるたびに scanner が保持している文字列は減っていくので気をつけてください。必要なら「float x = scanner2.nextFloat();」と一旦変数に格納するなどしてください。

- p.22 課題 1.5 scanner2.nextFloat() と scanner2.hasNextFloat() の関係: たとえば、次の例を考えてみます。scanner は new 時に保持した文字列から nextFloat で数字を一つ読みます。一つ読み込むと文字列は hasNextFloat はさらに文字が読めるかを TRUE/FALSE で判定します。

```
Scanner scanner = new Scanner("0.12 0.23 -1.5");
boolean result;
float x;
result = scanner.hasNext();
System.out.println(result); // true
x = scanner.nextFloat();    // "0.12 0.23 -1.5" -> "0.23 -1.5"
System.out.println(x);      // 0.12
result = scanner.hasNext();
System.out.println(result);
x = scanner.nextFloat();    // "0.23 -1.5" -> "-1.5"
x = scanner.nextFloat();    // "-1.5" -> ""
System.out.println(x);      // -1.5
result = scanner.hasNext();
System.out.println(result); // false
```

- p.22 課題 1.5 「ArrayList を用いて」とありますが、この課題自体は配列や ArrayList を使わなくても (というか使わないほうが) 解ける簡単な例なため、題意がわかりにくいと思います (すまん)。基本的には ArrayList を使ってみる、という演習です。なので、ArrayList を使ってもらえればどう使ってもらっても良いのですが、例えば、2 列目の値を全部格納した array2 と 3 列目の値を全部格納した array3 を ArrayList で作成し、ArrayList を引数として平均値を計算する getMean を定義し、getMean(array2) と getMean(array3) で平均値を計算する、などが一つのやり方です[2]。getMean を作成してみようという人は「ヒント」の ArrayList に対するループの書き方を参考にしてください。

- データファイルの中から解析や処理に使う部分だけを配列や ArrayList に保持しておく、というのはデータ解析などでは典型的な処理です。ArrayList を引数として平均値、分散、グラフ画像、などを計算するメソッドを作成し適用するというのがよくあるやり方です。

- p.23 課題 1.6 平均値をとるのは 2 列目と 3 列目で良いです。

- p.23 課題 1.6 「table.csv」の一番最後に空行が入っています。何も対処しないとエラーになってしまうので空行に対応するようにプログラムを作成してください。

- p.25 「cp」の行で「〜の後に宛先のファイルオペランドがありません」というエラーがでる人は cp コマンドの最後にスペース+「.(ピリオド)」が抜けています。UNIX では「.」はいまいる場所をさすので、file_a.dat をいまいる場所にコピーしたければ「cp file_a.dat .」となります。印刷の具合で見づらくてスミマセン。

- p.26 パッケージを Eclipse で作成する方法: この TestPackage.java をそのまま Eclipse のプロジェクトの src に import して実行しようとするとエラーになります。

  - もっとも簡単な方法は Eclipse にインポートした段階で、「org.mypackage.test」のところに赤線が出ると思いますが、その行の左端の赤いバツのアイコンをクリックして出てくるメニューから「'TestPackage.java' をパッケージ'org.mypackage.test' に移動」を選ぶと、このパッケージが作成され (対応する workspace の src ディレクトリに org/mypackage/test が作成され)、TestPackage.java がそこに移動され、無事コンパイルできます。
  - 以上を明示的にやる場合、パッケージエクスプローラーでプロジェクト名をクリックして選択し、ファイル → 新規 → パッケージを選び、新規 Java パッケージの名前欄に「org.mytest.package」を入力し完了します。すると、パッケージエクスプローラーの src の下に org.mypackage.test ができるので、TestPackage.java をドラッグアンドドロップで移動します。これでコンパイル、実行ができるようになります。

[2] ArrayList の ArrayList として table.dat 全部を保存するとか、各列を ArrayList につっこんで、get メソッドで 2 番目と 3 番目を取り出すとか、他にもいろいろ使い方のパターンはあると思いますけど。

- p.29 課題 1.7 charts4j のサンプルですが、以前まではトップページのしたのギャラリー (グラフの絵の例がならんでいるところ) の「Bar Chart Example 2」などをクリックするとコードが見えたのですが、見えなくなっているようです。(次に続く)

- p.24 commons-lang3-3.4-bin.tar.gz を tar で解凍しようとしたとき「`gzip: stdin: not in gzip format`」などと言われる場合、おそらく右クリックでダウンロードしたと思います。Binaries のところの当該ファイルのところを「左クリック」してください。しばらくするとどこに保存するか聞かれるので適宜指定してください。

- p.29 課題 1.7 charts4j の JAR ファイルがどこにあるか分からない。
charts4j-1.3.zip をダウンロードして、解凍すると、まず「home」と「charts4j-1.3」の 2 つのフォルダが見えると思います。ここで charts4j-1.3 → lib の中に「charts4j-1.3.jar」があります。このファイルをパスの設定の際に外部 JAR ファイルとして指定してください。

- p.29 課題 1.7 charts4j のサンプルは、上の「DOWNLOAD LATEST STABLE VERSION」などが並んでいる行の最後の「Github」をクリック → 「src」をクリック → 「test/java/com/googlecode/charts4j」をクリック → 「example」をクリック、でみることができます。トップページの「Bar Chart Example 2」を出力するためのソースコードは「BarChartExample.java」の中の「publci void example2()」に対応します (たぶん)。

- p.29 課題 1.7 charts4j のサンプルは、そのままダウンロードしてコンパイルしようとすると失敗すると思います。たとえば「BarChartExample.java」をダウンロードして import してもそのまま実行しようとすると赤線がたくさん表示されてできないと思います。参考にするときは、作成してみたいサンプルの java ソースの該当するメソッド (たとえば example1() など) の中身を見て、「// EXAMPLE CODE START」から String url を生成する「// EXAMPLE CODE END」のところまで (最後の assert や Logger は実行には関係ありません) を、自分で書いた main メソッド (p.29 の LineChartExample.java の main など) にはりつけて実行してください。最後に生成された String url の文字列を「System.out.println(url);」で出力すると生成された画像の URL が出力されます。
なお、この example に置かれている、.java ファイルは「テスト」用のコードで、もしちゃんとライブラリが動いていたら正しい出力はこうです、ということを確認する (assert) もので、JUnit という Java のテストを行うライブラリ用のコードです[3]。

- p.37 課題 1.10 「課題 1.9 の main メソッドを修正し」とありますが、main メソッド以外にも修正を加えて構いません。

- p.43 「黙的スーパー・コンストラクター ○○○ は未定義です。 別のコンストラクターを明示的に呼び出す必要があります。」
欄外注 8 にもありますが、いま、コンストラクタとして String をとるものを定義しているため、引数がなにもないコンストラクタは作られなくなります。こういうクラス X を継承したクラス Y を考えたときクラス Y のインスタンスを作成するとき、親 X をまず引数なしのコンストラクタでインスタンス化する仕様になっています。なので、X に引数なしのコンストラクタが定義されてない場合、このようなエラーが出ます。空メソッド (処理は何もしないメソッド) で良いので定義してください。

- p.46 課題 1.13 JankenPlayerTypeA も、JankenPlayerTypeB も、RandomJankenPlayer を extends して作られているため、p.45 のソースコード 1.3 の

```
RandomJankenPlayer player2 = new JankenPlayerTypeB("Yamamoto");
```

にあるとおり、左辺の RandomJankenPlayer 変数 player2 で受け取れます。このインスタンス player2 は実体は JankenPlayerTypeB なので、RandomJankenPlayer のメソッドに加えて、新たに加えたメソッド (もしあれば) が本来プラスアルファで使えます。しかし、ここでは「RandomJankenPlayer」の変数として受け取っているので、player2 では RandomJankenPlayer のメソッドしか使えません (プラスアルファが使えません)。しかし、ここでは対戦させるだけなので、RandomJankenPlayer のメソッド showHand しか使わないため、これで十分です。しかも実体としては showHand

[3]興味ある人は JUnit とかユニットテストでググってください。

は上書きされているため、挙動は実はランダムではありません。つまり、同じメソッド showHand を呼び出しているだけだが、異なる挙動を与えることができています。これが「多態性」で継承を用いるメリットのひとつです。

- p.46 課題 1.13 この RandomJankenPlayer はもちろん、勝敗を保持するようにいろいろと書き換えたもので構いません。
- p.56 課題 1.17 Judge2 で使っている JankenPlayer は notify や report のメソッドを持っています。このソースは参考情報なのでまず分析してください。課題の内容が検証できるプログラムであれば、この Judge2 をそのまま実行できるように書き換える必要はありません。もし Judge2 を活用して行う場合は適宜 JankenPlayer に notify や report を実装してみてください。