プログラミングの基本要素

2002 年 10 月 10 日　増原 英彦

1　プログラムの基本概念

1.1　プログラミングと Java (教科書 1.1)

1.2　仮想マシンとバイトコード (教科書 1.2)

1.3　プログラムの基本要素 (教科書 2.1–2.3)

- コンストラクタ式: `new` クラス名 `()`

  クラス名クラスのオブジェクトを 1 つ作る。例 `new TurtleFrame()`

- 変数の型宣言: 型名 変数名;

  型名型の値をしまう変数を 1 つ用意する。クラス名は型名の 1 つ。例 `TurtleFrame f;`

- 代入: 変数名 `=` 式;

  式の値を変数名の変数にしまう。例 `f = new TurtleFrame();`

- 初期値付き変数の型宣言: 型名 変数名 `=` 式;

  型名型の値をしまう変数を 1 つ用意し、その中に式の値をしまう。例 `Turtle m = new Turtle();`

- (インスタンス) メソッドの呼び出し文: オブジェクト式 `.` メソッド名`(`式 1`,`式 2`,...);`

  オブジェクト式が表わすオブジェクトに、メソッド名という処理 (メソッド) を実行することを指示する。その際、メソッドには 式 1, 式 2,... が表わす値 (引数あるいはパラメータと呼ばれる) が渡され、メソッドはその値を使って処理内容を変化させることができる。例 `f.add(m); m.fd(100);`

2　プログラムのコンパイルと実行 (教科書 1.3, 2.4)

ここでは以下のことを説明する。なお説明は教育用計算機システムの NC 環境を仮定する。教育用計算機システムの Unix 環境や、いわゆるパソコン上では、その環境に応じたコマンド等を使う必要がある。

- 教科書で用いるライブラリプログラムをホームディレクトリにコピーする
- コピーしたプログラムをコンパイルし実行する
- コピーしたプログラムをもとに別のプログラム編集、コンパイル、実行する

2.1　ライブラリプログラムのコピー (準備)

教科書では、タートルグラフィクスを用いたプログラムを主に説明のために用いている。このタートルグラフィクスには、絵の表示など頻雑な処理が多く含まれている。こういった処理をするプログラムをゼロから作成することを避けるため、基本的なプログラムをコピーした上で演習を行う。

手順 1 *(ライブラリプログラムのコピー)*

NC 環境のターミナルウィンドウ *(kterm)* に「***/nchome/masuhara/cp1*** Return」と入力せよ。自分のホームディレクトリの下の ***javabook*** というディレクトリに、ライブラリプログラムがコピーされる。

注意*:* この操作は ***javabook*** の下にあるファイルを**上書き**する。

手順 2 *(手順 1 の確認)*

手順 *1* に続けて、ターミナルウィンドウ *(kterm)* に「***cd ; ls javabook*** Return」と入力せよ。以下のように表示されればライブラリプログラムがコピーされていることが分かる。

```
masuhara@as304> cd ; ls javabook
README    event     gui       javabook  turtle
applet    graphics  io        net
masuhara@as304>
```

2.2 ソースプログラムのコンパイル

プログラムのコンパイルはjavac というコマンドを用いる。コンパイルしたいソースプログラムは、javac コマンドの引数として指定する。ここでは、ライブラリプログラムに含まれる T21.java というソースプログラムをコンパイルする手順を説明する。

手順 3 *(コンパイル) NC* 環境の *kterm* で以下の操作をせよ。ただし、 *()* で囲まれた操作は確認のためのものであり、コンパイルのために必須のものではない。

1. cd ~/javabook/turtle Return と入力し、ライブラリプログラムがコピーされたディレクトリへ移動する。*(*何もメッセージ表示されなければ成功*)*
2. *(****pwd*** Return と入力し、現在のディレクトリを表示する。***/nchome/g123456/javabook/turtle*** と表示されれば成功*)*
3. *(****cat T21.java*** Return と入力し、プログラムの内容を表示する*)*
4. ***javac T21.java*** Return と入力し、コンパイルする。*(*何もメッセージ表示されなければ成功*)*
5. *(****ls -l T21.class*** Return と入力し、コンパイルされたクラスファイルが作られていることを確認する*)*

2.3 オブジェクトコードの実行

上でコンパイルしたプログラムのオブジェクトコード (クラスファイル) を、Java 仮想機械を使って実行する。Java 仮想機械を動かすには java というコマンドを用いる (「c」がないことに注意。) 実行したいオブジェクトコードは、java コマンドの引数として**クラス名**(クラスファイルの名前でないことに注意!) を書くことで指定する。

手順 4 *(実行) NC* 環境の *kterm* で ***java T21*** Return と入力する。数十秒持つと画面上にウィンドウが開き、表示が変化する。

多くのプログラムは処理が完了すると勝手に終了する。しかし教科書で用いている例題プログラムは、一連の表示をした後も指示を待ち続けるように作られているので、終了を指示する必要がある。

手順 5 *(終了)* 次の方法のどれかによって手順 *4* で実行したプログラムを終了させる*:*

- 開いたウィンドウの「*File*」メニューの「*Quit*」を選ぶ
- 開いたウィンドウの「閉じる」ボタンをクリックする
- ***java*** コマンドを実行したターミナルウィンドウ *(kterm)* で、「Control c」とタイプする

3　プログラムの編集

ライブラリプログラムをもとに、新たなプログラムを作ってみる。ここでは T21.java をもとに T20.java を作ってみる。

手順 6 *(プログラムの編集)*

1. ターミナルウィンドウ *(kterm)* に「*cp T21.java T20.java* Return」と入力する。*(*何もメッセージ表示されなければ成功*)*
2. エディタ *(emacs)* で *T20.java* を開く。
3. ファイルの中身を変更する。
   - 区別のため、最初の行のコメントの内容を変えておく
   - クラス名を *T21* から *T20* に変える。
   - *main* メソッドの内容を適当に変える。例えば「*m1.rt(90);*」と「*m1.fd(90);*」と書かれた *2* 行を、その直後にコピーしてみる。
   - 変更したファイルを保存する。
4. 手順 *3* と同様にして、編集したファイルををコンパイルする。ファイル名を *T20.java* に変えたことに注意。
5. 手順 *4* と同様にして実行する。クラス名が *T20* に変わっていることに注意。

4　練習

練習 1-1: (星)　T21.java をもとにして、星の絵を描くT20.java を作り、コンパイル、実行せよ。

練習 1-2: (逐次実行)　T21.java では 1 匹目の亀が 2 回動いた後で 2 匹目が 1 回動く。これを変更して 2 匹の亀が交互に動く T20Kougo.java を作れ。

練習 1-3: (変数と型)　変数の型宣言は、その変数にしまうことのできる値の集合を表わしている。T21.java の「TurtleFrame f;」を「Turtle f;」に変えた場合に何が起きるかを想像してみよ。実際にそのように変えた T20Kata.java を作りコンパイルし、そのときに起きた現象を説明せよ。

練習 1-4: (変数)　T21.java の中では、亀の回転角は「m.rt(90);」における「90」のように直接、数値式を書いていた。これを angle という変数を宣言し、「m.rt(angle);」のように変数を使って 2 匹の亀の回転を指示するプログラムT20Variable.java を作れ。ただし、整数値の型の名前は int である。

練習 1-5: (オブジェクトの区別)　上の練習からは、数値式 (例えば 100) を、それと同じ値を持つ変数式 (例えば distance) に置き換えてもプログラムは同じ動きをすることが分かった。

逆に、オブジェクトを値として持つ変数式をオブジェクト生成式で置き換えてもプログラムは同じように動くだろうか?

例えば T21.java では「m」という変数には「Turtle m = new Turtle();」という文の「new Turtle()」というオブジェクト生成式で作られたオブジェクトがしまわれている。そこで「f.add(m);」や「m.fd(100);」のようなメソッド呼び出し文における「m」を「new Turtle()」に書き換えた T20Kubetsu.java を作り、コンパイル・実行してみよ。もとの T21.java と同様に動かない場合は、その理由を説明せよ。