電子情報工学実験実習報告書
題目：Javaオブジェクト指向プログラミング

2402　飯田宗玄

平成14年 11月 17日

1　目的

- Java によるプログラミングを学習する．
- Java によるプログラミングを通し，オブジェクト指向の初歩を理解する．
- アップレット製作のための知識や技術面での準備をする．

2　原理

2.1　オブジェクト指向とは

「オブジェクト指向」とはプログラミングをする上での考え方のひとつで，これに沿ってプログラミングを行うと，より現実の世界に沿った構造のプログラミングができると言われている．また，オブジェクト指向は巨大なプログラムを作る場合や，複数の人が関わる場合に力を発揮する．

オブジェクト指向での「オブジェクト」とはある「もの」に関する情報を入れておく「変数」と，その情報に対して操作を加えたり，その情報をもとに仕事をしたりする「メソッド」のかたまりのことである．

Java では「クラス」として，まず，作成したいオブジェクトの中に入れる関数とメソッドを定義し，これをにをもとに「インスタンス」をつくり，インスタンスを使って作業をすることになる．Java での「オブジェクト」にあたるものは，このインスタンスである．クラスはインスタンスの設計図のようなものであるから，１つのクラスに対して，たくさんのインスタンスを作る事もできる．

ただし，クラスにはインスタンスを作らない別の使い方もある．例えば,「継承」がある．作りたい機能の一部だけを持っているクラスに，機能を追加したり変更を加えたりして，新たなクラスを作ることができる．継承は GUI を持ったプログラムを作る際などに使用される．また,「スタティック（static）」なメソッドを含むクラスはインスタンスを作らなくても使用できる．

クラスは機能や使い方などの「外見」さえ知っていれば使うことが出来るので，プログラマはクラスの中のコードがどうなってるか等「中身」は知る必要がない．これは，巨大なプログラムや複雑なプログラムを書く場合に，最後のほうになって最初に書いた部分がどうなっているか忘れた，などという事があっても，クラスの外見さえ分れば内容まで読む必要がないので，非常に便利である．同様に，複数のプログラマがいる場合も，他人が作ったクラスを「外見」を知るだけで簡単に使うことが出来るので，便利である．

2.2 Javaとは

Javaは，C++などと同様に（または，それ以上に）オブジェクト指向のプログラミング言語で，Sun Microsystems社によって，元々は主に家電などで製品同士の連携をとるために開発された．しかし，その後インターネットが広がるに連れて，異なるプラットフォームでもほぼ同じ動作をするということで注目されるようになった．現在では，一部の携帯電話上でもアップレットが動く．

Javaのもう１つの特徴はインタープリタ型言語とコンパイラ型言語両方の性質を持つという点である．これは，はじめに書いたプラットフォームに関係無く動作する事と関係がある．プログラマが書いたソースを単独で実行できるようにしてしまうと，どうしても実行できる環境が決まってしまう．しかし，完全にインタープリタ型とすると，プラットフォームを気にしなくてよくなる代わりに実行速度が犠牲になる．そこで，Javaでは，ソースをまず「中間コード」という状態にコンパイルし，これを各プラットフォームに対応した「仮想マシン (Java VM, Java Virtual Machine)」の上でインタープリタ的に実行している．こうする事で，ある程度の実行速度と，プラットフォームに依存しないという性質とを共存させている．また,「JIT（Just-In-Time)」よばれる，はじめに中間コード全体をプラットフォームに対応した状態（ネイティブ・コード）に完全にコンパイルしてから実行し，実行速度を上げるしくみもある．

3 方法

3.1 使用した環境

Microsoft Windows 2000 Professional，JDK1.3.1_02

3.2 コンパイルと実行の方法

※今回の実習では,「C: jdk1.3.1_02」というディレクトリ構造になっていた．

1. 「スタートメニュー」から「コマンドプロンプト」を起動．
2. まず,「PATH」という環境変数を設定する．(以下の様に入力)

```
PATH %PATH%;c: jdk1.3.1_02 bin [Enter]
```

3. コンパイルしたいソースのあるディレクトリで，

```
javac（クラス名).java [Enter]
```

と入力すると，コンパイルされ「（クラス名）.class」という名前の中間コードが出来る．

4. この中間コードをインタープリタで実行する．

```
java（クラス名）[Enter]
```

4　結果

配布プリントに掲載されているプログラムのうちの５つについて，ソースと実行結果をプリントアウトしたものを添付する．

5　考察と結論

5.1　添付した各プログラムについて

プログラムの各部分がどのような役割を果たすのかなどを検討する．なお，行頭の数字はソースの行数と対応している．

5.1.1　課題 2.2.5

1: 「manyTimesHello」という名前の「public」なクラスを定義している．

2: 「main」という「public」で「static」な，返り値を持たないメソッドを定義している．最初にこのメソッドが実行される．

3: 整数型の変数 T を定義し 0 を代入．

5-8: T が 4 以下である限り繰り返し，毎回，T の値を 1 増やし「HELLO...（回数）回目」というような表示を行う．

5.1.2　課題 2.2.6

1-3: 上記とほぼ同様．

5: 変数 n の値を毎回 1 増やしていき，100 以下である限り繰り返す．

6: 変数 i の値を毎回 1 増やしていき，n の 1/2 以下である限り繰り返す．

7: もし，n が i で割り切れたなら外側の for 文に戻る．

8: 出力した数字が１桁の場合は空白を出力して桁数を合わせる．

5.1.3 課題 2.3.3

1-5: 上記とほぼ同様.

7: 変数 a の値を 0 から毎回 1 増やしていき，５回限り繰り返す.

8: 乱数を発生させ，それを引数として「printFact」メソッドを呼び出す.

12: 「printFact」メソッドを定義している.

13-17: T を 1 づつ増やしながら引数で指定された回数くりかえし，1 に毎回そのときの T を掛けていき階乗を計算し，計算したものを返す.

5.1.4 課題 2.5.3

1: このように書いておくと「System.io」のメンバはそれ自体の名前を直接書く事が得きる.

3: ほぼ上記と同じだが,「IOException」が起きた場合は処理を自分でやらず，任せてある.

6-7: 「Buffered Reader」クラスのインスタンス「KB」を,「System.in」で初期化した「InputStreamReader」クラスのインスタンスで初期化する.

9-11: KB を使って入力を取得.

13-26: 変数 k を 1 づつ増やして，毎回，その数で因数分解したい数が割り切れなくなるまで割り, 割り切れなくなったら k そのときの k と割った回数 T を表示．これを，因数分解する数と k が等しくなるまで繰り返す.

5.1.5 Parabola.java

10: Motion メソッドを a と x を引数にして呼び出す.

12: x の配列の長さと同じ回数だけ繰り返す.

13-20: 行頭に 2 桁の幅で，そのときの値を表示．ただし，10 以上は 5 きざみ.

22-29: Motion から返された値より 1 少なく空白を表示し，最後に「*」を表示して改行.

32: Motion メソッドを定義.

33-36: 「時間」に相当する変数 t を 1 ずつ増やしながら，毎回，t を使いその時間でのグラフの値（具体的にはスペースの数+1）をもとめる.

5.2 結論

今回改めてJavaを学習し，今までより，Javaとオブジェクト指向についてより理解が深まった.