構造化プログラミングからオブジェクト指向プログラミングへ

（Java が動かない PC の場合は JavaSDK1.3 以上をインストールして進めて下さい。）

１．構造化プログラミングとは（順次　選択　繰り返し　の３大要素でプログラムを組む）

プログラム起動開始から順次処理を行い、共通する処理部分をくくり出して別の処理にまとめ、条件で分岐し、範囲を決めて繰り返す処理をいう。

構造化の簡単な例

以下のようなプログラムを構造化すると下記のようになる。

構造化前

```
public class Non_struct1{
        public static void main(String[] args){
                System.out.println(“処理 1”);
                System.out.println(“処理 2”);
                System.out.println(“処理 3”);
                System.out.println(“処理 1”);
                System.out.println(“処理 2”);
                System.out.println(“処理 3”);
        }
}
```

構造化後 その１：　共通部分（繰り返し）をくくり出す

```
public class Struct1{
        static void printout( ){
                System.out.println(“処理 1”);
                System.out.println(“処理 2”);
                System.out.println(“処理 3”);
        }
        public static void main(String[] args){   for( int i = 0 ; i < 2 ; i++){ printout( );}   }
}
```

構造化後 その２：　さらに共通部分（繰り返し）をくくり出す

```
public class Struct2{
        public static void main(String[] args){
                for( int i = 0 ; i < 2 ; i++){ printout( );}
        }
        static void printout( ){
                for( int i = 1 ; i <= 3 ; i++){ System.out.println("処理"+i);}
        }
}
```

構造化すると for ループや別クラスを準備するなど、オーバーヘッドと呼ばれる処理が必要になり、プログラム行数が増えます。しかし、例えプログラムの行数が増えたとしても、「処理の流れを見やすくしよう」というのが、構造化プログラミングです。

「処理を流れで考える」ことから、構造化プログラミングの設計は「流れ図」（フローチャート）で表記されます。

また上記の Struct2 は次のような階層型の構造になっており、このことから構造化と呼ばれています。

public class Struct2
　└public static void main(String[] args)
　　└static void printout ( )

単純な構造
　└単純構造
　　└単純構造

「各階層の機能が正しく動作していれば、全体も正しく機能する」と考えられますので、これが構造化プログラミングの核となる考え方です。

２．オブジェクト指向プログラミングとは（隠蔽化　継承　多態性　の３大要素でプログラムを組む）

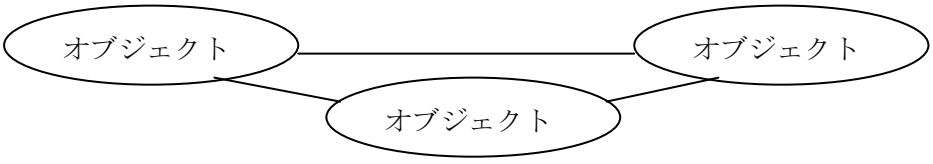
上記の構造化プログラムも分かりやすくなっていますが、これだけでは他のプログラムから利用したりするのが困難です。そこで再利用性を考慮して次のようにします。

オブジェクト指向後 その３ ：上記の発展形

```
public class Struct3{
        public static void main(String[] args){
                Print_Out    display;
                for( int i = 0 ; i < 2 ; i++){ display.printout( );}
        }
}
public class Print_Out{
        static void printout( ){
                for( int i = 1 ; i <= 3 ; i++){ System.out.println("処理"+i);}
        }
}
```

ここでは Print_Out という、どこプログラムからでも利用可能な設計図（クラス）を準備し、これに printout( )という機能を持たせています。そして実際に使うときは display という実体（インスタンス）を作り、インスタンスを通じて display.printout( )という機能（画面表示）を実現しています。

ポイントとなるのは、オブジェクト指向プログラミングでは、処理を流れで考えるのではなく、もの（オブジェクト）の連携で考えることです。



３．従来の構造化プログラミングとオブジェクト指向プログラミングの違い

構造化プログラミング：(トップダウン型)

全体の流れを考えてから、個々の機能を詳細化してゆく方法

オブジェクト指向プログラミング：(ボトムアップ型)

必要な部品を先に考える方法

オブジェクト指向型開発の場合は全体からではなく、個々の部品（もの：オブジェクト）から作り、そのオブジェクトを組み合わせて全体を構成して行きます。構造化プログラミングでは、まず全体が把握出来ないと設計も、プログラミングも出来ません。しかしオブジェクト指向プログラミングでは、全体を考える前に、部品化できるもの（オブジェクト）を探すことから始まります。

４．オブジェクト指向の考え方

例として「おみくじ」システムを作りましょう。通常、おみくじの機械にはお金を入れるとおみくじが出てきます。ここでは単純に画面上のボタンを押すと、お金を投入したと考えましょう。

構造化プログラミングで考えると次のようになります。

１）　機械の電源を入れ、プログラムを起動する
２）　メニュー画面を表示する
３）　１が入力されたら、乱数を元に「おみくじ」の結果を表示する
４）　２が入力されたら、メニュー画面にもどる
５）　２）へ
６）　もし９が入力されたら、プログラムを終了する

オブジェクト指向プログラミング：(ボトムアップ型)

１）　機械の電源を入れ、プログラムを起動する
２）　ボタンが押されたら、乱数を元に「おみくじ」の結果を表示する
３）　終了ボタンが押されたら、プログラムを終了する

いかがでしょう、処理の流れで考えず、私たちの世界と同様に「もの」を操作することで目的を達成しています。通常、１人でプログラムを組んでいれば意図しない処理は起こりません。しかし、複数名のプログラマが短期間でプログラムを組むと、ある人は勘違いをして、ボタンではなく、お金を入れた時点で、おみくじの結果を画面に出してしまう処理を書いてしまうかも知れません。

おみくじの結果算出にはボタンを押した後に、特別な処理をして内部情報を算出する予定だったにもかかわらず、先の処理は、いつも同じパラメータで乱数を呼び出すため、毎回「吉」という値を内部情報に書き込んで、「吉」しか画面表されないシステムになってしまう恐れもあります。

このように、設計者の意図しない方法で、おみくじ内部の情報が変えられてしまっては、元も子もありません。従って、オブジェクト指向では、実際のおみくじ機械（オブジェクト）と同じように、いきなり内部のおみくじを取り替えることが出来ないようにしておく必要があります。これを隠蔽化といいます。

演習１　おみくじ表示システムを構造化プログラミングで作ってみて下さい。

まずは簡単化のため、全て main メソッド内で処理を記述して下さい。

プログラム名：　Omikuji1.java

以下のプログラムは、実行時の時間で乱数を発生させ、おみくじを表示するプログラムです。

```
import java.math.*;
class Omikuji1 {
        public static void main(String args[]) {
                double randomNumber = Math.random();
                if(randomNumber < 0.2) { System.out.println("凶です");}
                else if(randomNumber < 0.4) { System.out.println("小吉です");}
                else if(randomNumber < 0.8) { System.out.println("吉です");}
                else { System.out.println("大吉です");}
        }
}
```

実行結果例：

＞ java Omikuji1

中吉です。

課題７　さらにおみくじプログラムにお金を入れる機能を追加しましょう。金額はコマンドプロンプトでプログラム実行時の引数として入力して下さい。

要件：　入力金額が 100 円に満たなかったら、＊＊円不足です。

もう一度、はじめからやり直して下さい、と表示して下さい。

実行結果例：

＞ java Omikuji1　120

おつりは 20 円です。

中吉です。

＞ java omikuji1　50

50 円不足です。

もう一度、はじめからやり直して下さい。

課題８　おみくじプログラムをオブジェクト指向で作りましょう。今度は[Man(人間)]と[Omikuji(おみくじ機械)]を別クラスにして部品化して作ってみて下さい。

まずは簡単化のため、プログラム起動時に自動的にボタンが押されて結果が表示され、プログラムが終了するものと考えて下さい。

メインのプログラム名：　Man.java おみくじ機械のプログラム名：　Omikuji.java

実行結果例：

＞ java Man

大吉です。