プログラミング言語IIIA(Java) テーマ02

内容

Javaはオブジェクト指向プログラミング言語であるため, オブジェクト指向に特有な, 難解で崇高な専門用語(英語/カタカナ)が大量に登場し, C言語のような手続き型プログラミング言語に慣れた初学者に, 非常に高いハードルを感じさせる.

しかし, それらの専門用語は, C言語の世界の言葉に置き換えて考えれば何も恐れる必要はない. 基本は変数・関数・メモリ・ポインタ・構造体である. 今回も前回に引き続き, 諸君が昨年度学習したC言語と, 初めて見たJavaのプログラムを比較し, Javaやオブジェクト指向とはどういうものかを探っていく.

課題

以下の課題のレポートは, レポートファイルreport02.txtを作成してアップロードにより提出すること. ただし, 課題1のみ紙での提出となるので注意のこと. A4用紙に手描きして, 情報棟3階天元教員室ポストへ提出すること.

1. 今回の預金口座プログラムについて, main関数におけるメモリの様子を図に描き表せ. 変数を箱で表し, さらにポインタを, 箱と箱をつなぐ矢印で表現せよ. なお, アドレスの値は printf("%p¥n", my_koza) で調べることができる(ただし実行する度に変化する). (紙に手描きして提出)

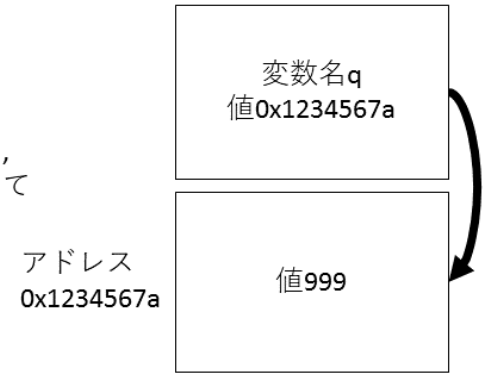
   準備: A4用紙の向きは縦で, まず上部に「4J 出席番号 名前 テーマ02 課題1レポート」と書く.

   ヒント: 「ポインタ(pointer)」とは, 「ポイントする(point) + もの(er)」, すなわち, 他の変数を指し示す変数の意である.

```
// qをintへのポインターとする.
// つまり, qが指す先はint型.
int *q;

// int型一つ分の大きさのメモリーを割り当て,
// そのアドレスをintへのポインターに変換して
// qに代入する.
q = (int *)malloc(sizeof(int));

// qが指す先に整数を一つ保存できる.
*q = 999;
```



   ヒント: 「構造体」とは, 複数の変数を, 連続したメモリにひとまとめに配置したものである. 絵的には, 一つの大きな箱の中に, 複数の小箱が入っている様子をイメージすると良い.

```
struct Koza      /* 預金口座構造体 */
{
    double riritsu; /* 利率 */
    int gankin;     /* 元金 */
};
```

   注意: 今回の図に関数は不要. kotae等のローカル変数も不要. 登場するのはriritsu, gankin, my_kozaの三つのみ. 矢印も一つのみ.

レポート自己チェックリスト

- ☐A4用紙を縦向きに使った.
- ☐上部に名前やテーマ番号等を記入した.
- ☐変数は riritsu, gankin, my_kozaの三つのみ書いた. 他の変数や関数は書いていない.
- ☐矢印は一つのみ描いた. 二つ以上描いていない.

知識: Java版のmain関数のmy_kozaには「*」(アスタリスク)が付いていないが, 実は, C言語版と同じポインタである. Javaではこれを「参照(リファレンス)」と呼ぶ.

2. 今回のJavaプログラムを, 元金をコマンドライン引数から入力できるように改造せよ. (ソース Koza2.java を作成し, レポートファイルに貼付け, コンパイル方法・実行方法と実行結果も貼付け)

レポート自己チェックリスト

- ☐ソースファイル名をKoza2.javaにした.
- ☐ソースコードを記載した.
- ☐コンパイル方法・実行方法を記載した.
- ☐実行結果を記載した.

```
ソースファイルのコピー方法の例
$ cp Koza.java Koza2.java
```

- ファイル名を変更したので, ソース中に書いてあるクラス名も「Koza2」に変更する. 修正漏れに注意のこと.

```
実行方法の例 ↓これがコマンドライン引数
$ java Koza2 20000
```

- コマンドライン引数は配列argsに, 0番から順に格納されている. つまり, 実行時にコマンドライン引数から与えた元金は, args[0]に格納されている. ただし, 文字列型であり, 整数型ではない.
- 文字列を整数に変換するには, 「Integer.parseInt()」という関数を使う. この関数の引数に文字列を一つ与えると, 整数型に変換されて返ってくる. それを元金の変数に代入すると良い.
- 知識: C言語では文字列型という型は存在せず, ¥0で終わる, 文字の配列であった. これに対し, Javaでは, 予め文字列型であるStringクラスが用意されており, 文字列を, 配列ではない, 一つの変数として扱える. 複雑だが, Javaのmain関数のargsは, この文字列型の, さらに配列である. → String型はテーマ08で, 配列はテーマ10で正式に扱う予定.

```
現時点では「JavaのString型」==「C言語のchar[]型」と考えて良い. どちらも「文字列型」.
```

3. C言語版のKoza_risoku関数の引数にある「struct Koza *koza」や, kekkaの計算式中の「koza->」が, Java版では不要で, とてもシンプルな記述となっている. その理由を推測せよ. (レポートファイルに記入)

レポート自己チェックリスト

- ☐理由を記載した.

ヒント: Java版のプログラムの中で, risoku関数がある場所と, その周り(外側)の位置関係を眺めてみると良い.

ヒント: 「class Koza」の閉じ中括弧「}」の場所を調べてみよう. 「struct Koza」の閉じ中括弧「}」の場所との違いに注目しよう.

4. 課題3より, Javaにおける「クラス(class)」とはプログラムコード的にどの様なものか, 自分の考えをまとめよ. 論理的に考えること(「～が～なので, クラスとは～ものと考えられる.」の形式で答える.) (レポートファイルに記入)

レポート自己チェックリスト

- ☐自分の考えを記載した.
- ☐「～が～なので, クラスとは～ものと考えられる.」の形式で書いた.

注意: 現時点でどの様に推測するか, 自分でどう考えるか, が大切なので, 教科書やネットを調べて正解(正確な定義)を答えるのが目的ではない.

ヒント: 構造体のメンバ変数がある場所, 関数がある場所, クラスの中括弧{}の場所, それらの位置関係

5. Java版の預金口座プログラムを, さらに自由に改造してみよ. 特に何も思い浮かばなければ, 6カ月に固定されている箇所を, ループで1カ月～12カ月に変化させてみよ. ファイル名・クラス名はKoza3とする. (ソース, コンパイル方法・実行方法, 実行結果, 改造した内容の説明をレポート)

レポート自己チェックリスト

- ☐ ソースファイル名をKoza3.javaにした.
- ☐ ソースコードを記載した.
- ☐ コンパイル方法・実行方法を記載した.
- ☐ 実行結果を記載した.
- ☐ 改造した内容の説明を記載した.

注意: ファイル名と同時に, ソース中に書いてあるクラス名も「Koza3」に変更する必要がある. 修正漏れに注意のこと.

アドバイス: for文やif文, while文などの基本的な文法はC言語と同じであるから, C言語で学習した様々な技術を試してみると良い. Javaの教科書も活用すると良い.

レポート

- 内容: 課題中に指示されている通り. 必要な項目を全て記載しているか, 十分に確認すること.