プログラミング演習２（第 15 回）　課題

２０２４年 １月 １９日

注意事項

- **今回はすべての問題で、授業後の提出はできません。**
- ソースファイルの先頭には必ず「学籍番号」「氏名」「**課題番号**」を下記の例のようにコメントとして入れること。main メソッドのあるクラス内に他のメソッドを作らないこと。

```
// 学籍番号:77H000, 氏名:産大太郎, 課題番号:課題1

class Kadai15_1
{
    public static void main(String[] args){
    {
        …
```

課題 1. (Kadai02_1.java)

実行例のように、2023 年 12 月のカレンダーを出力するプログラムを作成せよ。ただし、曜日、日付の表示の間には**水平タブ**をいれて、そろえて表示すること。なお、なるべく繰り返し文などを用いてプログラムを簡潔にすること。

[実行例]

```
                2023 年  12 月
日      月      火      水      木      金      土
                                        1       2
3       4       5       6       7       8       9
10      11      12      13      14      15      16
17      18      19      20      21      22      23
24      25      26      27      28      29      30
31
```

課題2. (Kadai15_2.java)

キーボードから整数値を入力して、実行例のように 8 倍 64 倍の値と、16 で割った答えを表示するプログラムを作成せよ。**ただし、数字を扱う変数は 1 つのみとする。また、シフト演算子と引き算のみを用い、それ以外の演算子を用いないこと。**

[実行例]　（下線部分はキーボードからの入力）

```
整数を 1 つ入力してください。2023
8 倍  16184  64 倍  129472
16 で割ると  126  あまり  7
```

```
整数を 1 つ入力してください。1008
8 倍  8064  64 倍  64512
16 で割ると  63  あまり  0
```

課題３．（Kadai15_3.java）

1 から 6 までの整数乱数を 100 個発生させ、実行例のようにその出現分布を棒グラフと値で出力するプログラムを作成せよ。

ただし、発生させた乱数を配列に格納するのではなく、**1 から 6 それぞれの出現回数を数えるための変数に配列を用いること**。（**出現回数をカウントするための変数を多数用意しないこと。**） 例えば、出現個数カウント用の配列 count[ ]を用意すると、 count[1]に 1 が出現した回数が格納される。もし、1 が 10 回出現した場合 count[1]=10 となる。また、**出現回数を数えるときに if や switch を使わないこと。**

[**実行例**] （乱数なので出力結果は毎回異なります）

```
3 2 3 2 5 3 6 5 2 5 6 5 4 4 1 1 1 2 5 6 1
3 1 2 1 3 5 3 3 3 5 6 1 2 6 2 2 4 1 6 6 4
4 1 4 2 4 2 5 5 2 3 6 3 5 6 6 2 5 1 4 4 2
2 2 3 6 5 2 4 1 4 3 2 5 3 5 2 4 4 5 1 2 3
2 3 1 2 4 4 3 5 2 5 1 6 2 5 1 3
1 *************** 15
2 ********************** 23
3 ***************** 17
4 *************** 15
5 ****************** 18
6 ************ 12
```

```
4 4 1 5 1 4 3 6 4 5 3 2 4 3 2 4 5 4 1 1 6
6 6 4 3 3 1 1 5 3 1 4 5 2 1 5 1 4 6 1 3 6
1 5 5 3 4 5 5 4 3 2 2 6 6 1 2 6 2 3 1 5 3
2 2 4 3 5 1 4 1 2 4 2 6 4 2 5 4 2 4 2 6 3
2 1 1 6 4 2 3 3 6 1 5 5 2 5 6 4
1 ****************** 18
2 ***************** 17
3 *************** 15
4 ******************** 20
5 *************** 16
6 ************* 14
```

課題4. (Kadai15_4.java)

実行例のように、**横幅**と**縦の長さ**を与えると長方形の**面積を計算**し、その値を表示するプログラムを作成せよ。プログラムは**以下の説明に従って作成する**こと。

① 以下のフィールドおよびメソッドを有する **Figure クラス**を作成。
   1. **width, height, square**： それぞれ横幅、縦の長さ、面積を保存するための実数型のフィールド
   2. **getArea**： 戻り値として、面積を実数型で返すメソッド。引数なし。
   3. **setLength**： **仮引数**として横 **w** と縦 **h** の値を受け取るメソッド。このメソッド内で**仮引数**の値をフィールドの **width** と **height** に代入し、その値を実行例のように表示する。戻り値なし。

② **main** メソッド内で **Figure クラスの変数 fig1 を用意**する。
③ **setLength** メソッドを使い、引数によって横、縦の長さをフィールドに代入する。
④ **getArea** メソッドを用いて面積を表示する。表示には **System.out.println** を使用する。

[**実行例**] （**青い字は引数の数値によって異なる。**）

```
図形の面積を求めます。                    <――  この行は main メソッドで表示
横 : 10.0  縦 : 40.0                      <――  この行は setLength メソッドで表示
面積は  400.0  です。                     <――  この行は main メソッドで表示
```

課題5. (Kadai15_5.java)

下記のサンプルプログラムは、収入と支出から貯金額を計算するプログラムの一部で、Kakeibo クラスは以下のようなフィールドおよびメソッドを有する。

Kakeibo クラス:

1. savings: 貯金の金額を格納する int 型のクラス変数とする。(ただしサンプルプログラムではクラス変数としてまだ設定されていない。)
2. income, expenses: それぞれ、収入および支出の金額を格納する int 型のフィールド
3. showData: 収入および支出の金額を表示するメソッド。引数無し。戻り値なし。
4. showSavings: 貯金の金額を表示する、引数なし、戻り値なしの、Kakeibo クラスのクラスメソッドとする。(ただしサンプルプログラムではクラス変数としてまだ設定されていない。)

このサンプルプログラムを編集して、実行例のように実行されるプログラムを作成せよ。用意されている Kakeibo クラスを、以下のように修正すること。

① savings を Kakeibo クラスのクラス変数となるように設定する。さらに、外部からアクセスできないように設定し、0 で初期化する。
② Kakeibo クラスの全てのフィールドを、外部からアクセスできないように設定する。
③ Kakeibo クラスの全てのメソッドを、外部からアクセスできるように明示的に設定する。
④ Kakeibo クラスに、全てのフィールドに 0 を代入する引数がないコンストラクタを作成し、外部からアクセスできないように設定する。
⑤ Kakeibo クラスに、以下のように実行する引数 2 個(inco, expe)のコンストラクタを作成し、外部からアクセスできるように明示的に設定する。
   1. 最初に引数の無いコンストラクタを呼び出す。
   2. 2 つの引数 inco, expe のうち少なくとも1つが 0 未満の場合はエラーを表示する。
   3. そうでなければ、inco および expe の値を income および expenses に、それぞれ代入する。
   4. あわせて、クラス変数 savings の値を更新する。Savings には収入を加算し、支出を減算する。
⑥ showSavings を Kakeibo クラスのクラスメソッドとなるように設定する。さらに外部からアクセスできるように明示的に設定する。

[実行例]

```
貯金 ： 0 円

収入 5000 円 支出 1500 円
貯金 ： 3500 円

***** エラー：入力した金額は無効です。 *****      ←この行は 0, -2000 で初期化
収入 0 円 支出 0 円
貯金 ： 3500 円

収入 0 円 支出 2000 円
貯金 ： 1500 円

***** エラー：入力した金額は無効です。 *****      ←この行は-3000, 500 で初期化
収入 0 円 支出 0 円
貯金 ： 1500 円

収入 3000 円 支出 500 円
貯金 ： 4000 円
```

```
class Kakeibo
{
        int savings;    // 貯金
        int income;     // 収入
        int expenses;   // 支出

        public void showData()
        {
                System.out.printf("収入 %7d 円 支出 %7d 円¥n", income, expenses);
        }

        showSavings()
        {
                System.out.printf("貯金  : %7d 円¥n", savings);
        }
}


class Kadai15_5
{
        public static void main(String args[])
        {
                Kakeibo kakeibo1;
                Kakeibo kakeibo2;
                Kakeibo kakeibo3;
                Kakeibo kakeibo4;
                Kakeibo kakeibo5;
        }
}
```

課題６．（Kadai15_6.java）

以下のようなプログラムを作成せよ。

1) 最初の文字列をキーボードから入力し String 型の変数 str1 に代入する。
2) 2 つ目の文字列をキーボードから入力し String 型の変数 str2 に代入する。
3) 文字列 str1 の中で**最後に現れる、文字列 str2 と同じ部分**を探し、その開始位置を変数 pos に代入。
4) **書き換え可能**な文字列(sb) を作成する。その際、**str1 の最初から pos の前までの文字列を小文字に変換した**文字列で初期化する。
5) sb に、**str2 を大文字に変換した文字列**を追加する。
6) sb を表示する

[**実行例**]（**赤字はキーボードから入力した文字、青字は変数の内容を表示した部分、**その他は文字列リテラルによる表示である）

```
文字列を入力
My name is Takeshi Takeyama.
文字列を入力
Tak

my name is takeshi TAK
```

課題 7. (Kadai15_7.java)

以下のような手順で実行例のように実行するプログラムを作成せよ。

① 以下のようなメンバを有する抽象クラス Person を用意する。
  A) 名前を収納する String 型のフィールド name。サブクラスからのみアクセス可能とする。
  B) フィールドの情報を表示するための抽象クラス show

② 以下のようなメンバを有する Student クラスを、Person クラスのサブクラスとして用意する。その際、フィールドおよび引数無しのコンストラクタを、サブクラスだけから直接アクセスできるようにすること。
  A) 学籍番号を収納する String 型のフィールド number
  B) number に“00H000”name に“No name”を代入する引数なしのコンストラクタ
  C) クラス内のすべてのフィールドを希望の値(文字列を含む)に初期化するための、外部からアクセス可能なコンストラクタ
  D) 実行例の2行目のように情報を表示するメソッド show()。(青文字部分は各フィールドの内容が反映されるようにすること)

③ 以下のようなメンバを有する、Student クラスをスーパークラスに持つサブクラス Score を用意する。
  A) 数学と英語の点数を収納するための integer 型の public フィールド math と english を作成する。
  B) Score クラスに、math と english に 0 を代入する引数なしのコンストラクタを public で用意する。
  C) スーパークラスのフィールドを含めたすべてのフィールドを希望の値(文字列を含む)に初期化するコンストラクタを用意する。その際、引数は学籍番号、名前、数学の点数、英語の点数の順とする。スーパークラスのフィールドの初期化にはスーパークラスのコンストラクタを利用すること。
  D) 実行例 1 行目のように情報を表示するメソッド show()。(青文字部分は各フィールドの内容が反映されるようにすること)

④ main 内で、抽象クラスの要素数 3 の配列変数 members を用意する。

⑤ 実行例を参考に、3 人分の学籍番号、名前、点数を、コンストラクタを利用して membersの各要素に代入する。(taro と hanako は Score クラス、ichiro は Student クラスのオブジェクトである。)

⑥ for ループを 1 つだけ用いて、members のすべての要素の内容を実行例のように表示する。その際、show( ) を利用すること。繰り返し回数は、配列の長さを読み取って設定すること。

[実行例] (青字は変数を表示した結果である。その他は文字列リテラルによる表示である)

```
学籍番号:77H777, 名前:Taro,  数学:70,  英語:80
学籍番号:77H778, 名前:Ichiro
学籍番号:77H779, 名前:Hanako, 数学:90, 英語:50
```