ＪａｖａプログラミングⅡ

１１回目　スレッド　課題

確認○×問題

次の各文は正しいか誤っているか答えなさい。

(1)スレッドは、1 つの実行箇所をもつ一連の処理の流れである
(2)マルチスレッドで各スレッドの処理は並行して実行される
(3)Java はマルチスレッド処理を記述できない
(4)新たにスレッドを生成する場合、Thread クラスを拡張し、かつ Runnable インタフェースを実装する必要がある
(5)新たなスレッドで実行する処理は Thread クラスまたは Runnable インタフェースの run()メソッドをオーバーライドして記述する
(6)複数のスレッドは、それを開始した順番に終了する
(7)同期とは、複数のスレッドの処理を互いに排他的に行うことである
(8)メソッドの修飾子に synchronized を付加するとそのメソッドの処理は排他的になる

■難易度★☆☆

課題 1 **2^n（n=1~30）を順次求めて一覧表示する処理を実行するスレッドを宣言しなさい。その後で、main()メソッドよりこのスレッドを起動しなさい。**

〔実行例〕

```
結果をお待ちください。          ←（main()メソッドはここで終了します）
2 の 1 乗は 2 です。            ←（以後、新しいスレッドによる出力です）
2 の 2 乗は 4 です。
2 の 3 乗は 8 です。
     :
2 の 30 乗は 1073741824 です。
```

〔2^n を順次求めて表示する処理を実行するスレッドの宣言〕

```
class TwoToPowerOf extends Thread{
    public void run(){
        2^n（n=1~30）を順次求めて一覧表示する処理を宣言します
    }
}
```

〔スレッドを起動する main()メソッド〕

```
class Assignments11_1{
    public static void main(String[] args){
        // 新しいスレッドを起動
        TwoToPowerOf ttp=new TwoToPowerOf();
        ttp.start();
        // メインスレッドはここで終わります
        System.out.println("結果をお待ちください。");
    }
}
```

■難易度★☆☆

課題 2 フィボナッチ数列の与えられた項の計算と表示を行うスレッドを宣言しなさい。Main()メソッドでキーボードから求めたいフィボナッチ数列の項を入力した後、このスレッドを起動してフィボナッチ数列の該当する項を求めなさい。以下に、フィボナッチ数列の与えられた項の計算と表示を行うスレッドの宣言、このスレッドを起動する main()メソッドのコードを示します。

〔フィボナッチ数列とは〕

1, 1, 2, 3, 5, 8, …
（漸化式）$n_1 = 1$、$n_2 = 1$、$n_k = n_{k-1} + n_{k-2}$ （$k \geqq 3$）

〔フィボナッチ数列の与えられた項の計算と表示を行うスレッドの宣言〕

```
class Fibonacci extends Thread{
    private int v1=1, v2=0, v3;
    private int n; // 求めたい項（1 以上）

    // 求めたい項をコンストラクタで設定します
    public Fibonacci(int i){
        n=i;
    }
    // フィボナッチ数列の与えられた項を求めます
    public int getTerm(){
        for(int i=0;i<n;i++){
            v3=v1+v2;
            v1=v2;
            v2=v3;
        }
        return v3;
    }
    // フィボナッチ数列の与えられた項を求め、表示します
    public void run(){
        フィボナッチ数列の該当する項を計算し、表示するコードを記述してください
                メンバの getTerm()をうまく用いましょう
    }
}
```

〔スレッドを起動するメインメソッド〕

```
class Assignment11_2{  // ★import java.io.*;は忘れずに一番上に書いてください★
    public static void main(String[] args) throws IOException{
        InputStreamReader isr=new InputStreamReader(System.in);
        BufferedReader br=new BufferedReader(isr);

        System.out.println("フィボナッチ数列の求めたい項を入力してください");
        String str=br.readLine();
        int n=Integer.parseInt(str);
        フィボナッチ数列の該当する項を計算して表示するスレッドを起動します
        System.out.println("結果をお待ちください");
    }
}
```

〔実行例〕

```
フィボナッチ数列の求めたい項を入力してください
30
結果をお待ちください                          ←（main()メソッドはここで終了します）
フィボナッチ数列 30 項目は 832040 です          ←（新しいスレッドによる出力です）
```

■難易度★★★

課題 3 銀行口座への操作には、預金や払い戻し、振込みなどがあります。一般に、複数の利用者が同一の口座へ並行してアクセスすることは容易に想定されます。例えば、ある預金者 C が本人の口座へ預金を行うのと並行して、他の利用者がその預金者 C の口座へ振り込みを行うなどです。このように並行して起こる処理はスレッドを用いると比較的容易に記述できます。次の内容をシミュレーションするコードをスレッドを用いて作成しなさい。

シミュレーションの内容：

1) 会社 A がある銀行に口座 a を持つ。
2) 会社 A の 3 人の社員が並行して次のように口座 a へ預金と払戻を行う。
   - 社員 1　100 万円預金して 75 万円払戻を行う
   - 社員 2　20 万円預金して 25 万円払戻を行う
   - 社員 3　50 万円預金して 20 万円払戻を行う
3) 銀行は各社員からの処理を整合性を取りながら実行する(synchronized)。

ヒント）教科書 p. 483 の Sample7「車会社と運転手」の例題を参考にしましょう
会社クラス　→　会社 A の銀行口座クラス
運転手クラス（スレッド）　→　社員クラス（スレッド）

■難易度★★☆

課題 4 次はキーボードから 2 つの整数を入力して加算を実行するスレッドです。

〔加算を実行するスレッドの宣言〕

```
import java.io.*;
class Addition extends Thread{
    public void run(){
        int num1=0, num2=0;

        // キーボードの準備
        InputStreamReader isr=new InputStreamReader(System.in);
        BufferedReader br=new BufferedReader(isr);

        // 2 つの整数の入力
        System.out.println("【加算】2 つの整数を入力してください。");
        try{
            num1=Integer.parseInt(br.readLine());
            num2=Integer.parseInt(br.readLine());
        }catch(IOException e){}

        // 結果出力
        System.out.println(num1+"+"+num2+"="+(num1+num2));
    }
}
```

この例を参考にして、

1．キーボードから数値を入力して、なんらかの計算を行い、結果を表示するスレッドを各自宣言しなさい。

2．main()メソッドから各自のスレッドを起動しなさい。

上の加算を実行するスレッドを起動する main()メソッドと実行例を以下に示します。

〔スレッドを起動する main()メソッド〕

```
class Assignment11_4{
    public static void main(String[] args){
        // 新しいスレッドを起動
        System.out.println("スレッドを開始します。");
        Addition add=new Addition();
        add.start();

        // メインスレッドはここで終わります
        System.out.println("メインメソッドを終了します。");
    }
}
```

〔実行例〕

```
スレッドを開始します。
メインメソッドを終了します。          ←（main()メソッドはここで終了します）
【加算】2 つの整数を入力してください。  ←（新しいスレッドによる出力です）
3
5
3+5=8
```

■難易度★☆☆

課題 5 次はタイマーを実行するスレッドです。このスレッドを起動する main()メソッドとその実行結果を下に示します。空欄を埋めてスレッドの宣言を完成させてください。

〔タイマーを実行するスレッドの宣言〕

```
class Timer extends Thread{
    // タイマー（ミリ秒）
    private int timer;

    // タイマーをコンストラクタで設定
    public Timer(int t){
        timer=t;
    }

    // タイマーの実行
    public void run(){
            指定された時間だけ一時停止するコードを記述してください
    }
}
```

〔スレッドを起動する main()メソッド〕

```
class Assignment11_5{
    public static void main(String[] args){
        int timeout=10000;
        Timer mytimer=new Timer(timeout);
        mytimer.start();
        System.out.println(timeout+"ミリ秒後に自動的に終了します");
    }
}
```

〔実行結果〕

```
10000 ミリ秒後に自動的に終了します          ←（main()メソッドはここで終了します）
10000 ミリ秒のタイマーを開始しました        ←（以後、新しいスレッドによる出力です）
10000 ミリ秒経過しました                    ←（10 秒後に表示されます）
```

■難易度★★☆

課題 6 次は与えられた範囲〔変数 from から変数 to まで〕を指定された刻み幅〔変数 skip〕で総計を求めるスレッドです。このスレッドを利用して 1～100 までの総計を以下のように分割して複数のスレッドで並列に求めたい。メインメソッドの空欄を埋めてください。

〔総計を求めるスレッドの宣言〕

```
class Sum extends Thread{
    // 総計を求める範囲と刻み幅
    private int from;
    private int to;
    private int skip;

    // 総計
    private int sum;

    // 各種値をコンストラクタで設定
    public Sum(int f, int t, int s){
        from=f;
        to=t;
        skip=s;
    }

    // 総計の計算を実行
    public void run(){
        sum=0;
        for(int i=from;i<=to;i+=skip)
            sum += i;
    }

    // 総計の取得
    public int getSum(){
        return sum;
    }
}
```

〔総計の求め方〕

(1) スレッド 1　1～100 までの偶数の合計を求めます
　　スレッド 2　1～100 までの奇数の合計を求めます
(2) それぞれのスレッドの合計を足して 1～100 の総計を表示します

〔スレッドを起動する main()メソッド〕

```
class Assignment11_6{
    public static void main(String[] args){
            総計の求め方で処理を実行して結果を表示するコードを書いてください
    }
}
```

〔実行結果〕

```
偶数の合計 2550                    ←（main()メソッドで結果を表示します）
奇数の合計 2500
総　　計 5050
```