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　２ⁿ（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、メインメソッドよりこのスレッドを起動しなさい。

〔実行例〕

```
>java Assignments11_1
結果をお待ちください。          ←（メイン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。
```

〔２ⁿ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(){
        ２ⁿ（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メインメソッド〕

```
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。メイン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とこのスレッドを起動するメイン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≧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〕

```
>java Assignment11_2
フィボナッチ数列の求めたい項を入力してください
30                                          ←（キーボード入力です）
結果をお待ちください                        ←（メイン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 A 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 A 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」の例題を参考にしなさい。

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．メイン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メインメソッドを参考として以下に示します。

〔スレッドを起動するメインメソッド〕

```
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。");
	}
}
```

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メイン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メインメソッド〕

```
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〕

```
>java Assignment11_5
10000 ミリ秒後に自動的に終了します          ←（メイン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）
```