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

- ○ ※マルチスレッドでは複数の実行箇所が同時に存在する

(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

- ○ ※各スレッドは互いにデータをやりとりしながら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

- × ※Thread クラスの機能を用いてマルチスレッド処理を記述できる

(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

- × ※どちらか一方でよい

(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

- ○ ※Thread クラスの start()メソッドによりオーバーライドされた run()メソッドが新たなスレッドで実行される

(6)複数のスレッドは、それを開始した順番に終了する

- × ※各スレッドは自身の処理が終わり次第終了する

(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

- ○ ※異なるスレッドが同一の変数にアクセスする場合は注意が必要である

(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

- ○ ※ある 1 つのスレッドがあるオブジェクト内の synchronized メソッドにアクセスしている間は、他のどのスレッドもこのメソッドにはアクセスはできない

```java
//////////////////////////////////////////////////////
// 課題１　２のｎ（１～３０）乗を順次求めて一覧表示する…
//////////////////////////////////////////////////////

class TwoToPowerOf extends Thread{
  public void run(){
    int ans;

    // ２のｎ乗を計算しながら表示
    ans=1;
    for(int i=1;i<=30;i++){
        ans*=2;
        System.out.println("2の"+i+"乗は"+ans+"です。");
    }
  }
}

class Assignment11_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。");
  }
}
```

```java
////////////////////////////////////////////////////////////
// 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…
////////////////////////////////////////////////////////////

import java.io.*;

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(){
    int ans;
    ans=getTerm();
    System.out.println("フィボナッチ数列"+n+"項目は"+ans+"です");
  }
}

class Assignment11_2{
  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);

    Fibonacci fibo=new Fibonacci(n);
    fibo.start();

    System.out.println("結果をお待ちください");
  }
}
```

```java
////////////////////////////////////////////////////////////////
// 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…
////////////////////////////////////////////////////////////////

// 会社Aの銀行口座クラス
class BankAccount
{
  private int balance=0; // 残高

  // 預金メソッド
  public synchronized void doDeposit(int m)
  {
    int tmp=balance;
    try{
      Thread.sleep(1000); // 預金処理時間をシミュレート（１秒）
    }catch(InterruptedException e){}
    tmp=tmp+m;
    System.out.println("[預金] "+m+"万円\n "+tmp+"万円残高");
    balance=tmp;
  }
  // 払戻メソッド
  public synchronized void doWithdraw(int m)
  {
    int tmp=balance;
    try{
      Thread.sleep(2000); // 払戻処理時間をシミュレート（２秒）
    }catch(InterruptedException e){}
    tmp=tmp-m;
    System.out.println("[払戻] "+m+"万円\n "+tmp+"万円残高");
    balance=tmp;
  }
}

// 社員クラス（スレッド）
class Worker extends Thread
{
  BankAccount acc;        // 銀行口座を指定
  int deposit, withdraw; // 預金・払戻金額の設定

  Worker(BankAccount acc, int dep, int wit)
  {
    this.acc=acc;
    deposit=dep;
    withdraw=wit;
  }

  public void run()
  {
    // 指定された口座へ預金を行う
    acc.doDeposit(deposit);
    // 指定された口座から払戻を行う
    acc.doWithdraw(withdraw);
  }
}

class Assignment11_3
{
  public static void main(String[] args)
  {
    // 会社Aの銀行口座を開く
    BankAccount companyA = new BankAccount();

    // 社員オブジェクトを３名分作る
    Worker[] workers = new Worker[3];

    // 各支社において預金・払戻の金額を設定する
    workers[0] = new Worker(companyA,100,75);
    workers[1] = new Worker(companyA, 20,25);
    workers[2] = new Worker(companyA, 50,20);

    // ３名の社員が並行して口座へのアクセスを開始する
    for(int i=0;i<workers.length;i++)
      workers[i].start();
  }
}

```

```java
////////////////////////////////////////////////////////////////
// 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…
////////////////////////////////////////////////////////////////

import java.io.*;

// 加算を実行するスレッドの宣言
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));
  }
}

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。");
  }
}
```

```java
////////////////////////////////////////////////////////////
// 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…
////////////////////////////////////////////////////////////

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(){
    System.out.println(timer+"ミリ秒のタイマーを開始しました");
    try{
      sleep(timer);
    }
    catch(InterruptedException e){}
    System.out.println(timer+"ミリ秒経過しました");
  }
}

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");
  }
}

```

```java
////////////////////////////////////////////////////////////
// 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…
////////////////////////////////////////////////////////////

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;
  }
}

class Assignment11_6{
  public static void main(String[] args){
    Sum odd=new Sum(1, 100, 2);
    Sum even=new Sum(2, 100, 2);
    try{
      odd.start();
      even.start();
      odd.join();
      even.join();

      int o=odd.getSum();
      int e=even.getSum();
      System.out.println("偶数の合計 "+e);
      System.out.println("奇数の合計 "+o);
      System.out.println("総　　計 "+(e+o));
    }catch(InterruptedException e){}
  }
}
```