第7章　上級プログラミング１最終課題

以下の指示に従って、クラス、オブジェクトを作成・実行せよ。なお、文中では個別に指示しないが、変更を行ったら適宜コンパイルすること。本課題はレポート提出を求める。締め切りは6月6日（水）23:59までとする。Scombにて提出すること。レポートをPDF形式で提出し、加えてソースファイルCalculator.javaを提出せよ。

7.1　スレッドを使った数値加算プログラム

本課題の最後にMap.javaとMapMain.javaのソースプログラムを提示している。これらを利用して、数字の羅列をファイルから読み取って和を求めるプログラムをつくるが、スレッドを利用して分散処理を行う練習も兼ねる。以下の使用を満たすCalculatorクラスを実装せよ。

- フィールドとして次の4つの変数を用意せよ。なお、Integerクラスはint型のラッパークラスである。これを適切に初期化せよ。
  - `ArrayList<Integer> as`
  - `ArrayList<Map> mp`
  - `long sum`
- 以下の処理を行うコンストラクタを作れ。ただし、コンストラクタはStringオブジェクトである変数fileNameを引数にとる。
  - fileNameで指定されるテキストファイルをFileInputStream、InputStreamReader、BufferedReaderを使って1行づつ読み込む。

  - 読み込んだ各行を整数値に直し (Integer オブジェクトを作れ)、フィールド as に追加する。

- 次の処理を行うメソッド map を作れ。ただし、引数は int 型の変数 n のみをとり、戻り値なし (void) とする。

  - n 回、以下を繰り返す。i 番目の繰り返しのとき、
    * Map クラスのオブジェクト m を生成する。このとき、コンストラクタの引数には Calculator オブジェクトを渡す必要があるが、このクラスのオブジェクト自身を使いたいので、this 変数を引数に入れる。
    * m の assign メソッドを呼び出し、引数にはフィールド as、i、n を渡す。
    * オブジェクト m を、リストであるオブジェクト mp に追加する。
    * Map クラスはスレッドクラスを継承しているので、オブジェクト m を別スレッドとして処理を開始する。

- 次の処理を行うメソッド addSum を作れ。ただし、引数は long 型の変数 *sum* のみをとり、戻り値なし (void) とする。また、一時点で単一のスレッドからのみ実行できるようにせよ。

  - 引数 sum の値を sum に加える。

- 次の処理を行うメソッド getSum を作れ。ただし、引数はなし、戻り値は long 型であるとせよ。また、メソッド内で適切に例外処理を行うこと。

  - フィールド mp は ArrayList オブジェクトであり、要素それぞれが別スレッドとして動作しているため、これらがすべて処理を終えるまで処理を停止する (join メソッドを使え)。
  - これらのスレッドがすべて終了したら、フィールド sum の値を返す。

  以上のコンストラクタとメソッドを持つ Calculator クラスをソースファイル Calculator.java に実装せよ[1]。

[1] ほかのファイルにソースコードを含めないこと。

これらのソースコードをコンパイルし、実行してみよ。この結果とMapMain.javaのmainメソッドにある`mr.map(3)`を`mr.map(1)`としたときに得られる結果を比較し、実施している処理についての解説を交えてなぜそのような結果になるかについて説明せよ。

【Map.java】

```
import java.util.ArrayList;

public class Map extends Thread {
  ArrayList<Integer> al = new ArrayList<Integer>();
  int label = 0;
  int total = 0;
  Calculator cal;

  Map(Calculator cal) {
    this.cal = cal;
  }

  void assign(ArrayList<Integer> al, int i, int total) {
    this.al = al;
    this.label = i;
    this.total = total;
  }

  public void run() {
    long sum = 0;
    int len = this.al.size();
    long start = System.nanoTime();
    for (int j = this.label * len / this.total;
              j < (this.label + 1) * len / this.total; j++) {
      sum += this.al.get(j);
      try {
        Thread.sleep(1);
      } catch (Exception e) {
        e.printStackTrace();
      }
    }
    cal.addSum(sum);
    long end = System.nanoTime();
    System.out.println("ThreadTime:" + (end - start) / 1000000f + "ms");
```

```
    }
}
```

【MapMain.java】

```
public class MapMain {

  public static void main(String[] args) {
    Calculator mr = new Calculator("source.txt");
    long start = System.nanoTime();
    mr.map(3);
    System.out.println("SUM:" + mr.getSum());
    long end = System.nanoTime();
    System.out.println("Time:" + (end - start) / 1000000f + "ms");
  }
}
```