プログラミングA 第10回・演習

演習の提出は Moodle で木曜日までに行ってください。各演習ごとに提出ファイルを zip 等で一つのファイルにまとめて該当する Moodle の課題に提出しなさい。この資料や関係するコードをインターネットなどに公開することは著作権上、禁止されています。

1 演習 1

添付の CountdownThread を実行すると、10 個のスレッドインスタンスがカウントダウンを始めるが、全てのカウントダウンを待たずして main スレッドが”FINISHED” と出力してさきに終了してしまう。CountdownThread の main メソッドを修正して、main スレッドが他のスレッドインスタンスによるカウントダウンの終了を待ち合わせて最後に”FINISHED” と出力して終了するようにし、修正したソースコードを提出せよ。

参考資料：CountdownThread.java

```
public class CountdownThread extends Thread {
   private int value = 0;

   public CountdownThread(int v) {
      value = v;
   }

   public void run() {
      for(int i = 10; i >= 0; i--) {
         try {
            Thread.sleep(i * value);
         } catch(InterruptedException ie) {
         }
         System.out.println(Thread.currentThread() + ": " + i * value);
      }
   }

   public static void main(String[] args) {
      Thread[] threads = new Thread[10];
      for(int i = 0; i < 10; i++) {
         threads[i] = new CountdownThread(i + 1);
         threads[i].start();
      }
      System.out.println("FINISHED.");
   }
}
```

2　演習 2

添付の Door を実行して、スレッド DoorUser の三つのインスタンスが並行動作することで以下に示すような出力を標準出力に得ながら停止せずに実行し続けたい（ただし出力は一例であり、環境や状況により出力順は異なる）。つまり、mutex.lock() と mutex.unlock() の間の箇所は、同時点でただ一つのスレッドインスタンスから実行されることを保証したい。このとき、Door および DoorUser を編集せずに、Mutex クラスを完成させて Mutex クラスを提出せよ。

```
XXXX passed.
ZZZZ passed.
ZZZZ passed.
• • •
ZZZZ passed.
YYYY passed.
• • •
```

ヒント: Guarded Suspension パターンを応用すること。lock メソッド内では wait、unlock メソッド内では notifyAll を用いるとよい。

参考資料：Door.java

```
public class Door {
  private String name = "----";
  private final Mutex mutex = new Mutex();

  public void pass(String s) throws InterruptedException {
    mutex.lock();
    try {
      name = s;
      Thread.sleep(1);
      check(s);
      System.out.println(name + " passed.");
    } finally {
      mutex.unlock();
    }
  }

  private void check(String s) {
    if (!s.equals(name)) {
      System.out.println(s + " does not match to the name: " + name);
      System.exit(-1);
    }
  }

  public static void main(String[] args) {
    Door door = new Door();
```

```
      new DoorUser(door, "XXXX").start();
      new DoorUser(door, "YYYY").start();
      new DoorUser(door, "ZZZZ").start();
   }
}
```

参考資料：DoorUser.java

```
public class DoorUser extends Thread {
   private final Door door;
   private final String name;

   public DoorUser(Door d, String s) {
      door = d;
      name = s;
   }

   public void run() {
      while (true) {
         try {
            door.pass(name);
         } catch(InterruptedException ie) {
         }
      }
   }
}
```

参考資料：Mutex.java

```
public final class Mutex {
   private boolean busy = false;

   public synchronized void lock() {

   }

   public synchronized void unlock() {

   }
}
```

3　演習 3

以下の 2 点を満たすように修正した後の FileCopy を提示せよ。

- 例外をキャッチしたらスタックトレースを標準エラーストリームに出力するようにせよ。
- プログラムは「資源を利用した後に必ず解放（ストリームを close()）しなければならない」という観点

に照らして誤っている。修正せよ。

ヒント:

- finally の用途を考えること。
- Throwable インタフェース（従ってあらゆる例外クラス）の printStackTrace メソッドを活用すること。

参考資料：FileCopy.java

```
import java.io.*;
public class FileCopy {
  public static void main(String[] args) {
    int data = 0;
    try {
      InputStream is= new BufferedInputStream(new FileInputStream(args[0]));
      OutputStream os= new BufferedOutputStream(new FileOutputStream(args[1]));
      while ((data = is.read()) != -1) {
        os.write(data);
      }
      is.close();
      os.close();
    } catch (Exception e) {
      // Throwing new RuntimeException by wrapping original exception
      throw new RuntimeException(e);
    }
  }
}
```

4 演習 4

標準入力からキーボードによる文字列の入力を一行ずつ受け付けて、ファイルおよび標準出力に書きだすプログラム FileWriting を作成し、提出せよ。ただし以下の全てを満たすこと。

- プログラムの起動時に引数で指定した名前のファイル（例えば test.txt）に一行ずつ書き込むこと。利用例: java FileWriting test.txt
- 同じ行を標準出力にも一行ずつ出力すること。
- ただし、出力にあたってはバッファリングを用いないこと。つまり、BufferedWriter を用いてはならない。
- 入力待ちの状態で先頭から END とだけ入力して改行した場合は、END をファイルや標準出力に書きださずに実行を終了すること。
- 入出力ストリームを正しく開き、正しく開放すること。

5　演習 5

指定されたファイルをコピーするプログラム CopyingFile を作成し、提出せよ。ただし以下の全てを満たすこと。

- ファイルのコピー開始からコピー終了までに要した時間（ms）を標準出力に出力して終了すること。
- プログラムの起動時に指定した第一引数のファイルを、第二引数のファイルとしてコピーすること。
- さらに、起動時に第三引数で true と指定するとバッファリングしながらコピーし、false と指定するとバッファリングせずにコピーすること。
- ディレクトリのコピーは扱わない。

期待する実行例 1: test.pdf を copy.pdf として、バッファリングなしでコピーする場合

```
java CopyingFile test.pdf copy.pdf false
23013
```

期待する実行例 2: test.pdf を copy.pdf として、バッファリングしながらコピーする場合

```
java CopyingFile test.pdf copy.pdf true
117
```