プログラミングA 第8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 Factorial について以下を完成させて、提出せよ。ただし、各メソッドにおいて与えられた引数が 0 以上の整数であることのチェックは省いて良い。

- static int factorial1(int n): 再帰呼び出しを用いずに、引数で与えられた 0 以上の整数の階乗を計算して返すこと。
- static int factorial2(int n): 再帰呼び出しにより、引数で与えられた 0 以上の整数の階乗を計算して返すこと。つまり、内部で factorial2 を呼び出すこと。
- static int factorial3(int n): 再帰呼び出しを用いずに、もしくは用いて、引数で与えられた 0 以上の整数の階乗を計算して返すこと。ただし、計算途中で値が int 型の最大値を超える場合は、RuntimeException のインスタンスを投げて終了すること。factorial3 の内部で factorial1 および factorial2 を呼び出さないこと（メソッドの定義を再利用して構わない）。ヒント: int の最大値は Integer.MAX_VALUE で得られる。

以上により、添付の Factorial を実行すると以下を標準出力に得ること。ただし例外出力の位置は異なっていて構わない。

```
1
1
3628800
3628800
Exception in thread "main" java.lang.RuntimeException
```

参考資料：Factorial.java

```
public class Factorial {
  public static int factorial1(int n){

  }
  public static int factorial2(int n){

  }
  public static int factorial3(int n){

  }
  public static void main(String[] args) {
    System.out.println(factorial1(0));
    System.out.println(factorial2(0));
    System.out.println(factorial1(10));
```

```
    System.out.println(factorial2(10));
    System.out.println(factorial3(13));
  }
}
```

2　演習 2

city パッケージのクラス House から利用できる形で、animal パッケージ下に Animal, Dog, Cat の 3 クラスを作成し、3 クラスの全てを提出せよ。ただし以下の条件を満たすこと。

- Animal は抽象クラスとし、インスタンスを作成できない。
- Cat のインスタンスを Cat クラスの外部からは生成できない。つまり、Cat のコンストラクタを Cat の外部からは呼び出し不可とすること。

House の main メソッドを実行して得られる出力は以下の通り。

ワン
ワン
ワン
ニャー

参考資料：House.java

```
package city;

import animal.Animal;
import animal.Cat;
import animal.Dog;

public class House {
  public static void main(String[] args) {
    Animal[] animals = new Animal[4];
    animals[0] = new Dog();
    animals[1] = new Dog();
    animals[2] = new Dog();
    animals[3] = Cat.getInstance();
    // animals[3] = new Cat();
    for(int i = 0; i < animals.length; i++) {
      System.out.println(animals[i].say());
    }
  }
}
```

3　演習 3

約 2 秒ごとに「XX」を標準出力に出力するスレッド ThreadXX と、約 0.5 秒ごとに「YY」を出力するスレッド ThreadYY をそれぞれ別のクラスとして作成し、クラス ThreadEx の main から実行できることを確認の上、ThreadXX および ThreadYY の全体を提出せよ。ただし、ThreadXX はクラス Thread のサブクラスとして作成し、ThreadYY は Runnable インタフェースを実装したクラスとして作成すること。

参考資料：ThreadEx.java

```
public class ThreadEx {
   public static void main(String[] args) {
      new ThreadXX().start();
      new Thread(new ThreadYY()).start();
   }
}
```

4　演習 4

クラス CalcClient は無限ループにおいて、クラス Calc が持つインスタンスフィールド value（初期値 0）の値を 1 つ増やして直後に 1 つ減らすことを繰り返す。従って value の値は 0 または 1 であることが期待されるが、CalcClient の main メソッドを実行すると、CalcClient がスレッドとして複数並行実行されるため、value の値はしばしば 2 や 3 になってしまう。CalcClient が複数平行実行されてもこの問題を生じないように CalcClient を修正し、CalcClient の全体を提出せよ。ただし、Calc には一切手を加えないこと。

ヒント: synchronized ブロックを活用せよ。

参考資料：CalcClient.java

```
public class CalcClient extends Thread {
   Calc calc = null;
   public CalcClient(Calc c) {
      calc = c;
   }
   public void run() {
      while(true) {
         calc.increment();
         calc.decrement();
      }
   }
   public static void main(String[] args) {
      Calc c = new Calc();
      new CalcClient(c).start();
      new CalcClient(c).start();
   }
}
```

参考資料：Calc.java

```
class Calc {
   int value = 0;
   void increment() {
      value++;
      System.out.println(Thread.currentThread() + ": " + value);
   }
   void decrement() {
      value--;
      System.out.println(Thread.currentThread() + ": " + value);
   }
}
```