プログラミングA 第8回・演習　回答

1　演習 1

参考資料：Factorial.java

```
public class Factorial {
  public static int factorial1(int n){
    int result = 1;
    if (n == 0)
      return result;
    for (int i = n; i > 0; i--)
      result *= i;
    return result;
  }
  public static int factorial2(int n){
    return (n == 0) ? 1 : factorial2(n - 1) * n;
  }
  public static int factorial3(int n){
    if (n == 0)
      return 1;
    int lastFactorial = factorial3(n - 1);
    if (Integer.MAX_VALUE / lastFactorial < n)
      throw new RuntimeException();
    return lastFactorial * 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));
   }
}
```

1.1　参考

- 脆弱性のない、安全なプログラムを開発するためのコーディング規約
  - JPCERT コーディネーションセンター, Java セキュアコーディングスタンダード CERT/Oracle 版 http://www.jpcert.or.jp/java-rules/
- 00. 入力値検査とデータの無害化 (IDS)
- 01. 宣言と初期化 (DCL)
- 02. 式 (EXP)
- 03. 数値型とその操作 (NUM)

- 04. オブジェクト指向 (OBJ)
- 05. メソッド (MET)
- 06. 例外時の動作 (ERR)
- 07. 可視性とアトミック性 (VNA)
- 08. ロック (LCK)
- 09. スレッド API (THI)
- 10. スレッドプール (TPS)
- 11. スレッドの安全性に関する雑則 (TSM)
- 12. 入出力 (FIO)
- 13. シリアライズ (SER)
- 14. プラットフォームのセキュリティ (SEC)
- 15. 実行環境 (ENV)

NUM00-J. 整数オーバーフローを検出あるいは防止する (事前条件テスト)

```
static final int SafeMultiply(int left, int right)
  throws ArithmeticException {
    if (right > 0 ? left > Integer.MAX_VALUE/right
         || left < Integer.MIN_VALUE/right :
         (right < -1 ? left > Integer.MIN_VALUE/right
          || left < Integer.MAX_VALUE/right :
          right == -1 && left == Integer.MIN_VALUE) )
      throw new ArithmeticException("Integer overflow");
    return left * right;
  }
```

2 演習 2

参考資料：Animal.java

```
package animal;
public abstract class Animal {
  public abstract String say();
}
```

参考資料：Dog.java

```
package animal;
public class Dog extends Animal {
  public String say() {
    return " ワン ";
  }
}
```

参考資料：Cat.java

```
package animal;
public class Cat extends Animal {
   private static Cat instance = new Cat();
   private Cat() {}
   public static Cat getInstance() {
      return instance;
   }
   public String say() {
      return " ニャー ";
   }
}
```

3 演習 3

参考資料：ThreadXX.java

```
class ThreadXX extends Thread {
   public void run() {
      while(true) {
         try {
            Thread.sleep(2000);
         } catch(InterruptedException e){
         }
         System.out.println("XX");
      }
   }
}
```

参考資料：ThreadYY.java

```
class ThreadYY implements Runnable {
   public void run() {
      while(true) {
         try {
            Thread.sleep(500);
         } catch(InterruptedException e){
         }
         System.out.println("YY");
      }
   }
}
```

4 演習 4

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) {
         synchronized(calc) {
            calc.increment();
            calc.decrement();
         }
      }
   }
   public static void main(String[] args) {
      Calc c = new Calc();
      new CalcClient(c).start();
      new CalcClient(c).start();
   }
}
```