プログラミングA 第7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

クラス GcQuiz2 の main メソッド内の実行時点「※」でガーベッジコレクションを実行するとき、info の値が” あ”〜” こ” の全ての X および Y について、それぞれ以下の選択肢のいずれであるのか、および、その理由を簡潔に説明せよ。

(a) 存続する（参照されている）
(b) ガーベッジコレクタに収集される
(c) そもそも生成されていない

「info の値: 選択肢 理由」の形式で回答すること。以下は例であり、記述が正しいとは限らない。

あ：（c）〜のため、〜されることはない。
い：（a）〜の初期化時に生成され、〜により参照され続けている。
・・・
こ：（c）〜のため、〜されることはない。

参考資料：GcQuiz2.java

```
class X {
    String info = null;
    X x = null;
    X(String s) {
        info = s;
    }
}

class Y extends X {
    static Y last = null;
    Y(String s, X newXX) {
        super(s);
        x = newXX;
        last = this;
    }
}

public class GcQuiz2 {
    static X x = new X("あ");
    Y y = new Y("い", x);
    public static void main(String[] args) {
        X[] a = new X[3];
```

```
        a[0] = new X("う");
        a[1] = new Y("え", a[0]);
        a[2] = new Y("お", new X("か"));
        if(new X("き") == new X("き")) {
            a[2] = new X("く");
        } else {
            a[2] = a[1];
        }
        if(new Y("け", x) instanceof X) {
            a[0] = new X("こ");
        }
        a[1] = null;
        System.gc();
        // ※この時点
    }
}
```

2　演習 2

クラス ExceptionExam の main メソッドを実行すると、例外が投げられて終了する。例外が投げられても処理を続行し以下を標準出力に出力するように ExceptionExam を修正し、クラス ExceptionExam の全体を zip としてまとめてアップロードせよ。

例外は: java.lang.ArrayIndexOutOfBoundsException: 3
プログラムを終了します。

参考資料：ExceptionExam.java

```
class ExceptionExam {

  public static void main(String[] args) {
     int[] array = new int[3];
     for(int i = 0; i < 4; i++){
        array[i] = 0;
     }
     System.out.println("プログラムを終了します。");
  }

}
```

3　演習 3

添付のクラス Length について以下のそれぞれを完成させて、Length のクラスメソッド encode および decode を提出せよ。

- encode は、引数に与えられた char 型の配列 data において文字 i が j 個（ただし $0 < j < 10$）連続している場合に、i と j を連結して並べた新たな文字列を char 型の配列として戻すこと。例えば文字列 aaabbbbbcc は 3a5b2c に変換し、aaaaaaaaaa は 9a1a に変換すること。
- decode には引数に、encode の実行により変換された文字列を char 型の配列 data として与えられるとする。ここで decode は、与えられた文字列を変換前の文字列へと復元し、復元結果の文字列を char 型の配列として戻すこと。例えば文字列 3a5b2c は aaabbbbbcc に変換すること。
- decode における復元処理において、与えられた文字列が期待する規則性に従っていない場合は、例外として RuntimeException もしくはそのサブクラスのインスタンスを投げて終了すること。例えば文字列 aaabbbbbcc が与えられたら、例外を投げて終了すること。

以上により、Length の main() を実行して以下を得ること。ただし 4 行目以降の例外の出力については、RuntimeException のサブクラス（例えば NumberFormatException）でも構わない。

```
aaaaaaaaaabbbbbcc
9a1a5b2c
aaaaaaaaaabbbbbcc
Exception in thread "main" java.lang.RuntimeException
. . .
```

参考資料：Length.java

```
public class Length {

  public static char[] encode(char[] data) {

  }

  public static char[] decode(char[] data) {

  }

  public static void print(char[] data) {
    for (int i = 0; i < data.length; i++) {
      System.out.print(data[i]);
    }
    System.out.println();
  }

  public static void main(String[] args) {
    char input[] = {
        'a','a','a','a','a','a','a','a','a','a',
        'b','b','b','b','b',
        'c','c',
    };
    print(input);
    print(encode(input));
```

```
    print(decode(encode(input)));
    print(decode(input));
  }
}
```

4　演習 4

タグが入れ子構造を取る文書を処理するため、以下を満たすクラス Element, Paragraph, Text を作成しなさい。

クラス Element はあらゆる文書要素を表現する抽象クラスであり、以下を満たす。

- 引数がなく戻り値の型が void の抽象かつインスタンスメソッド print を持つ。このメソッドは文書要素の内容を出力する。
- 引数に Element 型の参照を一つとり、戻り値の型が void のインスタンスメソッド add を持つ。Element では、add の本体は空とする。

クラス Paragraph は、Element の拡張クラス（サブクラス）であり段落要素を表現する。直接に 10 個までの文書要素を内包でき、以下を満たす。

- 当該時点における Paragraph のインスタンス生成個数を持つ。
- Paragraph インスタンスは、当該インスタンスを生成した直後の段階におけるインスタンス生成個数を、当該インスタンスの番号として持つ。例えば Paragaph のインスタンスを 2 個生成すると、1 および 2 という番号がそれぞれインスタンスに割り当てられる。
- add を実行すると、実行直前に内包するインスタンス群が 9 個以内であれば、引数で指定した Element 型として参照されるインスタンスを、自身が内包するインスタンス群に追加する。実行直前に内包するインスタンス群が 10 個に達している場合は、例外として RuntimeException もしくはそのサブクラス（例えば ArrayIndexOutOfBoundsException）のインスタンスを投げ、呼び出した側でその例外をキャッチしてプログラムを終了せずに以降を続行すること。
- print を実行すると、最初に開始タグ<pN>（ただし N は当該インスタンスの番号）を標準出力に出力して改行し、続いて内包するすべてのインスタンス群の内容を出力し、最後に終了タグ</pN>を出力して改行する。

クラス Text は、Element の拡張クラス（サブクラス）でありテキスト要素を表現する。他の文書要素を内包せず、以下を満たす。

- 具体的なテキストを表す String を引数にとるコンストラクタを持つ。
- print を実行すると、最初に開始タグ<t>を標準出力に出力し、続いてテキストを出力し、最後に終了タグ</t>を出力して改行する。

以上を応用して、例えば添付の TagMain クラスを実行すると以下に示す期待結果を標準出力に得る。このとき、if 文や switch・case 文を使用せずに Element, Paragraph, Text を作成し、zip として添付してください。

```
<p1>
<t>aabbcc</t>
<t>XYZ</t>
<p2>
<t>10000</t>
</p2>
<p3>
</p3>
<p4>
</p4>
<p5>
</p5>
<p6>
</p6>
<p7>
</p7>
<p8>
</p8>
<p9>
</p9>
</p1>
```

参考資料：TagMain.java

```
public class TagMain {
  public static void main(String[] args) {
    Element p1 = new Paragraph();
    Element p2 = new Paragraph();
    Element t1 = new Text("aabbcc");
    Element t2 = new Text("XYZ");
    Element t3 = new Text("10000");
    try {
      p1.add(t1);
      p1.add(t2);
      p2.add(t3);
      p1.add(p2);
      while(true) {
        p1.add(new Paragraph());
      }
    } catch(RuntimeException re) {
    }
    p1.print();
  }
}
```