平成１４年度　春期　ＦＥ　午後問題　Java

問８　次の Java プログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

整数（int 型）値のスタックを実現するクラスと，そのテストプログラムである。

プログラム 1 で定義されるクラス IntStack の使用方法は，次のとおりである。

(1)　**スタックに 1 件のデータを格納するには，メソッド push を用いる。スタックの容量（capacity）は，必要に応じて動的に拡張される。つまり，データを格納する時点でスタックの容量が不足しているならば，所定の増分（INCREMENT）だけスタックの容量を拡張する。**

(2)　**スタックから，直前に格納した 1 件のデータを取り出すには，メソッド pop を用いる。空のスタックに対して pop を実行すると，例外 EmptyStackException が発生する。**

(3)　**直前にスタックへ格納したデータを参照するためには，メソッド peek を用いる。**

プログラム 2 は，IntStack のテストプログラムである。端末に表示したプロンプト => に対して入力された整数値を IntStack 型のオブジェクトに格納し，空の値（改行文字だけ）が入力された時点で，その内容を表示する。

プログラム 2 の実行例を図に示す。

```
=> 1
=> 2
=> 3
=>
3
2
1
--- bottom of Stack ---
```

図　プログラム 2 の実行例

〔プログラム 1〕

```
import java.util.EmptyStackException;

Public class IntStack {
   Private static final int INITIAL_CAPACITY = 10;
   Private static final int INCREMENT = 5;
   Private int capacity = INITIAL_CAPACITY;
   Private int[] content;
   Private int n_elements = 0;

   Public IntStack() { content = new int[capacity]; }

   Public boolean empty() { return n_elements == 0; }

   Public void push(int value) {
      if (n_elements == content.length) {
         // 配列を拡張する
         int[] newContent = new int[capacity +
                                    INCREMENT];
         for (int i = 0; i < capacity; i++) {
            [ a ] ;
         }
         capacity += INCREMENT;
         content = newContent;
      }
      [ b ] = value;
   }

   Public int peek() throws EmptyStackException {
      if (n_elements > 0) {
         return [ c ] ;
      }
      throw new EmptyStackException();
   }
   Public int pop() throws EmptyStackException {
      int value = peek();
      n_elements--;
      return value;
   }
}
```

〔プログラム 2〕

```
import java.io.*;

Public class TestIntStack {
   Public static void main(String[] args) {
      IntStack stack = new IntStack();
      // 標準入力ストリームから読み込むための Reader オブジェクトを生成する
      BufferedReader in =
         new BufferedReader(new
                  InputStreamReader(System.in));
      while (true) {
         System.out.print("=> ");
         try {
            String input = in.readLine(); // 標準入力から 1 行分読み込む
            if (input.equals("")) break;
            int n = Integer.parseInt(input);
            stack.push(n);
         } catch (Exception e) {
            e.printStackTrace();
         }
      }
      while ( [ d ] ) {
         System.out.println(stack.pop());
      }
      System.out.println("--- bottom of Stack ---");
   }
}
```

なお，プログラム 2 の 8 行目で使用する InputStreamReader と BufferedReader とは，ともにパッケージ java.io に含まれるクラスである。クラス

InputStreamReaderは，入力ストリーム（端末）からのバイト入力を文字に変換する。クラス BufferedReaderは，文字入力ストリームからの入力をバッファリングし，メソッド readLine による行単位での入力処理を可能とする。

設問　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　newContent[i] = content[i].clone()

イ　newContent.set(i, content[i])

ウ　newContent[i] = content[i]

エ　newContent[i + INCREMENT] = content[i]

b，c に関する解答群

ア　content[n_elements]

イ　content[n_elements--]

ウ　content[n_elements-1]

エ　content.elementAt(n_elements)

オ　content[n_elements++]

カ　content.elementAt(n_elements--)

キ　content[n_elements+1]

ク　content.elementAt(n_elements++)

d に関する解答群

ア　true

イ　stack.n_elements >= 0

ウ　stack.peek() != null

エ　!stack.empty()

問 12　次の Java プログラムの説明及びプログラムを読んで，設問 1～3 に答えよ。

〔プログラムの説明〕

飛行機の利用者の種別（一般利用者，ゴールド利用者）と利用区間からマイレージを求めて積算マイレージを計算するプログラムである。利用者の各種別のマイレージの計算方法は，次のとおりである。

・一般利用者：マイレージ＝基準距離
・ゴールド利用者：マイレージ＝基準距離×1.25

利用区間とその基準距離を，次表に示す。

表　利用区間と基準距離

利用区間			基準距離
成田	－	関西	300
成田	－	ロサンゼルス	5,400
成田	－	パリ	6,200
関西	－	ロサンゼルス	5,700
関西	－	パリ	6,100
ロサンゼルス	－	パリ	4,000

計算したマイレージを，それまでの積算マイレージに加える。利用者は積算マイレージを無料往復チケットに交換することができる。

プログラムは，次の五つのクラスと一つのインタフェースで構成されている。

MileageTest
メソッド main をもつテスト用のクラスであり，次の処理を行う。

(1)　一般利用者とゴールド利用者のインスタンスを生成し，利用者名，積算マイレージの情報を初期化する。

(2)　新しい積算マイレージを計算し，実行結果を出力する。

コンストラクタ MileageTest の引数は，**利用者名，出発地，到着地，無料往復チケット指定**である。**無料往復チケット指定**には，無料往復チケットに交換する場合は交換に必要なマイレージを指定し，無料往復チケットに交換しない場合は 0 を指定する。

Passenger
利用者を表す抽象クラスである。抽象メソッド addMileage は，積算マイレージを計算する。メソッド awardTravel は，無料往復チケットの交換が可能か否かの判定を行い，可能であれば積算マイレージから無料往復チケット交換に必要なマイレージを減算し，不可能であれば例外処理によってメッセージを出力する。メソッド getMileage は，積算マイレージを返す。メソッド getName は，利用者名を返す。

NormalPassenger
一般利用者を表すクラスであり，積算マイレージを計算するメソッド addMileage を実装する。

GoldPassenger
ゴールド利用者を表すクラスであり，積算マイレージを計算するメソッド addMileage を実装する。

NotEnoughMileageException
無料往復チケットに交換できない場合の処理を行うクラスである。

MileageServices
プログラムで使用する定数を宣言しているインタフェースである。

〔プログラム〕

(行番号)

```
 1  public class MileageTest implements
                                  MileageServices {
 2    public static void main(String args[]) {
 3      NormalPassenger taro = new NormalPassenger
                                    ("Taro", 0);
 4      GoldPassenger mark = new GoldPassenger
                                  ("Mark",100000);
 5      NormalPassenger june = new
                     NormalPassenger("June", 0);
 6      GoldPassenger jiro = new GoldPassenger
                                   ("Jiro",50000);
 7      new MileageTest(taro, NARITA, PARIS,
 8                    DOMESTIC_ROUND_TRIP);
 9      new MileageTest(mark, LOSANGELES, PARIS,
10                    ASIA_PACIFIC_ROUND_TRIP);
11      new MileageTest(june, PARIS, KANSAI,
                                  US_ROUND_TRIP);
12      new MileageTest(jiro, KANSAI, NARITA, 0);
13      new MileageTest(taro, PARIS, NARITA,
                             DOMESTIC_ROUND_TRIP);
14    }
15    public MileageTest(Passenger passenger, int
                                    from, int to,
16                     int awardTrip) {
17      passenger.addMileage(from, to);
18      System.out.println("\n" +
                              passenger.getName() +
19                    "'s mileage: " +
                          passenger.getMileage());
20      if ( [      a      ] != 0)
21        try {
22          passenger.awardTravel(awardTrip);
23          System.out.println("You get an award
                                      trip.\n" +
24                         "Your mileage is now " +
25                   passenger.getMileage() + ".");
26        } catch (NotEnoughMileageException e) {
27          System.out.println(e);
28        }
29    }
30  }
31
32  interface MileageServices {
33    final static int NARITA = 0;
34    final static int KANSAI = 1;
35    final static int LOSANGELES = 2;
36    final static int PARIS = 3;
37    final static int[][] MILEAGE =
                           {{  0, 300,5400, 6200},
38                         { 300,    0, 5700, 6100},
39                         {5400, 5700,    0, 4000},
40                         {6200, 6100, 4000,    0}
41                         };
42    final static int DOMESTIC_ROUND_TRIP = 15000;
43    final static int ASIA_PACIFIC_ROUND_TRIP
                                        = 20000;
44    final static int US_ROUND_TRIP = 40000;
45
46    final static double NORMAL = 1.00;
47    final static double GOLD = 1.25;
48  }
49
50  [      b      ] class Passenger implements
                               MileageServices
{
51    int totalMileage;
52    String name;
53
54  [      c      ] (String name, int totalMileage) {
55      this.name = name;
56      this.totalMileage = totalMileage;
57    }
58    public abstract void addMileage(int from,
                                        int to);
59    public void awardTravel(int award) throws
60                    NotEnoughMileageException
{
61      if (award > [      d      ] )
62        throw new NotEnoughMileageException(name);
63      else
64        totalMileage -= award;
65    }
66    public int getMileage() {
67      return totalMileage;
68    }
69    public String getName() {
70      return name;
71    }
72  }
73
74  class NormalPassenger extends Passenger {
75    NormalPassenger(String name, int totalMileage)
                                                {
76      super(name, totalMileage);
77    }
78    public void addMileage(int from, int to) {
79      totalMileage += (int)(MILEAGE[from][to] *
                                      NORMAL);
80    }
81  }
82
83  class GoldPassenger extends Passenger {
84    GoldPassenger(String name, int totalMileage)
                                                 {
85      super(name, totalMileage);
86    }
87    public void addMileage(int from, int to) {
88      totalMileage += (int)(MILEAGE[from][to] *
                                        GOLD);
89    }
90  }
91
92  class NotEnoughMileageException extends
                                      Exception {
93    String name;
94    public NotEnoughMileageException(String name)
                                                 {
95      this.name = name;
96    }
```

```
97    public String toString() {
98      return "Sorry, your mileage isn't enough for
                                                  " +
99            "your award trip, " + name + ".";
100   }
101 }
```

プログラムの実行結果を図に示す。

```
Taro's mileage: 6200
Sorry, your mileage isn't enough for your award trip,
Taro.

Mark's mileage: 105000
You get an award trip.
Your mileage is now 85000.

June's mileage: 6100
Sorry, your mileage isn't enough for your award trip,
June.

Jiro's mileage: 50375

Taro's mileage: 12400
Sorry, your mileage isn't enough for your award trip,
Taro.
```

図　実行結果

設問１　プログラム中の [　　] に入れる正しい答えを，解答群の中から選べ。

解答群

ア	abstract	イ	awardTrip
ウ	GoldPassenger	エ	NormalPassenger
オ	Passenger	カ	private
キ	public	ク	totalMileage

設問２　プログラムの行番号 5 を次のとおりに変更した場合，行番号 11 を実行した後の，利用者 June の積算マイレージの値として正しい答えを，解答群の中から選べ。

```
NormalPassenger june =
    new NormalPassenger("June", 50000);
```

解答群

ア	3900	イ	10000
ウ	16100	エ	96100

設問３　プログラムの行番号 37～41 を次のとおりに変更したとき，同じ結果を得るために，行番号 78 の直後及び行番号 87 の直後に挿入する内容として正しい答えを，解答群の中から選べ。

```
final static int[][] MILEAGE =
                        {{   0},
                         { 300,    0},
                         {5400, 5700,    0},
                         {6200, 6100, 4000, 0}
                        };
```

解答群

ア
```
if (from == 0) {
    int tmp = from;
    from = to;
    to = tmp;
}
```

イ
```
if (from != to) {
    int tmp = from;
    from = to;
    to = tmp;
}
```

ウ
```
if (from > to) {
    int tmp = from;
    from = to;
    to = tmp;
}
```

エ
```
if (from < to) {
    int tmp = from;
    from = to;
    to = tmp;
}
```

平成１４年度　秋期　ＦＥ　午後問題　Java

問８　次の Java プログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕
ディジタル論理回路シミュレータを作成するためのクラスとテスト用クラスである。

(1)　ゲートを表す抽象クラス Gate のサブクラスとして，NOT ゲートを表すクラス NotGate 及び AND ゲートを表すクラス AndGate を定義する。

(2)　Gate のすべてのサブクラスは，メソッド connectOutputTo，getInput 及び getOutput を継承し，メソッド tick を実装する。

(3)　メソッド connectOutputTo は，ゲートの出力信号線を，指定されたゲートの入力信号線に接続する。

(4)　メソッド getInput 及び getOutput は，ゲートの入力信号線及び出力信号線をそれぞれ返す。

(5) メソッド tick は，各ゲート固有の演算を実行し，出力信号線に true 又は false のいずれかの値を出力する。

(6) クラス Wire は，入力信号線又は出力信号線を表す。

(7) 各ゲートの出力信号線は，１本とする。

(8) テスト用クラス LogicCircuitTest で作成する回路及び動作の例を，次に示す。メソッド main は，演算結果の値を標準出力に表示する。

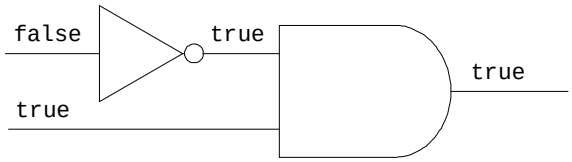

図　LogicCircuitTest で作成する回路及び動作の例

〔プログラム〕

```
class Wire {
    private boolean value;
    public boolean getValue() { return value; }
    public void setValue(boolean value) { this.value
                                          = value; }
}

abstract class Gate {
    protected Wire[] input;
    protected Wire output;
    public Gate(int nInputs) {
        input = [ a ];
        for (int i = 0; i < nInputs; i++) { input[i]
                                          = new Wire(); }
        output = new Wire();
    }
    public void connectOutputTo(Gate otherGate,
                                      int nthInput) {
        try {
            [ b ];
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public Wire getInput(int n) { return input[n]; }
    public Wire getOutput() { return output; }
    abstract public void tick();
}

class NotGate extends Gate {
    public NotGate() { super(1); }
    public void tick() { output.setValue( [ c ] ); }
}

class AndGate extends Gate {
    public AndGate() { super(2); }
    public void tick() { output.setValue( [ d ] ); }
}

public class LogicCircuitTest {
    public static void main(String[] args) {
        Gate not = new NotGate();
        Gate and = new AndGate();
        not.connectOutputTo(and, 0);
        not.getInput(0).setValue(false);
        and.getInput(1).setValue(true);
        not.tick();
        and.tick();
        System.out.println(and.getOutput().
                                    getValue());
    }
}
```

設問　プログラム中の [　　] に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　new Wire(nInputs)

イ　new Wire[nInputs]

ウ　new Wire()

エ　new Wire[] {new Wire()}

b に関する解答群

ア
```
input[nthInput].setValue
          (otherGate.output.getValue())
```

イ
```
input[nthInput] = otherGate.output
```

ウ
```
otherGate.input[nthInput].
            setValue(output.getValue())
```

エ
```
otherGate.input[nthInput] = output
```

c, d に関する解答群

ア
```
input[0].getValue() ||
                    input[1].getValue()
```

イ
```
!input[0].getValue() ||
                   !input[1].getValue()
```

ウ
```
input[0].getValue() &&
                    input[1].getValue()
```

エ
```
!input[0].getValue() &&
                   !input[1].getValue()
```

オ
```
input[0].getValue() !=
                    input[1].getValue()
```

カ　input[0].getValue()

キ　!input[0].getValue()

ク　1 - input[0].getValue()

問 12　次の Java プログラムの説明及びプログラムを読んで，設問に答えよ。

〔プログラムの説明〕

プログラム１は，コンマ区切り（CSV 形式）のデータを解析するための汎用クラス CSVParser である。プログラム２は，クラス CSVParser を拡張して，コンマ区切り形式の入力データファイルをタグ付き形式の出力データに変換するプログラムである。

クラス CSVParser は，コンストラクタに指定されたコンマ区切りのデータファイルをメソッド parse で解析し，次に示す状態に応じて各メソッドを呼び出す。

状態	呼び出されるメソッド	引数
ファイルの読込みを開始した	startDocument	なし
新しいレコードに達した	startRecord	n：レコード番号
レコード中のフィールドを１項目読み込んだ	value	chars：フィールドの値 n：フィールド番号
レコードの終端に達した	endRecord	n：レコード番号
ファイルの処理を完了した	endDocument	なし

プログラム 2 の入力データファイル test.csv と出力結果を，それぞれ図１及び図２に示す。

```
Ichiro Tanaka,tanaka@sales.example.com,111-1111
Jiro Yamada,yamada@eng.example.com,222-2222
Saburo Suzuki,suzuki@mkt.example.com,333-3333
```

図１　入力データファイル test.csv

```
<addressbook>
  <person id="1">
    <name>Ichiro Tanaka</name>
    <email>tanaka@sales.example.com</email>
    <phone>111-1111</phone>
  </person>
  <person id="2">
    <name>Jiro Yamada</name>
    <email>yamada@eng.example.com</email>
    <phone>222-2222</phone>
  </person>
  <person id="3">
    <name>Saburo Suzuki</name>
    <email>suzuki@mkt.example.com</email>
    <phone>333-3333</phone>
  </person>
</addressbook>
```

図２　出力結果

入力データの形式は次のとおりである。

(1)　１レコードは，三つの項目からなり，各項目はコンマ（“,”）で区切られている。レコードの終端には改行文字がある。

(2)　項目には，氏名，電子メールアドレス及び電話番号があり，１レコード中にこれらの項目が１個ずつ，この順に並ぶ。

①　項目にコンマが含まれることはない。

②　項目は省略できず，空白文字だけからなる項目はない。

出力データの形式は次のとおりである。

(1)　値を開始タグと終了タグとで囲んだデータの単位を，要素と呼ぶ。

①　開始タグは <タグ名> と記述し，終了タグは </タグ名> と記述する。

②　開始タグには，<タグ名 属性名="属性値"> という形式で，属性を記述できる。

③　以降の説明では，この要素のことを，“要素 タグ名”と呼ぶ。

(2)　要素は入れ子にすることができる。すなわち，ある要素の中に他の要素を入れることができる。

(3)　すべての開始タグと終了タグとは，対応がとれている必要がある。すなわち，ある開始タグで始まる要素は，同じタグ名をもつ終了タグによって閉じる。

入力データ形式から出力データ形式への変換手順は，次のとおりである。

(1)　入力データ全体をまとめて，要素 addressbook を作成する。

(2)　入力データの１レコードごとに，要素 person を作成する。要素 person には，レコードを読み込んだ順番を属性 id の値として付加する。

(3)　１レコード中の各項目から，レコード中での項目の出現の順に，name，email 及び phone の各要素をそれぞれ作成する。

テキストファイルの内容を読み込むのに，クラス java.io.FileReader を用いる。動作は次のとおりである。

(1)　コンストラクタにファイル名を示す文字列を与えると，当該ファイルからの入力ストリームを開く。

(2)　メソッド read は，入力ストリームから１文字を読み込んで int 型として返す。

(3)　入力ストリームの末尾に達した時点でメソッド read を呼び出すと，−1を返す。

〔プログラム 1〕

```
import java.io.FileReader;
import java.io.IOException;
import java.io.FileNotFoundException;

public class CSVParser {
  private FileReader reader;

  public CSVParser(String fileName) throws
                        FileNotFoundException {
    // テキスト入力ファイル用の reader を生成する。
    reader = new FileReader(fileName);
  }

  public void startDocument() {}
  public void startRecord(int n) {}
  [          a          ]
  public void endRecord(int n) {}
  public void endDocument() {}

  public void parse() throws IOException {
    int c;
    StringBuffer buf = new StringBuffer();
    boolean [          b          ];
    int fieldNumber = 0;
    int recordNumber = 0;

    startDocument();
    while ((c = reader.read()) != -1) { // reader
                                  から１文字読み込む。
      // メソッド read は入力ストリームに利用可能な文字
                                  がない場合-1 を返す。
      char ch = (char)c;
      switch (ch) {
        case ',':
          [          c          ];
          buf.delete(0, buf.length());
          break;
        case '\n':
          if (!endOfRecord) {
            endOfRecord = true;
            [          c          ];
            buf.delete(0, buf.length());
            fieldNumber = 0;
            [          d          ];
          }
          break;
        default:
          if (endOfRecord) {
            [          e          ];
          }
          endOfRecord = false;
          [          f          ];
      }
    }
    endDocument();
  }
}
```

〔プログラム 2〕

```
import java.io.FileNotFoundException;

public class TaggedDataGenerator extends
                                      CSVParser {
  public TaggedDataGenerator(String fileName)
    throws FileNotFoundException {
    super(fileName);
  }
  public void startDocument() {
    System.out.println("<addressbook>");
  }
  public void startRecord(int n) {
    System.out.println("\t<person
                              id=\""+n+"\">");
  }
  public void value(String chars, int n) {
    String tag = (n == 1) ? "name" : (n == 2) ? "email" :
                                        "phone";
    System.out.println("\t\t<"+tag+">"+chars+"
                                  </"+tag+">");
  }
  public void endRecord(int n) {
    System.out.println("\t</person>");
  }
  public void endDocument() {
    System.out.println("</addressbook>");
  }
  public static void main(String [] args) {
    TaggedDataGenerator parser = null;
    try {
      parser = new
               TaggedDataGenerator("test.csv");
      parser.parse();
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

設問　プログラム中の [　　] に入れる正しい答えを，解答群の中から選べ。

a に関する解答群

ア　public void value(StringBuffer chars,
                                  int n) {}

イ　public void value(String chars, int n) {}

ウ　abstract public void value(String chars,
                                  int n) {}

エ　public void value(String chars, int n);

b に関する解答群

ア　endOfRecord = null
イ　endOfRecord = false
ウ　endOfRecord = true
エ　endOfRecord

c に関する解答群

ア　value(buf, fieldNumber)
イ　value(buf.toString(), fieldNumber)
ウ　value(buf, ++fieldNumber)
エ　value(buf.toString(), ++fieldNumber)

d, e に関する解答群

ア　startDocument()
イ　startDocument(++recordNumber)
ウ　startRecord(recordNumber)
エ　startRecord(++recordNumber)
オ　endDocument()
カ　endDocument(recordNumber)
キ　endRecord()
ク　endRecord(recordNumber)

f に関する解答群

ア　buf += ch
イ　buf[buf.length] = ch
ウ　buf.append(ch)
エ　buf.append(ch.toString())

平成１４年度　春期　ＦＥ　午後解答　Java

問８

設問
a－ウ　b－オ　c－ウ　d－エ

問 12

設問１
a－イ　b－ア　c－オ　d－ク

設問２
ウ

設問３
エ

平成１４年度　秋期　ＦＥ　午後解答　Java

問８

設問
a－イ　b－エ　c－キ　d－ウ

問 12

設問
a－イ　b－ウ　c－エ　d－ク
e－エ　f－ウ