平成１６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〕

このプログラムは，数量の単位変換を行う共通機能を提供するクラス群と，それらのテストプログラムからなる。テストプログラムでは，セルシウス温度（セ氏温度，℃）及びカ氏温度（°F）の変換を行うクラスを利用する。

(1) インタフェース Converter は，単位変換を行うクラスが実装すべきインタフェースを定義する。

(2) クラス ConverterRunner は，単位変換クラスを利用するプログラムからアクセスするためのクラスである。
① メソッド setConverter は，単位変換クラスのインスタンスを設定する。
② メソッド run は，単位変換クラスのメソッド convert を呼び出す。

(3) クラス CtoF は，セ氏温度の値を，カ氏温度に変換する処理を実装するクラスである。

(4) クラス FtoC は，カ氏温度の値を，セ氏温度に変換する処理を実装するクラスである。

(5) クラス TestConverter は，単位変換クラスを利用するテストプログラムである。メソッド main は，実行すべき単位変換処理（1：カ氏－セ氏変換，2：セ氏－カ氏変換，q：終了）の選択を促してから，入力された値を指定された処理に従って変換する。実行例を図に示す。

```
% java TestConverter
Type 1(FtoC), 2(CtoF), or q(Quit): 1
Type input value: 68
20.0
```

図　クラス TestConverter の実行例

〔プログラム 1〕

```
public interface Converter {
    a
}
```

〔プログラム 2〕

```
public class ConverterRunner {
  private  b  conv;

  public void setConverter
                        ( b ) conv) {
    this.conv = conv;
  }
  public void run(String input) {
    if (conv == null) return;
    try {
      double in = Double.parseDouble(input);
      double out = conv.convert(in);
      System.out.println(out);
    } catch (NumberFormatException e) {
      System.err.println("invalid input "
                                  + input);
    }
  }
}
```

〔プログラム 3〕

```
public class CtoF  c  {
  public double convert(double input) {
    return 9.0 / 5.0 * input + 32.0;
  }
}
```

〔プログラム 4〕

```
public class FtoC  c  {
  public double convert(double input) {
    return 5.0 / 9.0 * (input - 32.0);
  }
}
```

〔プログラム 5〕

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;

public class TestConverter {
  public static void main(String[] args) {
    // 標準入力から入力を受け取る reader を
       作成する
    BufferedReader reader =
      new BufferedReader(new
          InputStreamReader(System.in));
    ConverterRunner runner = new
                      ConverterRunner();
    try {
      while (true) {
        System.out.print("Type 1(FtoC),
              2(CtoF), " + "or q(Quit): ");
        // 標準入力から 1 行読み込む
        String choice = reader.readLine();
        if ("1".equals(choice)) {
          runner.setConverter(new FtoC());
        } else if ("2".equals(choice)) {
          runner.setConverter(new CtoF());
        } else if ("q".equals(choice)) {
          break;
        } else continue;
```

```
            System.out.print("Type input
                                    value: ");
            // 標準入力から 1 行読み込む
            String value = reader.readLine();
                        d
          }
        } catch (IOException 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　abstract public void convert();

イ　public double convert(double input) { }

ウ　public double convert(double input);

エ　public double convert(String input);

b に関する解答群

ア　Converter

イ　ConverterRunner

ウ　CtoF

エ　FtoC

c に関する解答群

ア　extends Converter

イ　extends ConverterRunner

ウ　implements Converter

エ　implements ConverterRunner

d に関する解答群

ア　runner.conv.convert(Double.parseDouble
(value));

イ　runner.conv.run(value);

ウ　runner.convert(Double.parseDouble
(value));

エ　runner.run(value);

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〕

インタフェース CharIterator は，データ構造に依存せずにそのインスタンスから文字（char）を順番に取り出すための操作を定義する。CharIterator には，次のメソッドが定義されている。

public char next()
次の文字があればそれを返し，なければ java.util.NoSuchElementException を投げる。例えば，この CharIterator インスタンスが文字を n 個もっているとき，最初の呼出しで 1 番目の文字を返し，2 回目の呼出しで 2 番目の文字を返し，…というように，n 回目の呼出しまで文字を順番に返していく。n ＋ 1 回目以降の呼出しでは，NoSuchElementException を投げる。
なお，NoSuchElementException は java.lang.RuntimeException のサブクラスである。

public boolean hasNext()
次の文字があれば true を返し，なければ false を返す。

クラス CharIteratorFactory は，引数に指定したデータ型に一致した CharIterator を返すメソッドを定義する。
CharIteratorFactory には，次のクラスメソッドが定義されている。

public static CharIterator getCharIterator
(String data)
引数に指定した String から文字を順番に取り出す CharIterator を返す。引数に null が指定されたときは，NullPointerException を投げる。

public static CharIterator getCharIterator
(char[][] data)
引数に指定した char を要素型とする配列の配列（2 次元の文字配列）から文字を順番に取り出す CharIterator を返す。引数に null が指定されたときは，NullPointerException を投げる。文字を取り出す順序は，文字配列とその配列のそれぞれのインデックス値の小さい順とする。すなわち，char[][] 型の m に対し，その要素 m[i][j] を i の小さい順，同じ i に対しては j の小さい順に取り出す。

クラス CharIteratorTest は，CharIteratorFactory で定義されたメソッドをテストするためのプログラムである。メソッド main を実行すると，図の実行結果が得られる。

```
'H' '1' '6'
'2' '0' '0' '4'
```

図　CharIteratorTest.main の実行結果

〔プログラム 1〕

```
public interface CharIterator {
   public boolean hasNext();
   public char next();
}
```

〔プログラム 2〕

```
import java.util.NoSuchElementException;

public class CharIteratorFactory {
   public static CharIterator getCharIterator
                                    (String data) {
      if (data == null)
         throw new NullPointerException();
      // String データから文字を順番に返す CharIterator
         のインスタンス
      // を生成して返す。
      return ; [          a          ];
   }

   public static CharIterator getCharIterator
                                  (char[][] data) {
      if (data == null)
         throw new NullPointerException();
      //2次元の文字配列から文字を順番に返す CharIterator
         のインスタンス
      // を生成して返す。
      return ; [          b          ];
   }
}

class StringCharIterator implements CharIterator {
   private String data;
   private int index = 0;
   StringCharIterator(String data) {
      this.data = data;
   }
   // data に次の文字があるかどうかをチェックする。
   public boolean hasNext() {
      return ; [          c          ];
   }

   public char next() {
      // 次の文字がないときは，NoSuchElementException
         を投げる。
      if (index >= data.length())
         throw new NoSuchElementException();
      // data の次の文字を返し，インデックス値を更新する。
      return ; [          d          ];
   }
}

class Char2DArrayCharIterator implements
                                    CharIterator {
   private char[][] data;
   private int index1 = 0, index2 = 0;

   Char2DArrayCharIterator(char[][] data) {
      this.data = data;
   }
   public boolean hasNext() {
      // data[index1][index2]の要素があれば true を
         返し，なければ次
      // に定義されている要素を探す。次の要素がなければ
         false を返す。
      for (; index1 < data.length; index1++) {
         if (data[index1] != null
             && index2 < data[index1].length) {
            return true;
         }
         [          e          ];
      }
      return false;
   }
   public char next() {
      // メソッド hasNext を呼び出して次の要素があるか
         どうかを調べ，
      // あればその要素を返し，インデックス値を更新する。
         なければ，
      // NoSuchElementException を投げる。
      if (hasNext()) {
         return [          f          ];
      }
      throw new NoSuchElementException();
   }
}
```

〔プログラム 3〕

```
public class CharIteratorTest {
   public static void main(String[] args) {
      CharIterator itr = CharIteratorFactory.
                           getCharIterator("H16");
      printIterator(itr);

      itr = CharIteratorFactory.getCharIterator(
              new char[][] {{ '2' },
                            { '0' },
                            null,
                            { '0', '4' }});
      printIterator(itr);
   }
   private static void printIterator
                             (CharIterator itr) {
      while (itr.hasNext()) {
         System.out.print("'" + itr.next()+ "' ");
      }
      System.out.println();
   }
}
```

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, b に関する解答群

ア　getCharIterator((char[][]) data)

イ　getCharIterator((String) data)

ウ　new Char2DArrayCharIterator()

エ　new Char2DArrayCharIterator(data)

オ　new StringCharIterator()

カ　new StringCharIterator(data)

c に関する解答群

ア　index < data.length()

イ　index <= data.length()

ウ　index >= data.length()

エ　index++ < data.length()

オ　index++ <= data.length()

カ　index++ >= data.length()

d に関する解答群

ア　data.charAt(++index)

イ　data.charAt(--index)

ウ　data.charAt(index + 1)

エ　data.charAt(index++)

オ　data.charAt(index--)

カ　data.charAt(index)

e に関する解答群

ア　index1 = 0

イ　index1 = data.length

ウ　index1 = index2

エ　index2 = 0

オ　index2 = data.length

カ　index2 = index1

f に関する解答群

ア　data[index1++][index2++]

イ　data[index1++][index2]

ウ　data[index1][++index2]

エ　data[index1][--index2]

オ　data[index1][index2 + 1]

カ　data[index1][index2++]

平成１６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〕

任意のオブジェクトを格納し，取り出すプログラムである。格納されたオブジェクトを取り出す方法として，先入れ先出し法（First-In-First-Out）及び後入れ先出し法（Last-In-First-Out）がある。

(1)　抽象クラス Store は，メソッド put，get，size を定義する。

public void put(Object value)
　引数で指定した value を Store のインスタンスに先着順に格納する。格納できるオブジェクトの個数の上限は 50 である。それを超えて格納しようとしたときは DataStoreException を投げる。

public abstract Object get()
　格納されているオブジェクトを一つ取り出して返す。どのオブジェクトを取り出すかは，このメソッドを実装するサブクラスによって決まる。オブジェクトがないときは DataStoreException を投げる。

public int size()
　格納されているオブジェクトの個数を返す。オブジェクトが格納されていないときは 0 を返す。

(2)　クラス FifoStore は，抽象クラス Store のサブクラスで，メソッド get は格納されているオブジェクトのうち最初に格納されたものを取り出して返す。すなわち，先入れ先出しとなる。

(3)　クラス LifoStore は，抽象クラス Store のサブクラスで，メソッド get は格納されているオブジェクトのうち最後に格納されたものを取り出して返す。すなわち，後入れ先出しとなる。

(4)　クラス StoreTest は二つのサブクラスをテストするプログラムである。プログラム起動時に指定された引数を FifoStore 及び LifoStore に格納し，取り出す操作をする。実行例を図に示す。ただし，図中の % はシステムのコマンドプロンプトを表し，コマンドの引数は String の配列としてメソッド main に渡されるものとする。

```
%java StoreTest 起 承 転 結
0: 起 1: 承 2: 転 3: 結
0: 結 1: 転 2: 承 3: 起
```

図　クラス StoreTest の実行例

〔プログラム 1〕

```
public abstract class Store {
  Object[] data = new Object[50];
  int index = 0;
```

```
   public void put(Object value) {
      if ( [      a      ] )
         throw new DataStoreException
                              ("overflow");
      data[index++] = value;
   }
   public abstract Object get();
   public int size() {
      return [      b      ];
   }
}
```

〔プログラム 2〕

```
public class FifoStore extends Store {
   public Object get() {
      if (index == 0)
         throw new DataStoreException
                              ("not exist");
      Object value = data[0];
      for ( [      c      ] )
         data[i] = data[i + 1];
      data[--index] = null;
      return value;
   }
}
```

〔プログラム 3〕

```
public class LifoStore extends Store {
   public Object get() {
      if (index == 0)
         throw new DataStoreException
                              ("not exist");
      Object value = data[--index];
      data[index] = null;
      return value;
   }
}
```

〔プログラム 4〕

```
public class StoreTest {
   public static void main(String[] args) {
      FifoStore fifo = new FifoStore();
      LifoStore lifo = new LifoStore();
      for (int i = 0; i < args.length; i++) {
         fifo.put(args[i]);
         lifo.put(args[i]);
      }
      printData(fifo);
      printData(lifo);
   }
   private static void printData
               ( [      d      ] store) {
      int size = store.size();
      for (int i = 0; i < size; i++)
         System.out.print(i + ": " + store.get()
                                    + " ");
      System.out.println();
   }
}
```

〔プログラム 5〕

```
public class DataStoreException extends
                          RuntimeException {
   public DataStoreException(String message) {
      super(message);
   }
}
```

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　index < data.length

イ　index <= data.length

ウ　index > data.length

エ　index >= data.length

オ　index + 1 >= data.length

b に関する解答群

ア　data.length

イ　data.length - 1

ウ　data.length - index

エ　index

オ　index + 1

c に関する解答群

ア　int i = 0; i < index; i++

イ　int i = 0; i < index - 1; i++

ウ　int i = 0; i < index - 2; i++

エ　int i = 1; i < index; i++

オ　int i = 1; i < index - 1; i++

d に関する解答群

ア　FifoStore　　イ　LifoStore

ウ　Object　　エ　Store

オ　StoreTest

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 1，2 に答えよ。

〔プログラムの説明〕

プログラムは，A 社のコールセンタのシミュレータである。 A 社はこのシミュレータを，利用者がコールセンタに電話をかけてからオペレータとつながるまでの待ち時間の推定に利用する。コールセンタへの一つの電話呼出しに対しては，1 人のオペレータだけが応答することができる。

このシミュレータでは，空きオペレータがいる限りは利用者を待たせることはないとする。例えば，オペレータが 2 人いるとき，コールセンタに利用者からの電話が 60 秒間隔で 6 回かかってきて，それぞれの通話時間が，130 秒，100 秒，150 秒， 90 秒，110 秒，140 秒だったとすると，利用者の待ち時間は 3 番目が 10 秒，5 番目が 30 秒になり，その他は 0 秒である（図 1 参照）。

なお，このシミュレータでは，実世界の 1 秒を 0.1 秒でシミュレートする。

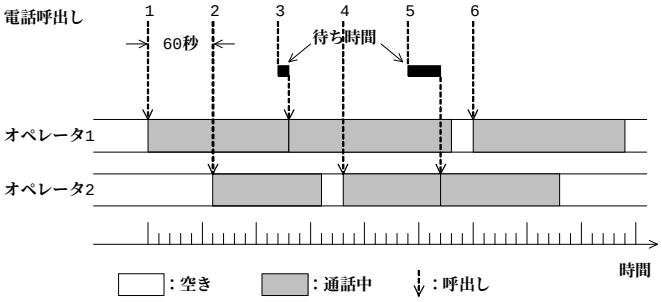

図１　利用者の待ち時間

プログラムは次のクラスから構成される。

(1) CallCenter
　コールセンタを表すクラスである。コンストラクタの引数にオペレータ数，通話時間の配列，呼出しを発生させる間隔を指定して，シミュレーションを実行する。次のメソッドと内部クラスをもつ。

public static void main(String[] args)
　シミュレータをテスト用に起動するメソッドである。

Call answer()
　オペレータが，利用者からの呼出しに応答するために呼ぶメソッドである。オペレータに割り当てる呼出し（ Call オブジェクト）を返す。このメソッドは，オペレータに割り当てるべき呼出しが発生するまで，オペレータのスレッドを待たせる。ただし，このクラスが生成するすべての呼出しの割当てが完了し，割り当てるべき呼出しがなくなったときは，null を返す。

Operator
　オペレータをシミュレートするスレッドクラスである。各オペレータの処理は個別のスレッドで実行される。コールセンタにかかってきた電話に応答し，通話する。CallCenter が生成するすべての呼出しの割当てが完了するまで，処理を繰り返す。

(2) Call
　利用者からの呼出しを表すクラスである。コンストラクタの引数には通話時間を指定する。次のメソッドをもつ。

public void talk()
　オペレータが利用者との通話中の状態をシミュレートするメソッドである。このメソッドは，オペレータに割り当てられた利用者の待ち時間を秒単位（ 1秒未満は四捨五入）で表示した後，通話時間として指定された時間だけ，オペレータのスレッドを停止させる。

プログラムが図 1の例をシミュレートした場合の実行結果を，図 2 に示す。
なお，図中の (s) は（秒）を表す。

```
0(s)
0(s)
10(s)
0(s)
30(s)
0(s)
```

図２　実行結果

プログラム中の java.util.Vector は可変長配列を表すクラスであり，次のメソッドをもつ。

public boolean add(Object obj)
　配列の最後尾に要素 obj を追加する。

public Object remove(int index)
　index で指定された位置の要素を配列から削除し，削除した要素を返す。先頭の要素の位置は 0 である。index より後方の要素は前方に詰められる。

public boolean isEmpty()
　要素がなければ true を，あれば false を返す。

〔プログラム 1〕

```
import java.util.Vector;

public class CallCenter {
   private final Vector waitingList = new Vector();
   private boolean running; // 呼出し生成中は true

   public static void main (String[] args) {
      int op = 2; // オペレータ数
      // 通話時間(秒)
      long[] duration = {130, 100, 150, 90, 110, 140};
      long interval = 60; // 呼出しを発生させる間隔(秒)
      new CallCenter(op, duration, interval);
   }

   public CallCenter(int op, long[] duration,
                                    long interval) {
```

```
    running = true;
    // オペレータのスレッドを生成し，開始する。
    for (int i = 0; i < op; i++) new Operator().
                                        start();
    long nextCallTime =
                    System.currentTimeMillis();
    for (int i = 0; i < duration.length; i++) {
      // 呼出しを一つ生成し，リストに追加する。
      synchronized (waitingList) {
        waitingList.add(new Call(duration[i]));
        waitingList.notify();
      }

      // 次の呼出しを生成するまで待つ。
      nextCallTime += interval * 100;
                                  // 10 倍の速さで動作
      long sleeping = [    a    ] ;
      try {
        if (sleeping > 0) Thread.sleep(sleeping);
      } catch (InterruptedException ie) {}
    }
    // すべてのオペレータのスレッドを終了させる。
    running = false;
    [    b    ] {
      waitingList.notifyAll();
    }
  }

  Call answer() {
    synchronized (waitingList) {
      while (waitingList.isEmpty()
                              [    c    ] ) {
        try {
          [    d    ] ;
        } catch (InterruptedException ie) {}
      }
      if (waitingList.isEmpty()) return null;
      return (Call)waitingList.remove(0);
    }
  }

  class Operator extends Thread {
    public void run() {
      Call call;
      [    e    ] call.talk();
    }
  }
}
```

〔プログラム 2〕

```
public class Call {
  private final long start, duration;
  public Call(long duration) {
    this.duration = duration;
    start = System.currentTimeMillis();
  }
  public void talk() {
    long elapsed = System.currentTimeMillis()
                                        - start;
    // 経過時間を四捨五入して表示する。
    System.out.println( [    f    ]
                                    + "(s)");
    try {
      Thread.sleep(duration * 100);
                              // 10 倍の速さで動作
    } catch (InterruptedException ie) {}
  }
}
```

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。

a に関する解答群

ア　nextCallTime

イ　nextCallTime - duration[i] * 100

ウ　nextCallTime
　　- System.currentTimeMillis()

エ　System.currentTimeMillis()
　　- nextCallTime

b に関する解答群

ア　for (int i = 0; i < op; i++)

イ　if (waitingList != null)

ウ　synchronized (this)

エ　synchronized (waitingList)

c に関する解答群

ア　&& !running　　イ　&& running

ウ　== !running　　エ　|| running

d に関する解答群

ア　notify()

イ　wait()

ウ　waitingList.notify()

エ　waitingList.wait()

e に関する解答群

ア　if ((call = answer()) != null)

イ　if (answer() != null)

ウ　while ((call = answer()) != null)

エ　while (answer() != null)

f に関する解答群

ア (elapsed + 50) / 100

イ (elapsed + 500) / 100

ウ (elapsed - 50) / 100

エ (elapsed - 500) / 100

設問2　CallCenter のインスタンスを，次に示すようにして生成したとき，インスタンスが生成されてからシミュレータの時間で 80 秒後（実時間で 8 秒後）に通話中の状態にあるオペレータの人数を，解答群の中から選べ。

```
new CallCenter(3, new long[]
                {70, 90,100, 110}, 30);
```

解答群

ア 0　　イ 1　　ウ 2　　エ 3

平成１６年度　春期　ＦＥ　午後解答　Java

問8

設問
a－ウ　　b－ア　　c－ウ　　d－エ

問 12

設問
a－カ　　b－エ　　c－ア
d－エ　　e－エ　　f－カ

平成１６年度　秋期　ＦＥ　午後解答　Java

問8

設問
a－エ　　b－エ　　c－イ　　d－エ

問 12

設問1
a－ウ　　b－エ　　c－イ
d－エ　　e－ウ　　f－ア

設問2
ウ