平成１５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〕

図形の面積を計算し，その結果を出力するプログラムである。図形は，三角形，長方形及び正方形のいずれかであり，プログラムでは，次の属性をもつ図形オブジェクトとして定める。

三角形：３辺の長さ
長方形：縦と横の２辺の長さ
正方形：１辺の長さ

このプログラムは，次の五つのクラスで構成される。

AreaTest
メソッド main をもつクラスである。このクラスは，次の処理を行う。
(1)　三角形，長方形及び正方形のオブジェクトを生成し，配列 figures に設定する。
(2)　各図形の面積を求めて，結果を出力する。ここで，各図形には，正しい図形となる数値が与えられているものとする。

Figure
図形の抽象クラスである。このクラスは，面積を計算してその結果を返す抽象メソッド getArea を宣言している。

Triangle
三角形のクラスである。このクラスは，属性を文字列にして返すメソッド toString と三角形の面積をヘロンの公式によって計算してその結果を返すメソッド getArea を定義している。

Rectangle
長方形のクラスである。このクラスは，属性を文字列にして返すメソッド toString と長方形の面積を計算してその結果を返すメソッド getArea を定義している。

Square
正方形のクラスである。このクラスは，属性を文字列にして返すメソッド toString を定義している。

プログラム１の実行結果を図に示す。

```
Triangle : sides = 2.0, 3.0, 3.0 : area
                 = 2.8284271247461903
Rectangle : height = 5.0, width = 8.0 : area
                  = 40.0
Square : width = 5.0 : area = 25.0
```

図　実行結果

〔プログラム 1〕

```
public class AreaTest {
   public static void main(String args[]) {
      Figure[] figures = {
         new Triangle(2, 3, 3),
         new Rectangle(5, 8),
         new Square(5)};
      for (int i = 0; i < figures.length; i++) {
         System.out.println(figures[i] +
             "area = " + figures[i].getArea());
      }
   }
}
```

〔プログラム 2〕

```
public abstract class Figure {
  public abstract double getArea();
}
```

〔プログラム 3〕

```
public class Triangle extends [ a ] {
  double la;
  double lb;
  double lc;
  public Triangle(double la, double lb, double lc) {
    this.la = la;
    this.lb = lb;
    this.lc = lc;
  }
  public String toString() {
    return "Triangle : sides = " + la + ",
                     " + lb + ", " + lc + " : ";
  }
  public double getArea() {
    double s = (la + lb + lc) / 2.0;
    return Math.sqrt(s * (s - la) * (s - lb)
                                  * (s - lc));
  }
}
```

〔プログラム 4〕

```
public class Rectangle extends [ b ] {
  double height;
  double width;
  public Rectangle(double height, double width) {
    this.height = height;
    this.width = width;
  }
  public String toString() {
    return "Rectangle : height = " + height +
                  ", width = " + width + " : ";
  }
  public double getArea() {
    return [ c ] ;
  }
}
```

〔プログラム 5〕

```
public class Square extends [ d ] {
  public Square(double width) {
    [ e ] ;
```

```
  }
  public String toString() {
    return "Square : width = " + width + " : ";
  }
}
```

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。解答は，重複して選んでもよい。

a，b，d に関する解答群

ア　abstract　　イ　Figure　　ウ　getArea
エ　Rectangle　　オ　Square　　カ　super

c に関する解答群

ア　height　　イ　height * height
ウ　height * width　　エ　width
オ　width * width

e に関する解答群

ア　super(height)
イ　super(height, height)
ウ　super(width)
エ　super(width, height)
オ　super(width, width)
カ　this.height = height
キ　this.height = width
ク　this.width = height
ケ　this.width = width

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〕

ある学校の図書館には自習用の席が 30 あり，次の規則で運用されている。

運用規則

(1)　生徒が席を使用するときは受付で申請し，割り当てられた席を使用する。使用終了時には，受付に届ける。

(2)　空席がないときに使用希望者がある場合は，１時間を超えて最も長く使用している者が使用希望者のために席を空けなければならない。空席がなく，1 時間を超えて使用している者がいない場合，使用希望者は席を使用できない。

(3)　１人で同時に複数の席を使用することはできない。

この運用規則に従った自習席管理を支援するためのプログラムを，次のように設計した。

(1)　図に示す双方向リストを実装するためのクラス ListElement を定義する。双方向リストは環状にし，要素の挿入・削除処理で終端を特別扱いしないようにする。 ListElement は，リストのヘッド（先頭）と要素の両方を表現する。 ListElement では，次のメソッドを実装する。

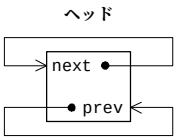
nextElement
　この ListElement インスタンスの次の要素を返す。

previousElement
　この ListElement インスタンスの前の要素を返す。

insertBefore
　この ListElement インスタンスを，引数で指定された要素の前に挿入する。

remove
　この ListElement インスタンスをリストから削除する。

空リスト



要素のあるリスト

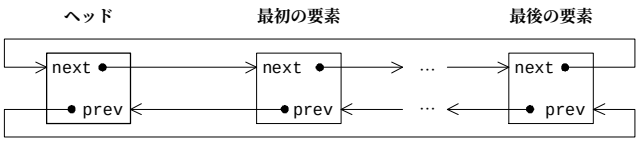


図　空のときと要素があるときの双方向リストの状態

(2) 席は，ListElement を拡張したクラス Seat で表す。各席に座席番号が割り振られる。

(3) クラス SeatManager で，運用規則を実現するためのメソッドを実装する。

(4) 空席，使用中の席のリストを，それぞれ freeSeats，occupiedSeats とする。 Seat のインスタンスが各リストの要素となる。

(5) 使用中の席は，occupiedSeats に登録する。リストは使用時間が短いものから長いものへと並ぶように管理する。

(6) SeatManager では，次の public メソッドを実装する。

checkin
　引数で指定された使用希望者が既に席を使用中でないことを確認する。 freeSeats（空席リスト）に空きがあればその席を，なければ 1時間を超えて使われている席を使用希望者に割り当てる。割り当てられた席を

occupiedSeats（使用中リスト）の最初の要素として登録し，その Seat インスタンスを返す。使用できる席がなければ null を返す。

checkout
引数で指定された使用者が使っていた席を occupiedSeats から削除して freeSeats へ戻し，true を返す。指定された使用者が occupiedSeats に見つからないときは false を返す。

最初に，クラス ListElement 及びクラス Seat を実装して単体テストを行い，正しく動作することを確認した。次に，クラス SeatManager を実装してテストしたところ，1 人で同時に複数の席が使用できるという不具合が見つかった。それ以外は，正常に動作することを確認した。

〔プログラム 1〕

```
public class ListElement {
   private ListElement prev, next;
   public ListElement() {
      prev = next = this;
   }
   public ListElement nextElement() { return next; }
   public ListElement previousElement()
                                  { return prev; }
   public void insertBefore(ListElement element) {
      next = element;
      prev = element.prev;
      next.prev = prev.next = [ a ] ;
   }
   public void remove() {
      [ b ] = next;
      [ c ] = prev;
      prev = next = this;
   }
}
```

〔プログラム 2〕

```
public class Seat extends ListElement {
   private String userID;     // 使用者
   private long checkinTime;  // 使用開始時刻
   private int seatNumber;    // 座席番号

   public Seat(int seatNumber) {
      this.seatNumber = seatNumber;
   }
   public int getSeatNumber() {
      return seatNumber;
   }
   public String getUserID() {
      return userID;
   }
   public void setUserID(String userID) {
      this.userID = userID;
   }
   public boolean isUsedBy(String userID) {
      return this.userID.equals(userID);
   }
   public long getCheckinTime() {
      return checkinTime;
   }
   public void setCheckinTime(long time) {
      checkinTime = time;
   }
}
```

〔プログラム 3〕

```
public class SeatManager {
   private static final int NSEATS = 30; // 席数
   // 最大使用時間〔ミリ秒〕
   private static final int MAXTIME =
                                 60 * 60 * 1000;
   // 空席リスト
   private ListElement freeSeats =
                                new ListElement();
   // 使用中リスト
   private ListElement occupiedSeats =
                                new ListElement();

   public SeatManager() {
      for (int i = 1; i <= NSEATS; i++) {
         Seat seat = new Seat(i);
         seat.insertBefore(freeSeats);
      }
   }

   // 空席リストに空席があればその Seat インスタンスを空席
   リストから// 削除し，そのインスタンスを返す。空きがなければ null を返す。
   private Seat getFreeSeat() {
      ListElement le = freeSeats.nextElement();
      if (le != freeSeats) {
         le.remove();
         return (Seat) le;
      }
      return null;
   }

   // 使用中リストを調べ，最大使用時間を超えて席を使用している// 使用者がいれば，その旨出力し，checkout を呼ぶ。
   private void vacateExpiredSeat(long time) {
      ListElement le = [ d ];
      if (le != occupiedSeats) {
         Seat seat = (Seat) le;
         if ((seat.getCheckinTime() + MAXTIME)
                                      < time) {
            System.out.println("Seat#" +
                     seat.getSeatNumber() + " " +
                     seat.getUserID() +
                     " must check out.");
            checkout(seat.getUserID());
         }
      }
   }

   // 使用中リストから指定された使用者が使っている席を探し，見つかれば// その席を，見つからなければ null を返す。
   private Seat findUser(String userID) {
      ListElement le = [ e ];
      while (le != occupiedSeats) {
         Seat seat = (Seat) le;
```

```
            if (seat.isUsedBy(userID)) {
                return seat;
            }
            le = le.nextElement();
        }
        return null;
    }

    public Seat checkin(String userID) {
        long now = System.currentTimeMillis();
        Seat seat = getFreeSeat();
        if (seat == null) {
            vacateExpiredSeat(now);
            seat = getFreeSeat();
        }
        if (seat != null) {
            seat.setCheckinTime(now);
            seat.setUserID(userID);
            seat.insertBefore(occupiedSeats.
                              nextElement());
        }
        return seat;
    }

    public boolean checkout(String userID) {
        Seat seat = findUser(userID);
        if (seat != null) {
            seat.remove();
            seat.setUserID(null);
            seat.insertBefore(freeSeats);
            return true;
        }
        return false;
    }
}
```

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 ～ c に関する解答群

ア　element　　イ　element.next
ウ　element.prev　　エ　new ListElement()
オ　next　　カ　next.prev
キ　null　　ク　prev
ケ　prev.next　　コ　this

d，e に関する解答群

ア　freeSeats
イ　freeSeats.nextElement()
ウ　freeSeats.previousElement()
エ　occupiedSeats
オ　occupiedSeats.nextElement()
カ　occupiedSeats.previousElement()

設問２　１人で同時に複数の席を使用できないようにプログラム３を修正したい。修正方法として適切なものを，解答群の中から選べ。ただし，プログラム中の [　a　] ～ [　e　] にはすべて正しい答えが入っているものとする。

解答群

ア　メソッド checkin で，席の割当て処理をする前に occupiedSeats を調べ，もしこの使用希望者が既に他の席を使用中であればエラーとし，席の割当て処理を行わない。

イ　メソッド checkout で，使用を終了した席が occupiedSeats から正しく削除されていないので，occupiedSeats から使用を終了した席を正しく削除する。

ウ　メソッド vacateExpiredSeat で，最大使用時間（MAXTIME）を超えて席を使用している者を強制的にチェックアウトさせるために，メソッド checkout を呼び出している。その直後に，その使用者の使っていた席を occupiedSeats から削除する処理を追加する。

平成１５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〕

プログラムは，バイナリデータのバイト列をあるアルゴリズムに従って変換し，その結果を文字の列として取得するためのクラス Encoder とそのテスト用クラスからなる。変換アルゴリズムは次のとおりである。

(1) バイト列の先頭から順に 6 ビットずつ取り出し，変換表の対応する文字に変換する。ただし，バイト数が 3 の倍数でないときは，ビット数が 6 の倍数になるように，最後に 4 ビット又は 2 ビットの 0 があるものとする。

(2) 変換後の文字数が，与えられたバイト数の 4／3 倍以上で最も小さい 4 の倍数になるように，文字の列の最後に必要な個数だけ“＝”を付加する。

(3) 変換例を図１ に示す。

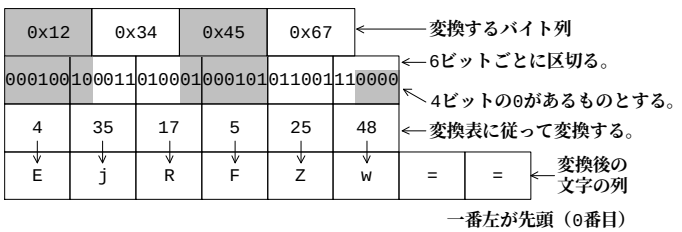


図１　変換例

(4) クラス Encoder は，next と hasNext の二つのメソッドをもつ。

ext は，変換アルゴリズムに従って変換された文字を 1 文字ずつ順に返すメソッドである。返すべき文字がなくなった後に呼ばれると，java.util.NoSuchElementException を投げる。

hasNext は，次の next 呼出しの際に，例外が発生せずに文字を返すことができるとき真となる。

(5) 変数 CHARS は，6 ビットの数値を 1 文字に変換するための表で，配列の大きさは 64 である。

変数 n は，次に next が呼ばれたときに返すべき文字が何番目の文字であるかを保持する（先頭の文字は 0 番目とする）。変数 bin は，変換するバイナリデータの列を保持する。

(6) 図 1 のバイト列を用いたテスト用クラス EncoderTest の実行結果を図2 に示す。

```
EjRFZw==
```

図2　実行結果

〔プログラム〕

```
public class Encoder {
   static final char[] CHARS = ("ABCDEFGHIJKLMN
    OPQRSTUVWXYZ" + "abcdefghijklmnopqrstuvwx
    yz0123456789+/").toCharArray();
   private int n = 0;
   private byte[] bin;
   public Encoder(byte[] bin) {
      if (bin == null) this.bin = new byte[0];
      else this.bin = bin;
   }
   public boolean hasNext() {
      return n < [          a          ];
   }
   public char next() {
      char letter;
      int pos = (int)(n * 0.75);
      // 変換対象の 6 ビットのデータが含まれる 2 バイト
      のデータを取得
      if (pos < bin.length) {
         int cell = bin[pos++] << 8;
         if (pos < bin.length) cell +=
                              bin[pos] & 255;
         // 変換すべき 6 ビットを抽出し，対応する 1 文
         字に変換
         letter = CHARS[(cell >> (n + 3)
                          % 4 * 2 + 4) & 63];
      } else {
         if ([       b       ])
            throw new java.util.
                     NoSuchElementException();
         else letter = [       c       ];
      }
      n++;
      return letter;
   }
}

class EncoderTest {
   public static void main(String[] args) {
      byte[] bin = {0x12, 0x34, 0x45, 0x67};
      Encoder encoder = new Encoder(bin);
      while (encoder.hasNext()) {
         System.out.print(encoder.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 に関する解答群

ア　(bin.length + 2) * 4 / 3

イ　(bin.length + 2) / 3 * 4

ウ　bin.length * 4 / 3

エ　bin.length / 3 * 4

b に関する解答群

ア　!hasNext()　　イ　hasNext()

ウ　n < bin.length　　エ　n >= bin.length()

c に関する解答群

ア　''　　イ　'='

ウ　CHARS[n]　　エ　next()

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〕

クラス Matrix は，要素の値が実数である行列を表すクラスであり，主な行列演算のためのメソッドを定義する。クラス MatrixTest は，クラス Matrix のオブジェクトをテストするメソッド main を定義するクラスである。

クラス Matrix の仕様は，次のとおりである。

(1) クラス Matrix のコンストラクタは，行列の要素の値を含む double[][] 型の引数をもつ。行及び列は null ではなく，各行の要素の個数は同一であるとする。空行列は考えない。

(2) メソッド toString によって返される行列オブジェクトの文字列表現は，次のとおりである。

[[1 行 1 列 1 行 2 列 … 1 行 N 列]
　　… [M 行 1 列 M 行 2 列 … M 行 N 列]]

MatrixTest 例1

行列 $\begin{pmatrix}1.0 & 2.0\\ 3.0 & 4.0\end{pmatrix}$ の文字列表現は，[[1.0 2.0] [3.0 4.0]] となる。

(3) メソッド add に，行列オブジェクトの引数を渡すと，オブジェクト this と引数のオブジェクトとの和の行列を返す。行列の和は，二つの行列の行数と列数がそれぞれ等しい場合にだけ定義される。

MatrixTest 例2

行列 $\begin{pmatrix}1.0 & 2.0 & 3.0\\ 4.0 & 5.0 & 6.0\end{pmatrix}$ と行列 $\begin{pmatrix}1.2 & 1.3 & 1.4\\ 1.5 & 1.6 & 1.7\end{pmatrix}$ との和は，$\begin{pmatrix}2.2 & 3.3 & 4.4\\ 5.5 & 6.6 & 7.7\end{pmatrix}$ になる。

オブジェクト this との和が定義されない行列オブジェクトが引数に渡された場合には，例外 MatrixArithmeticException を投げる。

(4) メソッド multiply に，実数の引数を渡すと，オブジェクト this の要素ごとにその引数を乗算した結果の行列を返す。

MatrixTest 例3

行列 $\begin{pmatrix}1.0 & 2.0\\ 3.0 & 4.0\end{pmatrix}$ と実数 2.0 との積は，$\begin{pmatrix}2.0 & 4.0\\ 6.0 & 8.0\end{pmatrix}$ になる。

(5) メソッド multiply に，行列オブジェクトの引数を渡すと，オブジェクト this と引数のオブジェクトとの積の行列を返す。k 行 l 列の行列 A と，m 行 n 列の行列 B との積 AB は，行列 A の列数と行列 B の行数が等しい，すなわち $l = m$ であるときにだけ定義される。$AB = C$ とするとき，積の行列 C は k 行 n 列になる。このとき，行列 C の第 i 行第 j 列の要素の値は，行列 A の i 行 x 列（ここで，$x = 1, \cdots, l$）の値と行列 B の x 行 j 列の値とを掛けて得られる l 個の実数の和である。

MatrixTest 例4

行列 $\begin{pmatrix}1.0 & 2.0 & 3.0\\ 4.0 & 5.0 & 6.0\end{pmatrix}$ と行列 $\begin{pmatrix}1.0 & 0.0\\ 0.0 & 1.0\\ 1.0 & 0.0\end{pmatrix}$ との積は，$\begin{pmatrix}4.0 & 2.0\\ 10.0 & 5.0\end{pmatrix}$ になる。オブジェクト this との積が定義されない行列オブジェクトが引数に渡された場合には，例外 MatrixArithmeticException を投げる。

〔プログラム〕

```
class MatrixArithmeticException extends
                                    Exception { }

public class Matrix {
  private double[][] values;

  public Matrix(double[][] initVal) {
    values = new double[initVal.length]
                       [initVal[0].length];
    // 配列 initVal の内容を配列 values にコピーする
    for (int i = 0; i < initVal.length; i++) {
      System.arraycopy(initVal[i], 0,
            values[i],0,initVal[i].length);
    }
  }

  public Matrix add(Matrix m) throws
                    MatrixArithmeticException {
    if ( [ a ] )
      throw new MatrixArithmeticException();
    double[][] result =
      new double[values.length]
                        [values[0].length];
    for (int i = 0; i < values.length; i++) {
      for (int j = 0; j < values[0].length;
                                        j++) {
        result[i][j] = values[i][j] +
                            m.values[i][j];
      }
    }
    [ b ];
  }

  public Matrix multiply(Matrix m)
    throws MatrixArithmeticException {
    if ( [ c ] )
      throw new MatrixArithmeticException();
    double[][] result =
      new double[values.length]
                       [m.values[0].length];
    for (int i = 0; i < values.length; i++) {
      for (int j = 0; j < m.values[0].length;
                                        j++) {
        for (int k = 0; k < m.values.length;
                                        k++) {
          result[i][j] += values[i][k] *
                            m.values[k][j];
        }
      }
    }
    [ b ];
  }

  public Matrix multiply(double d) {
    double[][] result =
      new double[values.length]
                        [values[0].length];
    for (int i = 0; i < values.length;
                                       i++) {
      for (int j = 0; j < values[0].length;
                                        j++) {
        result[i][j] = values[i][j] * d;
      }
    }
    [ b ];
  }

  public String toString() {
    StringBuffer rep = new StringBuffer("[";
```

```
    for (int i = 0; i < values.length; i++) {
      rep.append("[");
      for (int j = 0; j < values[0].length;
                                        j++) {
        rep.append(((j > 0) ? " " : "") +
                              values[i][j]);
      }
      rep.append("]");
    }
    rep.append(")");
    [          d          ];
  }
}

class MatrixTest {
  public static void main(String[] args)
   throws MatrixArithmeticException {
   // 例 1
   Matrix matrix0 =
    new Matrix(new double[][]{{1.0, 2.0},
                              {3.0,4.0}});
   System.out.println(matrix0);

   // 例 2
   Matrix matrix1 = new Matrix(new double[][]
          {{1.0, 2.0, 3.0}, {4.0, 5.0, 6.0}});
   Matrix matrix2 = new Matrix(new double[][]
         {{1.2, 1.3, 1.4}, {1.5, 1.6, 1.7}});
  System.out.println(matrix1.add(matrix2));

   // 例 3
   double d = 2.0;
   System.out.println(matrix0.multiply(d));

   // 例 4
   Matrix matrix3 = new Matrix(new double[][]
         {{1.0, 2.0, 3.0}, {4.0, 5.0, 6.0}});
   Matrix matrix4 = new Matrix(new double[][]
         {{1.0, 0.0}, {0.0, 1.0}, {1.0, 0.0}});
   System.out.println(matrix3.multiply
                                  (matrix4));
   }
}
```

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，c に関する解答群

ア　values.length == m.values[0].length

イ　values.length != m.values[0].length

ウ　values[0].length == m.values.length

エ　values[0].length != m.values.length

オ　values.length == m.values.length &&
　　values[0].length == m.values[0].length

カ　values.length != m.values.length ||
　　values[0].length != m.values[0].length

キ　values[0].length == m.values.length &&
　　values.length == m.values[0].length

ク　values[0].length != m.values.length ||
　　values.length != m.values[0].length

b，d に関する解答群

ア　return (Matrix)null

イ　return new Matrix()

ウ　return new Matrix(result)

エ　return new String(rep)

オ　return rep

カ　return rep.getBytes()

キ　return result

ク　System.out.println(rep)

平成１５年度　春期　ＦＥ　午後解答　Java

問８

設問
a－イ　　b－イ　　c－ウ　　d－エ　　e－オ

問 12

設問１
a－コ　　b－ケ　　c－カ　　d－カ　　e－オ

設問２
ア

平成１５年度　秋期　ＦＥ　午後解答　Java

問８

設問
a－イ　　b－ア　　c－イ

問 12

設問
a－カ　　b－ウ　　c－エ　　d－エ