問 11　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。
（Java プログラムで使用する API の説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

あみだくじの作成と結果の表示を行うプログラムである。

あみだくじとは，複数の平行に並ぶ縦線と，2 本の縦線だけに水平に接続する複数の横線から成るくじである。横線は縦線の上端及び下端には接続せず，縦線の同一箇所に複数の横線が接続されることもない。くじをたどる手順は次のとおりである。

(1)　1 本の縦線を選択し，上端から下方向にたどり始める。
(2)　途中に，接続する横線があるときは必ず曲がり，もう一方の縦線までたどる。
(3)　縦線に到達したら，また下方向にたどる。
(4)　下端に到達するまで(2)と(3)を繰り返す。
(5)　下端に到達したときの縦線の位置が，くじを引いた結果である。

あみだくじの例を図 1 に示す。図中の太線は，左端の縦線を選択したときにたどった経路である。

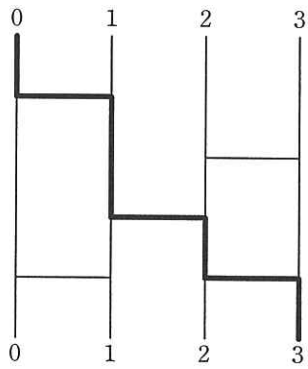

図 1　あみだくじの例

クラス GhostLeg は，あみだくじを作成し，作成したあみだくじをたどるプログラムである。

(1)　縦線を左からの順番で保持する。左端を 0 番目とする。
(2)　縦線の長さを 1.0 とし，上端の縦軸座標を 1.0，下端の縦軸座標を 0.0 とする。
(3)　横線の両端の縦軸座標は等しく，0.0 よりも大きく 1.0 未満である。

(4) 縦線の本数を引数とするコンストラクタをもつ。

(5) 次のメソッドをもつ。

① `void addHorizontalLine(int x1, int x2, double y)`

横線を追加するメソッドである。左から x1 番目の縦線と x2 番目の縦線を縦軸座標 y で接続する横線を追加する。ただし，x1 と x2 が等しいか，2 本の縦線のいずれか又は両方に，既に縦軸座標 y で接続する横線があるときには追加しない。

あみだくじは，クラス GhostLeg のインスタンスを生成した後，このメソッドを任意の回数だけ呼び出し，横線を追加することによって完成する。

② `int trace(int x)`

くじを引いた結果を返すメソッドである。左から x 番目の縦線を選択し，あみだくじをたどった結果を返す。

クラス GhostLegTester はテスト用のプログラムである。このプログラムは，図 1 に示したあみだくじを作り，左から順にくじを引き，結果を表示する。実行結果を，図 2 に示す。

```
0 -> 3
1 -> 1
2 -> 2
3 -> 0
```

図 2　クラス GhostLegTester の実行結果

〔プログラム 1〕

```
import java.util.ArrayList;
import java.util.List;
import java.util.SortedMap;
import java.util.TreeMap;

public class GhostLeg {
   private List<VerticalLine> verticalLines;

   public [ a ](int n) {
      verticalLines = new ArrayList<VerticalLine>(n);
      for (int i = 0; i < n; i++) {
         verticalLines.add([ b ]);
      }
   }
```

```
    public void addHorizontalLine(int x1, int x2, double y) {
        // 設問3(α)
        VerticalLine v1 = verticalLines.get(x1);
        // 設問3(β)
        VerticalLine v2 = verticalLines.get(x2);
        if (x1 != x2 && !v1.hasHorizontalLineAt(y) &&
            !v2.hasHorizontalLineAt(y)) {
        // 設問3(γ)
            v1.putHorizontalLine(y, v2);
            v2.putHorizontalLine(y, v1);
        }
        // 設問3(δ)
    }

    public int trace(int x) {
        double y = 1.0;
        VerticalLine v = verticalLines.get(x);
        while ((y = v.getNextY(y)) > [    c    ]) {
            v = [    d    ];
        }
        return verticalLines.indexOf(v);
    }

    private static class VerticalLine {
        SortedMap<Double, VerticalLine> horizontalLines =
            new TreeMap<Double, VerticalLine>();

        VerticalLine() {
            horizontalLines.put(0.0, null);
        }

        boolean hasHorizontalLineAt(double y) {
            return horizontalLines.containsKey(y);
        }

        void putHorizontalLine(double y, VerticalLine opposite) {
            horizontalLines.put(y, opposite);
        }

        double getNextY(double y) {
            // マップ horizontalLines が保持するキーの中で,
            // 引数 y よりも小さいもののうち, 最大のものを返す。
            return horizontalLines.headMap(y).lastKey();
        }

        VerticalLine getOpposite(double y) {
            return horizontalLines.get(y);
        }
    }
}
```

〔プログラム2〕

```
public class GhostLegTester {
   public static void main(String[] args) {
      GhostLeg gh = new GhostLeg(4);
      gh.addHorizontalLine(0, 1, 0.8);
      gh.addHorizontalLine(0, 1, 0.2);
      gh.addHorizontalLine(1, 2, 0.4);
      gh.addHorizontalLine(2, 3, 0.6);
      gh.addHorizontalLine(2, 3, 0.2);
      for (int i = 0; i < 4; i++) {
         System.out.printf("%d -> %d%n", i, gh.trace(i));
      }
   }
}
```

設問1　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

クラス GhostLeg の内部クラス VerticalLine は 1 本の縦線を表すクラスであり，インスタンスは，そのインスタンスが示す縦線に接続している横線を保持している。この内部クラスでは，横線を，［　　］として表現している。

解答群

ア　クラス HorizontalLine のインスタンスの列

イ　その横線が接続する縦軸座標とクラス HorizontalLine のインスタンスのマップ

ウ　その横線が接続する縦軸座標とその横線に接続するもう 1 本の縦線のマップ

エ　その横線に接続する 2 本の縦線のマップ

設問2　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　ghostLeg　　イ　GhostLeg

ウ　void ghostLeg　　エ　void GhostLeg

bに関する解答群

ア　0　　イ　i

ウ　new VerticalLine()　　エ　null

cに関する解答群

ア　-1.0　　イ　0.0　　ウ　x　　エ　y

dに関する解答群

ア　horizontalLines.get(x)　　イ　horizontalLines.get(y)

ウ　v.getOpposite(y)　　エ　verticalLines.get(x)

設問3　次の記述中の［　　］に入れる正しい答えを，解答群の中から選べ。

メソッド addHorizontalLine の呼出しにおいて，引数 x1 又は x2 に相当する縦線がないときには IndexOutOfBoundsException が投げられる。一方，引数 y には追加する横線の縦軸座標を指定するので，その値は 0.0 よりも大きく 1.0 未満でなければならないが，それ以外の値を指定しても例外は投げない。

これを，引数 x1 及び x2 の値によらず，引数 y の値が横線の縦軸座標として取り得る範囲から外れているときに IllegalArgumentException を投げるようにするには，プログラム 1 中のコメント“// 設問 3 (［　e　］)”の 1 行を次の if 文に置き換えればよい。

```
if (［　f　］) {
    throw new IllegalArgumentException();
}
```

eに関する解答群

ア　α　　イ　β　　ウ　γ　　エ　δ

fに関する解答群

ア　y <= 0.0 && y >= 1.0　　イ　y <= 0.0 || y >= 1.0

ウ　y > 0.0 && y < 1.0　　エ　y > 0.0 || y < 1.0