日商プログラミング検定 EXPERT（Java）サンプル問題

知識科目

第 1 問（知識 4 択：20 問）

1. クラス内で定義するフィールドにアクセス指定子を付けなかった場合のふるまいとして、もっともふさわしいものはどれか。次から選びなさい。

① 同じパッケージ内にあるクラスから変数を直接アクセスすることができる。
② あらゆるクラスから変数を直接アクセスすることができる。
③ 他のクラスから変数を直接参照することはできるが、書き換えることはできない。
④ この変数に直接アクセルできるのは、同じクラス内のみに限定されています。

2. 配列が以下のように初期設定されているとき、score[1][2]の値はいくつか。次の中から最も適切なものを選びなさい。

```
double  score[][] = {{116.59 , 96.40 , 212.99} , {61.13 , 48.47 , 109.60}};
```

① 109.6
② 96.4
③ 61.13
④ 48.47

3. 次のプログラム片を実行後の変数 b2 の値はどうなっているか。次のうち最もふさわしいものを選びなさい。

```
int b1 = -1;
int b2 = b1 >>> 2;
```

① b1 は負の数だが b2 は正の数となる。
② b2 の値は-1 となる。
③ b2 の値は b1 の値の 1/4 となる。
④ b2 の値は求められず、エラーになる。

4. 次のプログラム片の動作について、最もふさわしいものを選びなさい。

```
final int xf=10;
xf = 11;
```

① コンパイルエラーとなり、実行することはできない。
② コンパイルはできるが、実行時エラーとなる。
③ プログラムを実行することはでき、実行の結果 xf は 11 となる。
④ プログラムを実行することはでき、実行の結果、xf の値は 10 のままである。

5. 以下のプログラム片において、（A）に入れる「その他」にあたるキーワードはどれか。次の中から最も適切なものを選びなさい。

```
int y = 1;
switch (n) {
      case 2:
            y = y * 10;
      case 1:
            y = y * 10;
      (A):
            y = 0;
}
```

① default
② break
③ continue
④ while

6. 以下のプログラム片の動作の説明として最も適切なものを次から選びなさい。ただし、 メソッド func() の型は boolean 型であり、このメソッドは正しく定義されているものとする。

```
for(int i = 0; i < 20; i++){
      if ( func()){
            break;
      }
}
```

① メソッド func() を繰り返し実行し、戻り値が true となった時、または 20 回繰り返した時、繰り返しを終了する。
② メソッド func()を 20 回繰り返し実行する。
③ メソッド func()の戻り値が false になるまで、繰り返し実行する。
④ メソッド func()を繰り返し実行し、戻り値が false になり、かつ、20 回繰り返した時、繰り返しを終了する。

7. 以下のように宣言されたクラスの説明について、最も適切なものを次から選びなさい。なお、ClassA は正しく定義されているものとする。

```
final class ClassA{
      クラスの定義
}
```

① ClassA を継承することはできない。
② ClassA 内のメソッドをオーバライドすることはできない。
③ ClassA 内のメソッドをオーバーロードすることはできない。
④ ClassA は同じパッケージ内のクラスからしか利用することができない。

8．以下のように定義されたクラスの説明について、誤っているものを次から選びなさい。

```
class ClassA{
   private int x;
   public ClassA(int x){
      this.x = x;
   }
   public int getX(){
      return x;
   }
}
```

① ClassA のメソッドを利用すれば、変数 x の値を随時変更することができる。
② ClassA のメソッドを利用すれば、変数 x の値を随時読み出すことができる。
③ ClassA のインスタンスを生成するとき、引数なしのコンストラクタは使用できない。
④ ClassA の変数 x を利用するには、インスタンスの生成が必須である。

9. インターフェイス InterfaceB が InterfaceA を継承する時の定義は次のうちどれか。最も適切なものを選びなさい。

①

```
public interface InterfaceB extends InterfaceA {

}
```

②

```
public interface InterfaceB implements InterfaceA {

}
```

③

```
public interface InterfaceA implements InterfaceB {

}
```

④

```
public interface InterfaceA extends InterfaceB {

}
```

10. 以下のように ClassA・ClassB が宣言されている時の説明として、誤っているものを次から選びなさい。ただし、両クラスともメソッドは正しく定義されているものとする。

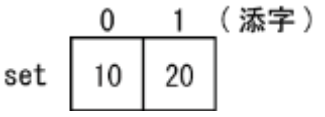
```
public class ClassA {
   private int xa;
      ClassA のメソッド
   class ClassB{
      private int xb;
         ClassB のメソッド
   }
}
```

① ClassA 内のメソッドにおいて　xb = 10;　という文を実行することができる。
② ClassB 内のメソッドにおいて　xa = 10;　という文を実行することができる。
③ ClassA 内のメソッドにおいて、ClassB のインスタンス b を利用して　b.xb = 10;　という文を実行することができる。
④ ClassA・ClassB とは別のクラス内で ClassB のインスタンスを作成して利用することはできない。

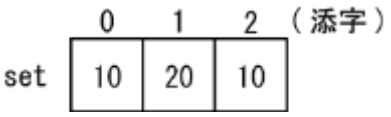
11. 以下のプログラム片を実行した時、可変長配列 set の内容について、最も適切なものを次から選びなさい。

```
java.util.Set<Integer> set = new java.util.HashSet<Integer>();
set.add(10);
set.add(20);
set.add(10);
```

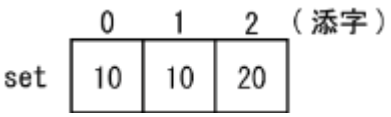
①



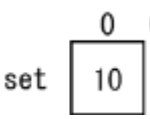
②



③



④



12. try/catch ブロックを用意、または、例外をスローしなければ、コンパイエラーとなるのはどれか。最も適切なものを次から選びなさい。

① ファイルにアクセスするプログラム
② ゼロで除算する可能性があるプログラム
③ 文字列を数値に変換するメソッド　Integer.parseInt()　を利用するプログラム
④ 用意された配列の範囲を超えてアクセスする可能性のあるプログラム

13. 以下のように定義されたクラス ClassA の説明として誤っているものは次のうちどれか。ただしメソッドは正しく記述されているものとする。

```
public class ClassA {
   private int xa;
   public static void foo1(){
        メソッドの記述
   }
   public void foo2(){
        メソッドの記述
   }
}
```

① メソッド foo1()内で ClassA の変数 xa に値を代入することができる。
② メソッド foo1()内で ClassA のメソッド foo2()を呼び出すことはできない。
③ 別に定義したクラス ClassB から ClassA のメソッド foo1()を呼びだす時、ClassA のインスタンスは必要ない。
④ 別に定義したクラス ClassB から ClassA のメソッド foo2()を呼びだす時、ClassA のインスタンスが必要である。

14. 以下のような int 型関数が定義されている時、関数呼出し func(4) による戻り値はいくつか。次の中から選びなさい。

```
int func(int x) {
    if (x == 1)
        return 1;
    else
        return func(x - 1) * x;
}
```

① 24
② 4
③ 1
④ 3

15. 以下のようなプログラム片をコンパイルしたところコンパイルエラーが発生した。エラーとなる文の組み合わせはどれか。最も適切なものを選べ。

```
public interface InterfaceA {・・・①
   private int a;           ・・・②
   public void foo1();      ・・・③
   public void foo2(){}     ・・・④
}
```

① ②と④
② ②と③
③ ①と④
④ ①と②

16. Swing を用いて画面を作成し、JPanel 上にコマンドボタンを配置した。以下は JPanel を実装するプログラム片である。MyJPanel クラスのコンストラクタに記述するボタンがクリックされたイベントを検出するメソッドとして、最もふさわしいものを次から選びなさい。ただし、イベント処理は正しく記述されているものとする。

```
public class MyJPanel extends JPanel implements ActionListener{
   public MyJPanel(){
      JButton b = new JButton("A");
      add(b);
      b.addActionListener(this);
   }

   public void actionPerformed(ActionEvent e) {
      イベント処理
   }

}
```

① b.addActionListener(this)
② addActionListener(b)
③ addActionListener(MyJPanel)
④ b.addActionListener(MyJPanel)

17. 複数のスレッドを同時に動作させている環境で、片方のスレッドが実行中のメソッドについて、その実行終わるまで他方のスレッドを待たせる時にメソッドに指定するキーワードは次のどれか。

① synchronized
② final
③ protected
④ private

18. 以下は、メソッド foo()の定義と、このメソッドを呼び出すプログラム片である。このプログラム片を実行した結果の呼び出し側の配列の状態として、もっともふさわしいものは次のどれか。

呼び出し側プログラム片

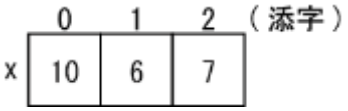
```
int[] x ={5, 6, 7};
int[] y;
y = foo(x);
```

メソッド定義

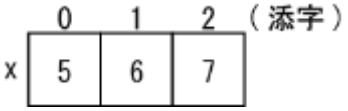
```
private static int[] foo(int[] z){
    z[0] = 10;

    return z;
}
```

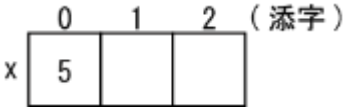
①



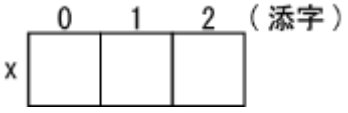
②



③



④



19. クラス ClassA のインスタンス a に対し、ガベージコレクションが可能にならないのは、次のどれか。

①

```
ClassA b;
b = a;
```

②

```
a=null
```

③

```
a = new Object();
```

④

a = new ClassA();

20. クラス ClassA を継承したクラス ClassB のコンストラクタに関して、誤っている説明はどれか。次から選びなさい。

① クラス ClassB のコンストラクタからクラス ClassA のコンストラクタを呼びだすには、ClassA();と記述する。
② クラス ClassB のコンストラクタからクラス ClassA のコンストラクタを呼びだすには、super();と記述する。
③ クラス ClassB では、ClassA で定義しているコンストラクタと同じ引数のコンストラクタをすべて用意しなければならない
④ クラス ClassB のメソッド内でスーパークラスである ClassA のインスタンスを生成することができる。

第 2 問（穴埋め：2 問）

【問題 1】

河川の水位を上流から下流まで 4 地点で毎時間同時に計測しており、時刻ごとに 4 地点のグラフが画像として保存されている。画像のファイル名は時系列に g01.jpg から連番であり、最後は g30.jpg である。画像は java のクラスファイルが保存されているフォルダ内の images フォルダに保存されている。

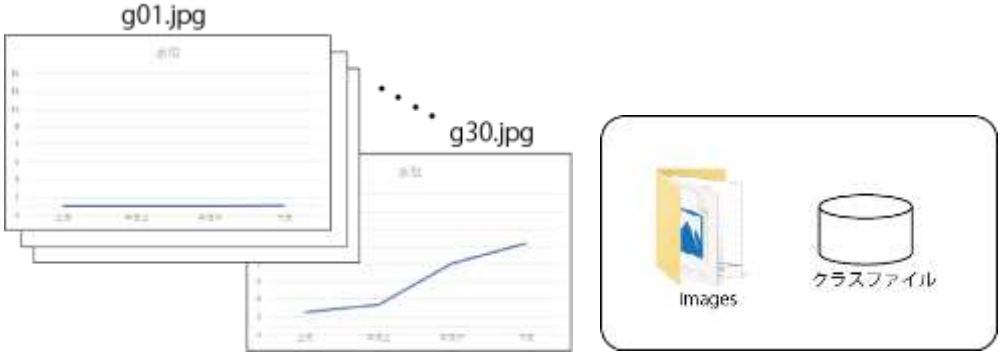


図 1　グラフ画像の保存フォルダ

図 2 に示すユーザフォームを作成し、イメージ領域にグラフを表示、「次へ」ボタンをクリックすると、次の時刻のグラフに切り替える。同様に、「前へ」ボタンをクリックすると、1 つ前の時刻のグラフを表示する。

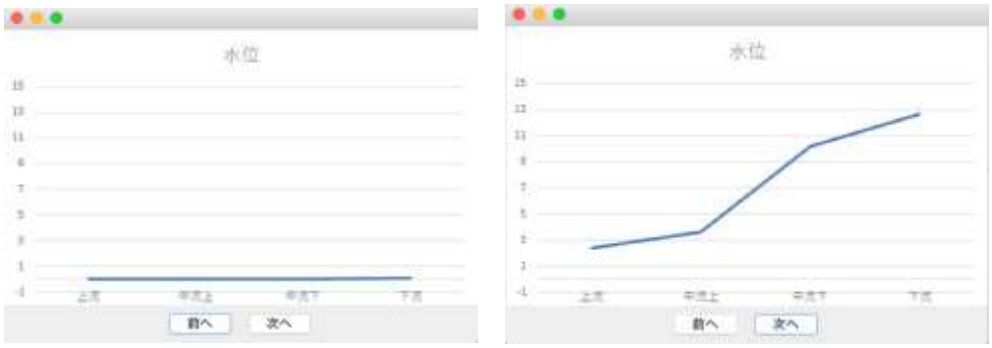


図 2　グラフを表示するユーザフォーム

最初のグラフを表示しているときに「前へ」ボタンが押されても、それ以上戻らず、最後のグラフを表示しているときに「次へ」ボタンが押されても、それ以上進むことはない。

プログラムでは、以下の変数を利用している。

表 1　変数一覧

	型	変数名	説明
static	int	START	最初グラフの画像番号
static	int	END	最後のグラフの画像番号
	int	now	現在表示しているグラフの画像番号

このプログラムは、以下の内部クラスを持つ。

表 2　内部クラス MyDrawPanel extends JPanel

メソッド		
修飾子	型	形式と説明
public	void	paintComponent(Graphics g) パネルに指定された番号のグラフ画像を表示する。

表 3　内部クラス MyButtonPanel extends JPanel

コンストラクタ	
修飾子	形式と説明
public	MyButtonPanel() 「前へ」ボタンと「次へ」ボタンを搭載したパネルを生成する。

表 4　内部クラス ChangeButton extends JButton

フィールド			
修飾子	型	名称	説明
private	int	value	１回のクリック操作における now の変更量。「前へ」ボタンでは-1 が、「次へ」ボタンでは 1 がコンストラクタで指定される。
コンストラクタ			
修飾子	形式と説明		
public	ChangeButton(String cap , int value) cap : ボタンに表示するキャプション value : 「前へ」ボタンでは-1 「次へ」ボタンでは 1 ボタンを生成してイベントリスナーを登録する。		
メソッド			
修飾子	型	形式と説明	
public	void	actionPerformed(ActionEvent e) value を更新し、value の許容範囲チェックを行う。value が許容範囲を逸脱しているときは、許容範囲になるように補正する。MyDrawPanel のインスタンスに対し、再表示を要求する。	

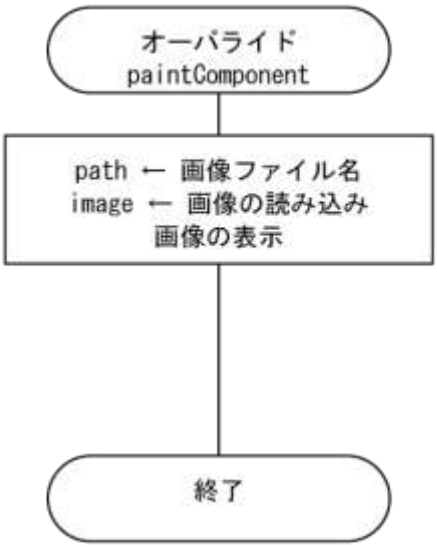
Java では以下のメソッドが利用できる。

クラス	修飾子	型	形式と説明
String	public static	String	format(String format , Object… args) format の書式に従って、文字列を生成する。

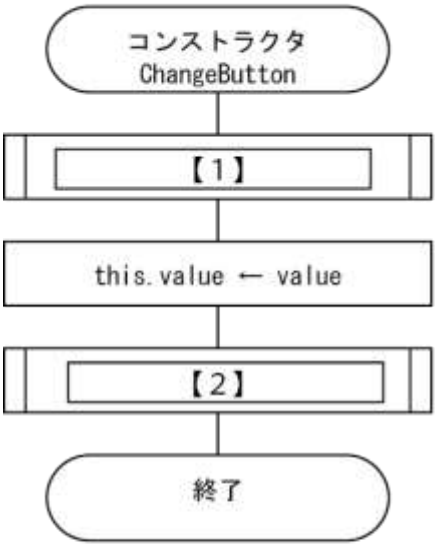
【アルゴリズム】

Problem クラス

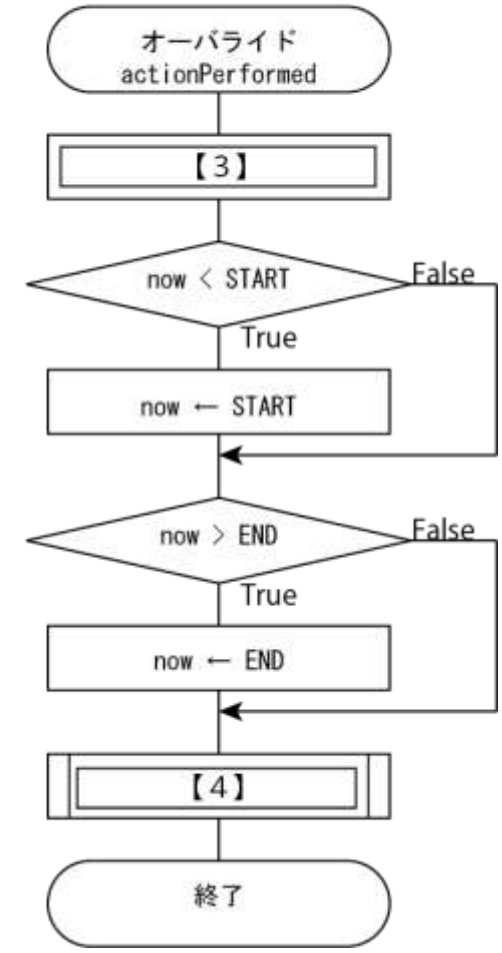
MyDrawPanel クラス



ChangeButton クラス



ChangeButton クラス



【プログラム】

```
import javax.swing.*;
import java.awt.*;
```

```
import java.awt.event.*;

public class Problem extends JFrame{
	private final static int START = 1;
	private final static int END = 30;
	private MyDrawPanel myDrawPanel;
	private int now;

	public static void main(String[] args) {
		Frame f = new Problem();
		f.setVisible(true);
	}

	public Problem(){
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		myDrawPanel = new MyDrawPanel();
		MyButtonPanel myButtonPanel = new MyButtonPanel();

		Container c = getContentPane();
		c.add(BorderLayout.SOUTH,myButtonPanel);
		c.add(BorderLayout.CENTER,myDrawPanel);

		setSize(480,340);
		setVisible(true);

		now = START;
	}

	class MyDrawPanel extends JPanel{
		public void paintComponent(Graphics g){
			String path = String.format("images/g%02d.jpg", now);
			Image image = new ImageIcon(path).getImage();
			g.drawImage(image, 0, 0, this);
		}
	}

	class MyButtonPanel extends JPanel{
		public MyButtonPanel(){
			ChangeButton b1 = new ChangeButton("前へ" , -1);
```

```
                add(b1);
                ChangeButton b2 = new ChangeButton("次へ" , 1);

                add(b2);
            }
        }

        class ChangeButton extends JButton implements ActionListener{
            private int value;
            public ChangeButton(String cap , int value){
                【1】;
                this.value = value;
                【2】;
            }

            public void actionPerformed(ActionEvent e) {
                【3】;
                if(now < START) now = START;
                if(now > END) now = END;
                【4】;

            }
        }
}
```

次の中から、上の空欄【1】～【4】に入る最も適切なものを選びなさい。

【1】 (1) super(cap) (2) JButton(cap)
(3) super(cap , value) (4) JButton(cap , value))

【2】 (1) addActionListener(this)
(2) addActionListener(myDrawPanel)
(3) addActionListener(myButtonPanel)
(4) addActionListener()

【３】　（１）　now += value　　（２）　now++
　　（３）　now--　　（４）　now = END

【４】　（１）　myDrawPanel.repaint()　　（２）　repaint()
　　（３）　this.repaint()　　（４）　myButtonPanel.repaint()

【問題 2】

列車の座席を自動販売機で予約することをシミュレートするプログラムである。予約のルールは以下のとおりである。

(1) 停車駅と座席のシート名は、Data クラスから与えられるものとする。
(2) 乗車駅番号と降車駅番号を入力すると、購入可能な座席番号を表示する。
(3) 座席番号を入力すると、予約が完了する。予約可能な座席がないときは、座席番号の入力はできない。
(4) 乗車駅と降車駅の間が連続して空席であれば予約することができる。
(5) 切符は 1 回の操作で 1 枚しか購入することはできない。
(6) 1 回購入するごとに、購入操作を続けるか、終了するか問合せを行い、True と入力すると、次の切符の購入操作を行うことができる。false と入力すると、プログラムを終了する。

なお、問題簡略化のため、車両の情報は省略しており、また、キーボードからの入力の誤りはないものとする。

例えば、座席番号 4A に上野から土浦まで予約が確定しているとき、土浦から友部までは新に予約をすることができるが、柏から友部までの予約はできない。

駅番号	駅名
0	上野
1	柏
2	土浦
3	石岡
4	友部

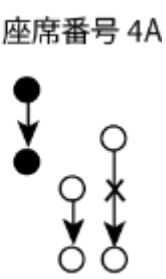


図 3　予約例

このプログラムは、以下のクラスから構成されている。

表 5　クラス Data

フィールド			
修飾子	型	名称	説明
public static	String[]	stationName	駅名を定義している。
public static	String[]	seatName	座席名を定義している。
メソッド			
修飾子	型	形式と説明	
public static	int	station_n() 駅の数を返す。	

表 6　クラス Seat

フィールド			
修飾子	型	名称	説明
private	Boolean[]	reserved	各座席について、駅間の予約状況を保持する。 空席の時は false、予約済みの時は true とする。

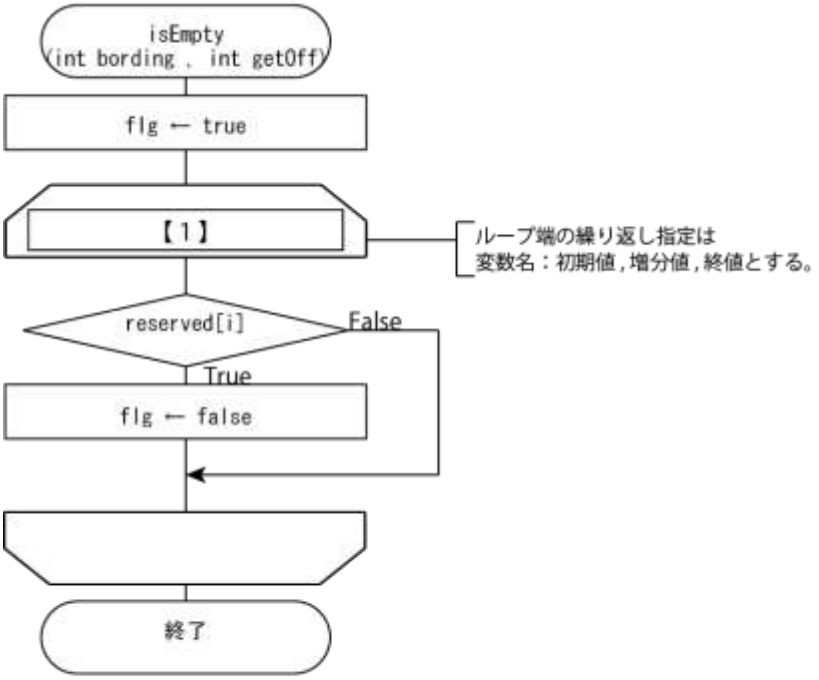
			列車が停車する駅の数- 1 個の要素を持つ。
コンストラクタ			
修飾子	形式と説明		
public	Seat() rserved を初期化する。初期値は全要素 false にする。		
メソッド			
修飾子	型	形式と説明	
public	boolean	isEmpty(int bording , int getOff) bording : 乗車駅番号 getOff : 降車駅番号 乗車区間の座席が連続して空いていれば true を、 1 駅でも予約されていれば false を返す。	
public	void	reserved(int bording , int getOff) bording : 乗車駅番号 getOff : 降車駅番号 乗車区間の reserved を true とする。	

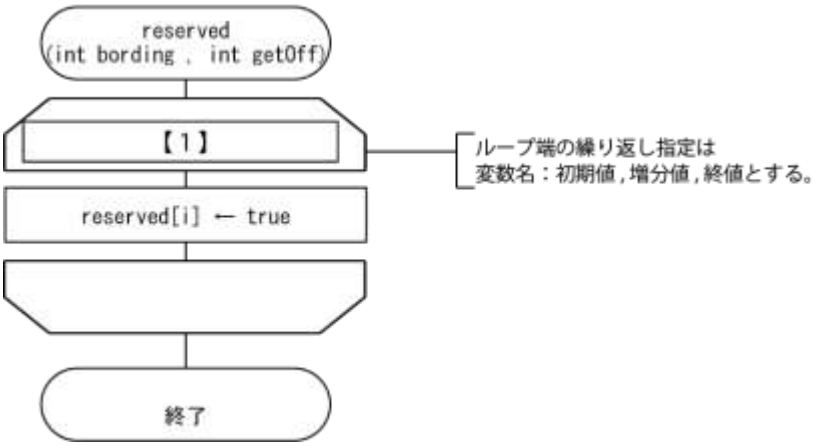
表 7　クラス Train

フィールド			
修飾子	型	名称	説明
private	Map<String , Seat>	seats	座席名を添字とする座席一覧
コンストラクタ			
修飾子	形式と説明		
public	Train() Data クラスから座席名を取得し、座席数分の Seat を生成して seats を構築する。		
メソッド			
修飾子	型	形式と説明	
public	int	dispEnable(int bording , int getOff) bording : 乗車駅番号 getOff : 降車駅番号 座席一覧と予約の可否を表示する。予約可能な座席は〇、予約できない座席は×とする。 予約できる座席の個数を返す。	
public	void	reserved(int bording , int getOff , String seatName) bording : 乗車駅番号 getOff : 降車駅番号 seatName : 座席名 指定座席の乗車区間の予約を行う。	

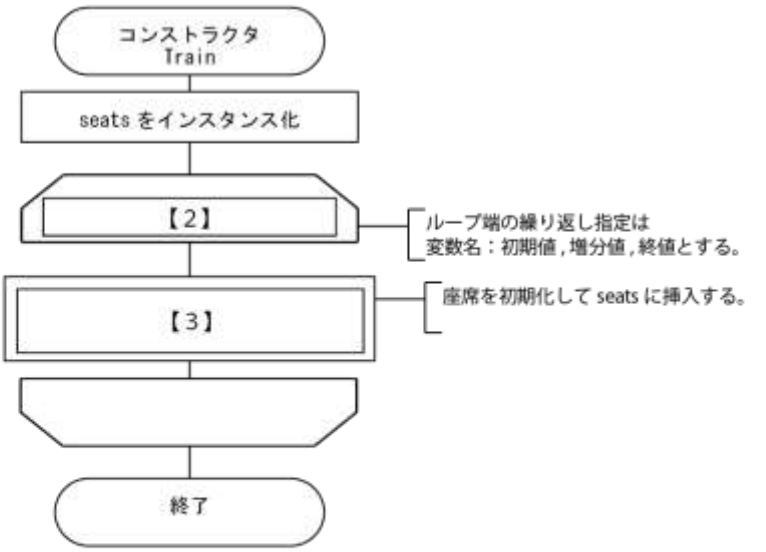
【アルゴリズム】

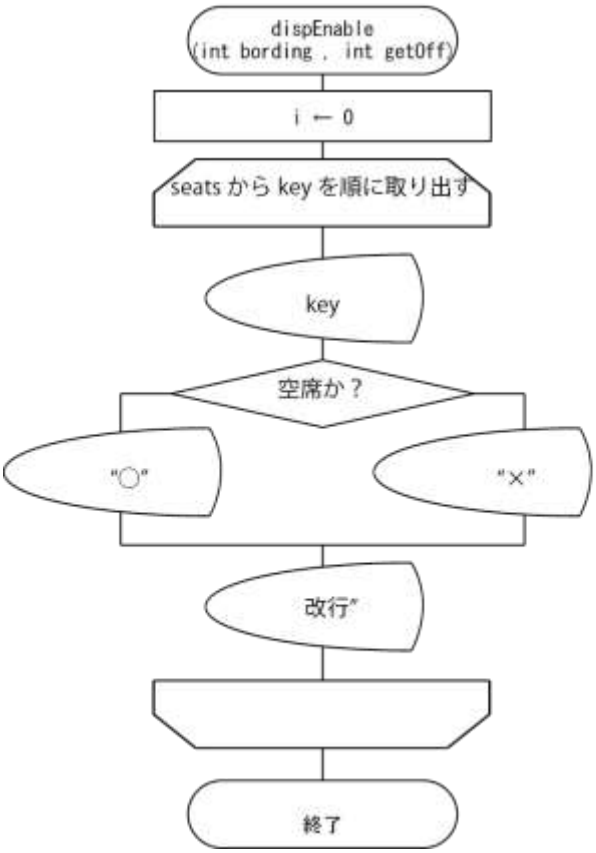
Seat クラス

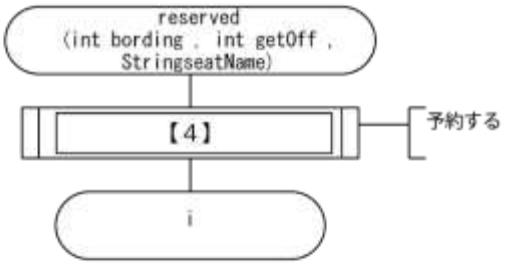




Train クラス


コンストラクタ
Train
seats をインスタンス化
【2】
ループ端の繰り返し指定は
変数名：初期値,増分値,終値とする。
【3】
座席を初期化して seats に挿入する。
終了


dispEnable
(int bording , int getOff)
i ← 0
seats から key を順に取り出す
key
空席か？
"○"
"×"
改行"
終了


reserved
(int bording , int getOff ,
StringseatName)
【4】
予約する
i

【プログラム】

```
import java.util.*;

public class Problem {
	public static void main(String[] args) {
		Train tokiwa55 = new Train();

		Scanner sc = new Scanner(System.in);
		boolean flg=true;
		while(flg) {
			System.out.print("乗車駅番号：");
			int bording = sc.nextInt();
			System.out.print("降車駅番号：");
			int getOff = sc.nextInt();
			int n = tokiwa55.dispEnable(bording, getOff);
			if (n > 0) {
				System.out.print("座席選択：");
				String seat = sc.next();
				tokiwa55.reserved(bording, getOff,seat);
			}
			System.out.print("Continue? true/false");
			flg=sc.nextBoolean();
		}
	}
}

class Data{
	public static String[] stationName = {"上野" , "柏" , "土浦" , "石岡" , "友部"};
	public static int station_n() {
		return stationName.length;
	}
	public static   String[] seatName = {"1A" , "2A" , "3A" , "4A"};
}

class Seat{
	private boolean[] reserved;

	public Seat() {
		reserved = new boolean[Data.station_n()-1];
		for(int i = 0; i< reserved.length; i++) {
			reserved[i] = false;
		}
	}

	public boolean isEmpty(int bording , int getOff) {
		boolean flg = true;
		for(          【1】          ) {
			if (reserved[i]) flg = false;
```

```
                }
                return flg;
        }

        public void reserved(int bording , int getOff ) {
                for(        【1】        ) {
                        reserved[i] = true;
                }
        }
}

class   Train{
        private Map<String , Seat> seats;

        public Train() {
                seats = new TreeMap<String , Seat>();
                for(                【2】                ) {
                        【3】        ;
                }
        }

        public int dispEnable(int bording , int getOff) {
                int i = 0;
                for(String key : seats.keySet()) {
                        System.out.print(key +" : "    );
                        if(seats.get(key).isEmpty(bording, getOff)) {
                                System.out.print("○       ");
                                i++;
                        }
                        else {
                                System.out.print("×       ");
                        }
                }
                System.out.println();
                return i;
        }

        public void reserved(int bording , int getOff , String seatName ) {
                        【4】                ;
        }
}
```

次の中から、上の空欄【１】～【４】に入る最も適切なものを選びなさい。

【１】　（１）　int i = bording; i < getOff; i++
（２）　int i = 0; i < reserved.length; i++
（３）　int i = bording; i <= getOff; i++
（４）　int i = 0; i <= reserved.length; i++

【２】　（１）　int i = 0; i < Data.seatName.length; i++
（２）　int i = 0; i <= Data.seatName.length; i++
（３）　int i = bording; i < getOff; i++
（４）　int i = bording; i <= getOff; i++

【３】　（１）　seats.put(Data.seatName[i] , new Seat())
（２）　seats.add(Data.seatName[i] , new Seat())
（３）　seats.put(new Seat())
（４）　seats.add(new Seat())

【４】　（１）　seats.get(seatName).reserved(bording, getOff)
（２）　seats.reserved(bording, getOff)
（３）　seats.add(seatName).reserved(bording, getOff)
（４）　seats.put(seatName).reserved(bording, getOff)

第 3 問（読解：1 問）

【問題】

社会現象をシミュレートするときに欠かすことができないのが乱数である。Java には一様乱数を発生する Random クラスが整備されており、これを使って各種乱数を発生することができる。一様乱数を発生する Random クラスのメソッドは以下のとおりである。

戻り値	関数の形式	機能
double 型	nextDouble()	0 以上 1 未満の 1 つの浮動小数点数を得る。

以下は乱数を 1 つ発生するプログラムである。

【プログラム】

```
import java.util.Random;
public class Problem {
        static int func(int a, double b) {
                int k = 0;
                for (int i = 0; i < a; i++) {
                        k += One(b);
                }
                return k;
        }

        static int One(double b) {
                Random random = new Random();
                double rnd = random.nextDouble();
                if (rnd < b) return 1;
                else return 0;
    }
}
```

上のプログラムにより生成される乱数の特徴を表しているのは次のうちどれか。最もふさわしいものを選択しなさい。

【選択肢】

【１】 関数 func を何度も呼び出すことで、成功する確率が b である事象を a 回行ったとき、成功する回数の分布が得られる。

【２】 関数 func を何度も呼び出すことで、時間 b のあいだに a 回事象が発生する分布が得られる。

【３】 関数 func を何度も呼び出すことで、平均 a、分散 b の正規分布が得られる。

【４】 関数 func を何度も呼び出すことで、生起率 b で事象が起きるとき、次に起きるまでの時間の分布が得られる。

実技科目

【問題１】

スタート地点からゴール地点に移動するにあたり、複数のルートの中から、もっとも短時間での移動が見込まれるルートを選択する。ただし、途中に休憩できる施設があるルートでなければならない。例えば、下表のように５つのルートがある場合、ルートＣを選択する。

表 8 ルート一覧の例

	ルートA	ルートB	ルートC	ルートD	ルートE
休憩施設	なし	あり	あり	なし	あり
見込み時間	3.8 時間	4.5 時間	4.2 時間	3.7 時間	4.4 時間

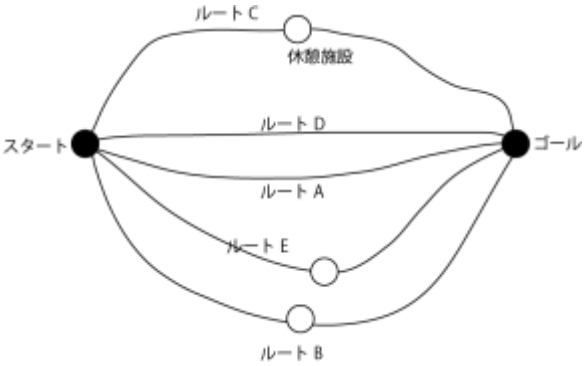


図 4　ルート例

各地点に関する情報は、表２に示すクラス Route に収められており、ルート数を要素数とする構造体配列を構成している。

表 9　クラス Route の構成

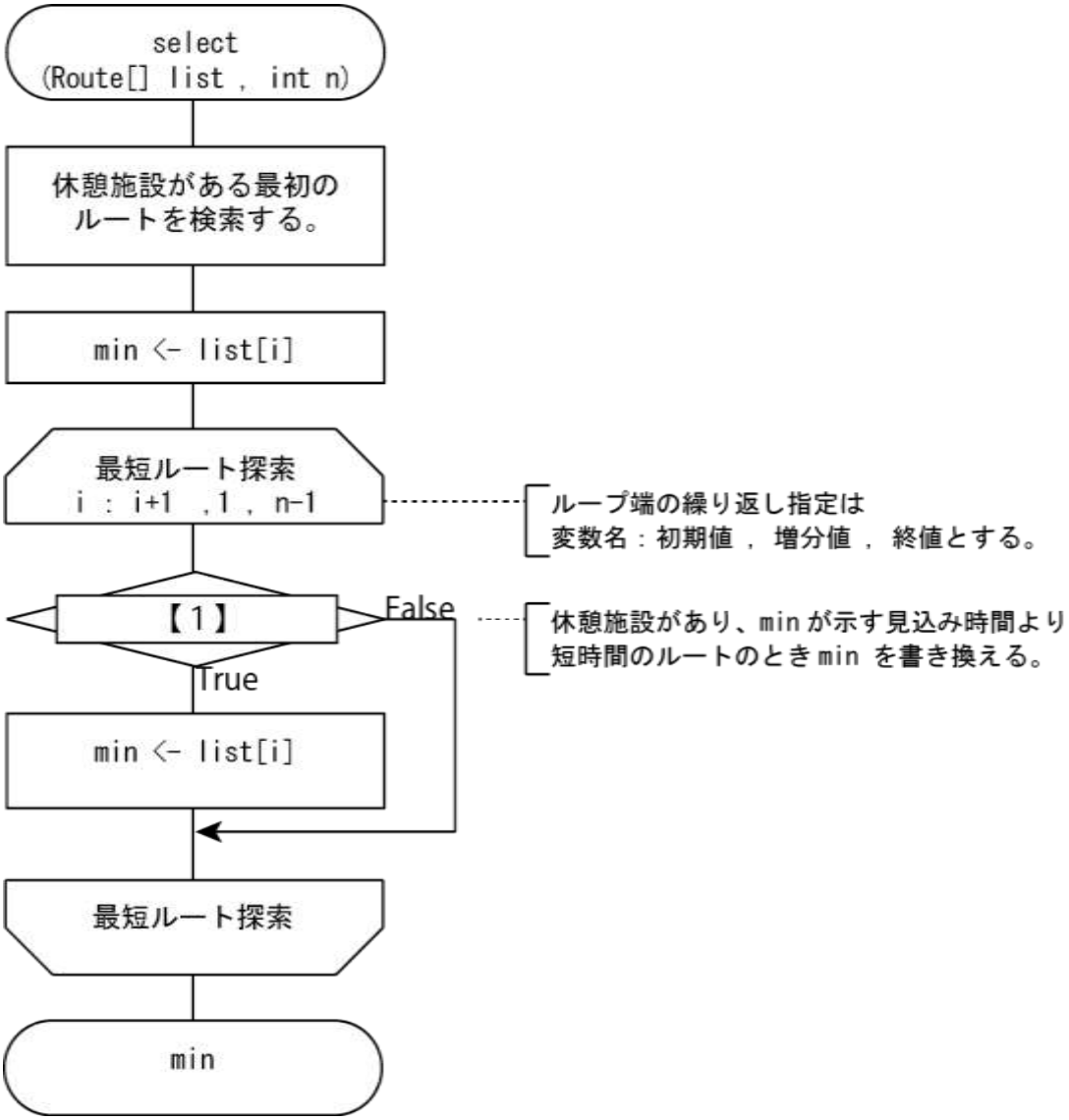
型	メンバ名	説明
String	name	ルート名
boolean	restPlace	休憩施設　true:あり　false:なし
double	requiredTime	見込み時間（単位：時間）

関数 select は、以下の仕様である。

表 10　関数 select の仕様

戻り値	関数の形式と機能
static Route	select(Route　list[] , int　n) p：ルート情報を収めた Route クラスの配列 n：ルートの個数

【流れ図】



以下のプログラムの空欄に入る適当なプログラム片を入力してプログラムを完成させなさい。

```
public class Problem {

    public static void main(String[] args) {
        Route[] data = new Route[5];
        data[0] = new Route("Route-A" , false , 3.8);
        data[1] = new Route("Route-B" , true , 4.5);
        data[2] = new Route("Route-C" , true , 4.2);
        data[3] = new Route("Route-D" , false , 3.7);
        data[4] = new Route("Route-E" , true , 4.4);

        Route selected = select(data , 5);
        System.out.println(selected.toString());
```

```
    }

    static Route select(Route[] list , int n) {
        int i;
        Route min;

        for (i = 0; i < n; i++) {
                if (list[i].isRestPlace())break;
        }

        min = list[i];
        for (i++; i < n; i++) {
                if (list[i].isRestPlace() && 【1】) {
                        min = list[i];
                }
        }
        return min;
    }
}

class Route{
    private String      name;           //ルート名
    private boolean     restPlace;      //休憩施設
    private double      requiredTime;//見込み時間

    public Route(String name , boolean restPlace , double requiredTime){
        this.name = name;
        this.restPlace = restPlace;
        this.requiredTime = requiredTime;
    }

    public boolean isRestPlace() {
        return restPlace;
    }

    public boolean isSmaller(Route target) {
        if (this.requiredTime > target.requiredTime)
                return true;
        else
```

```
                return false;
        }

        public String toString() {
            return "ルート：" + name + "　見込み時間：" + requiredTime + "時間";
        }
}
```

【問題２】

図１のような迷路をスタート地点からゴール地点に最短で移動する経路をすべて列挙するプログラムである。■は壁であり、進むことができない。また、最短で移動するために、右と下にのみ移動できる。

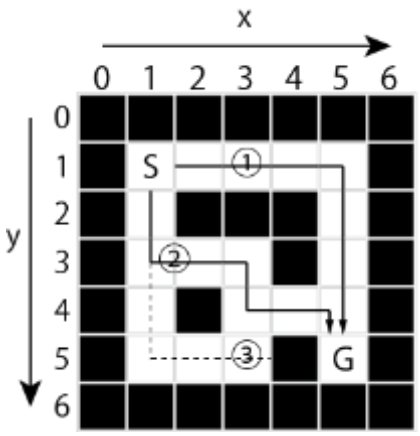


図 5　迷路の例

図１の例では、①と②はゴールにたどり着くことができるが、③は、途中で行き止まりになってしまい、ゴールにたどり着くことができない。

プログラムでは、壁の構成は、Maze クラス内の２次元配列 maze_data で与えられ、各地点は(y , x)で表現する。スタート地点 S は、(1,1)、ゴール地点 G は(5,5)である。また、地点を表す x と y の組は、クラス Position を利用する。クラス Position の仕様は表１のとおりである。

通過した位置は、Mazu クラス内の可変長配列 route に記録していく。ゴールに到達したときは route の内容を先頭から順に表示する。その後、順に戻りながら、次のルートを探す。このとき、route の内容は１つずつ破棄する。行き止まりに阻まれたときも、同様に、route の内容を破棄しながら戻っていく。クラス Maze の仕様は表２のとおりである。

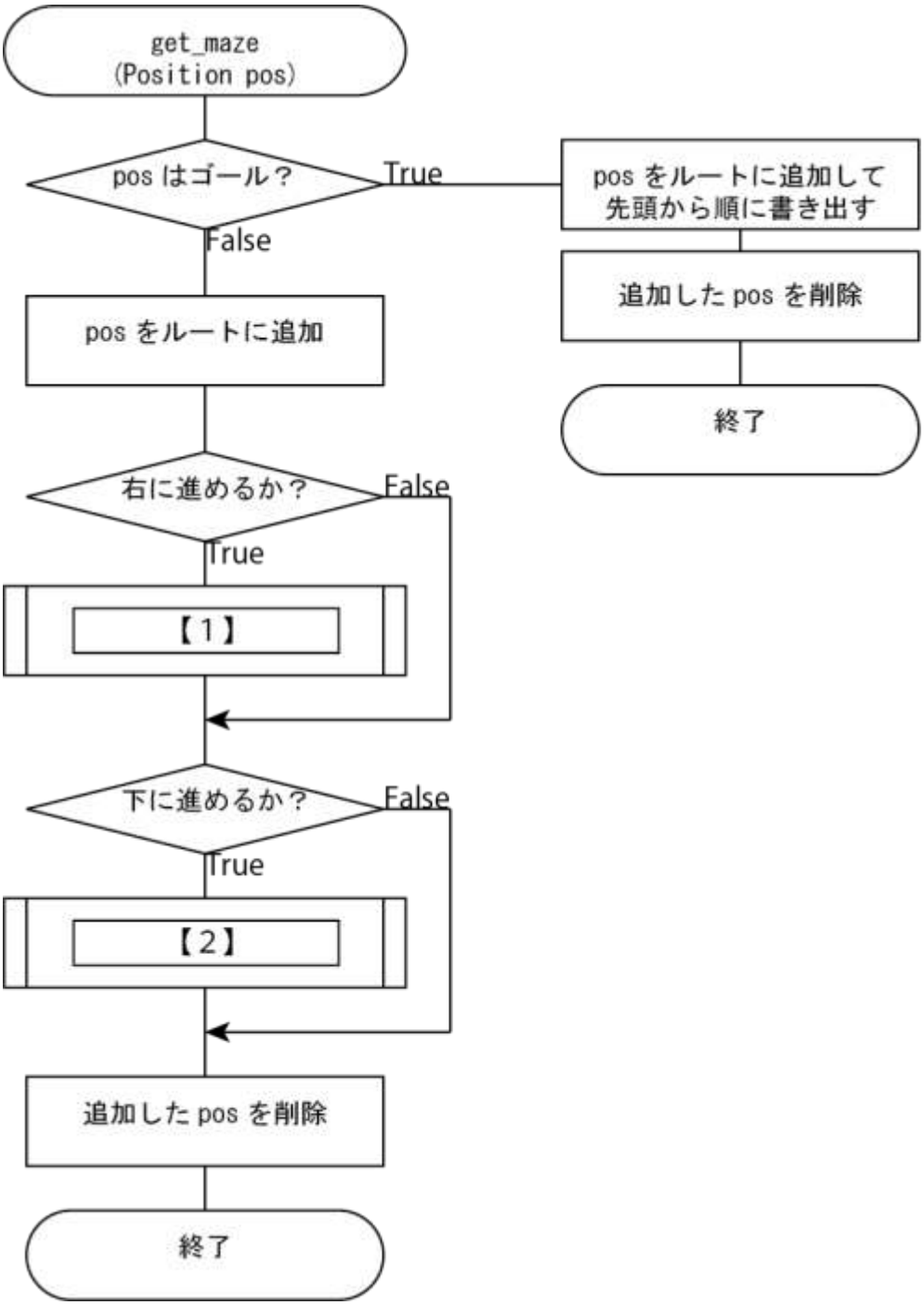
表 11　Position クラス

フィールド			
修飾子	型	名称	説明
public	int	x	横方向の添字
public	int	y	縦方向の添字
コンストラクタと説明			
Position(int x , int y) x , y を指定して初期化する。			
メソッド			
修飾子	型	メソッドと説明	
public	boolean	isGoal() ゴール位置であれば true そうでなければ false を返す。	
public	String	toString() (y , x)となる文字列を返す。	

表 12 Maze クラス

フィールド			
修飾子	型	名称	説明
private static	int[][]	maze_data	迷路の壁情報
private	List<Position>	route	経路を記録
コンストラクタと説明			
Maze() 経路を格納する route をインスタンス化する。			
メソッド			
修飾子	型	メソッドと説明	
public	void	get_maze(Position pos) スタート位置からゴール位置までの経路を表示する。 pos：開始位置を指定する。	
private	void	print_route() 経路が格納されている配列 route の内容を表示する。	
private	Boolean	check_go(Position pos , int direction) pos の右または下の壁の状態を返す。 direction　1：右　2：下	

【流れ図】



以下のプログラムの空欄に入る適当なプログラム片を入力してプログラムを完成させなさい。

```
import java.util.*;

public class Problem {

        public static void main(String[] args) {
```

```
        Maze maze = new Maze();
        maze.get_maze(new Position(1,1));
    }
}

class Maze{
    private static int[][] maze_data = {
                { 1,1,1,1,1,1,1 },
                { 1,0,0,0,0,0,1 },
                { 1,0,1,1,1,0,1 },
                { 1,0,0,0,1,0,1 },
                { 1,0,1,0,0,0,1 },
                { 1,0,0,0,1,0,1 },
                { 1,1,1,1,1,1,1 }
    };
    private List<Position> route;

    public Maze() {
        route = new ArrayList<Position>();
    }

    public void get_maze(Position pos) {
        if (pos.isGoal()) {
                route.add(pos);
                print_route();
                route.remove(route.size()-1);
                return;
        }

        route.add(pos);
        if (!check_go(pos, 1)) {
                【1】;
        }
        if (!check_go(pos, 2)) {
                【2】;
        }
        route.remove(route.size()-1);
    }
```

```
        private void print_route() {
            for (Position one :route ) {
                        System.out.print(one.toString());
            }
            System.out.println();
        }

        private boolean check_go(Position pos , int direction) {
            int i, j;
            switch (direction) {
            case 1:
                        i = pos.x + 1;
                        j = pos.y;
                        break;
            case 2:
                        i = pos.x;
                        j = pos.y + 1;
                        break;
            default:
                        i = pos.x;
                        j = pos.y;
            }

            if(maze_data[j][i] == 0) return false;
            else return true;
        }
}

class Position{
        public int      x;
        public int      y;

        public Position(int x , int y) {
            this.x = x;
            this.y = y;
        }

        public boolean isGoal() {
            if (x == 5 && y == 5)
```

```
                return true;
            else
                return false;
        }

        public String toString() {
            return "(" + y + " , " + x + ")";
        }
}
```

【問題 3】

シャーレの中でバクテリアがどのように増殖するのかをシミュレートするプログラムである。シャーレを微小な空間に分け、その小さな一つひとつの空間にバクテリアが最大 1 匹存在できるものとする。図 1 では初期状態として、6 匹のバクテリアが存在している。

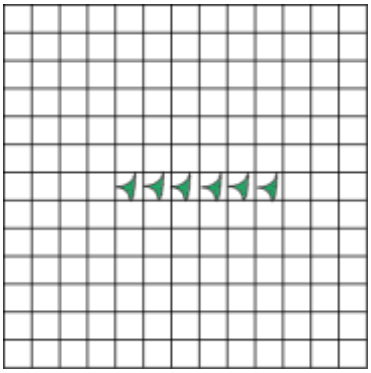


図 6　微小空間にいるバクテリア

バクテリアは、環境により次のように変化する。

① バクテリアは、図 2 に示す近傍 8 個のセルのうち 2 個または 3 個のセルにバクテリアが存在すれば、生き続けられる。しかし、2 個未満のときは、寂しくて死んでしまう。また、4 個以上の時は、過密すぎて死んでしまう。

② バクテリアが存在しないセルでは、図 3 に示す近傍 8 個のセルのうち 3 個のセルにバクテリアが要れば、分裂により新たに誕生することができる。

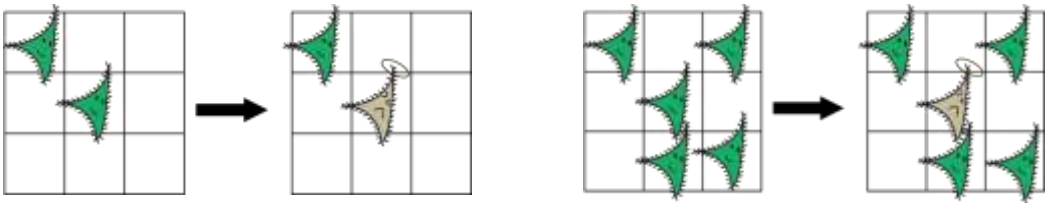


図 7　周りのバクテリアが少なくても多くても死んでしまう

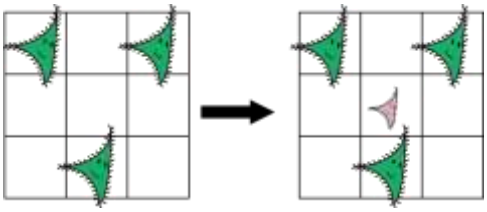


図 8　近傍が 3 匹であれば赤ちゃん誕生

以上のルールに従って、時間が進んだとき、バクテリアがどのように変化しているかをシミュレートする。ただし、領域の外側にはバクテリアは生存せず、生誕することもできないものとする。

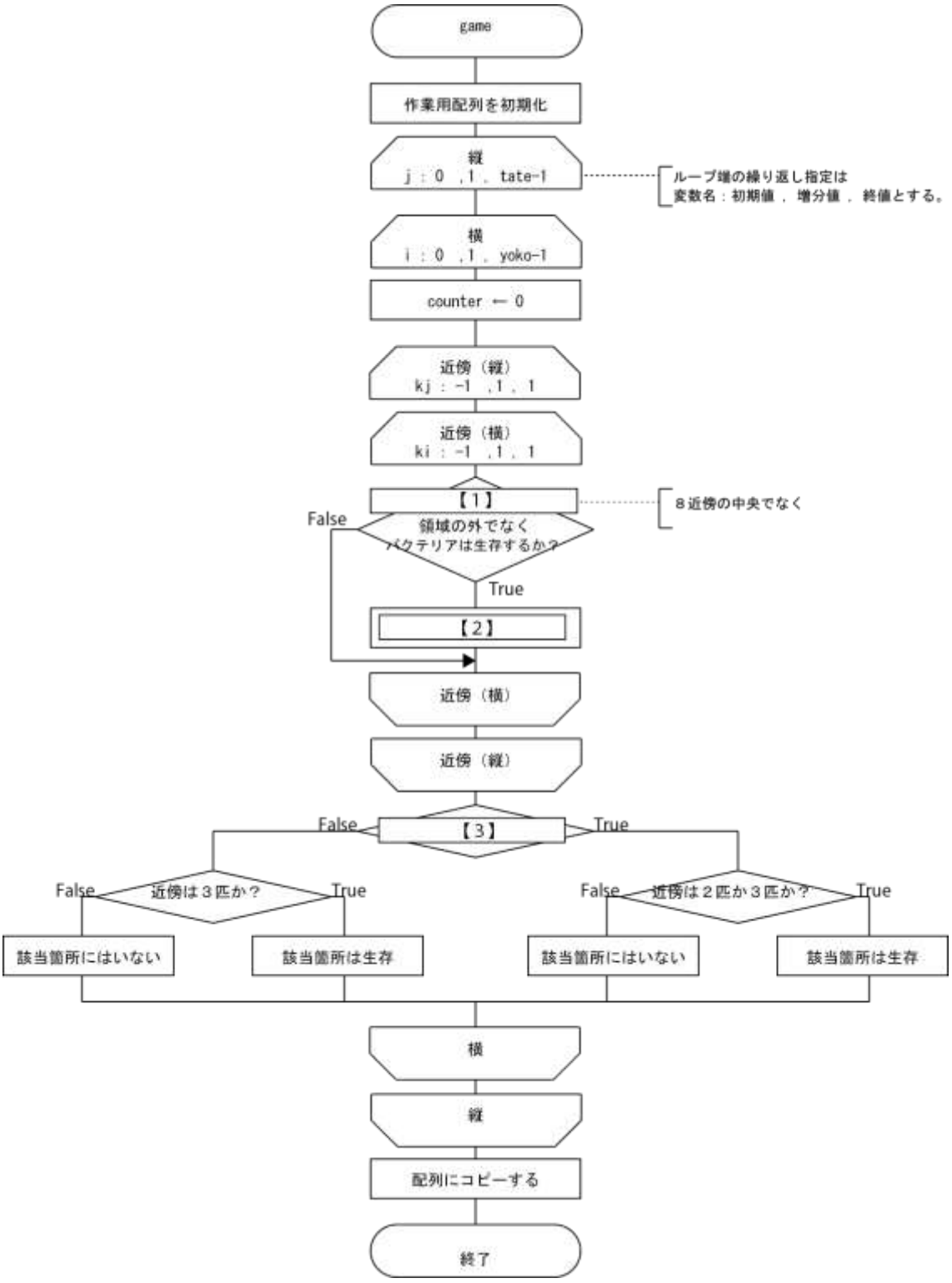
プログラムでは、13×13 の領域に分けており、定数 TATE と YOKO で定義している。微小空間の管理はクラス World で行う。World 内の 2 次元配列 cell に各微小空間のバクテリアの状態を記録しており、バクテリアが生きている状態を true、いない状態を false とする。

バクテリアの状態を 1 2 回更新したときのバクテリアの状態を表示する。World クラスの仕様は表 1 のとおりである。

表 13　World クラス

フィールド			
修飾子	型	名称	説明
private	int	tate	縦方向の要素数
private	int	yoko	横方向の要素数
private	Boolean[][]	cell	バクテリアの状態を記録する配列
コンストラクタと説明			
World(int tate , int yoko) 配列 cell を初期化			
メソッド			
修飾子	型	メソッドと説明	
public	void	set(int i , int j) cell[i , j]をバクテリアがいる状態、true とする。	
public	void	game() バクテリアの状態を 1 回更新する。配列 cell が更新される。	
public	void	disp() 配列に従い、バクテリアが生存すれば■を、しなければ□を表示する。	

【流れ図】



以下のプログラムの空欄に入る適当なプログラム片を入力してプログラムを完成させなさい。

```
public class Problem {
	private static final int TATE = 13;
	private static final int YOKO = 13;

	public static void main(String[] args) {
		int[][] init= { {TATE/2 , YOKO/2-2} , {TATE/2 , YOKO/2-1} , {TATE/2 , YOKO/2} ,
				{TATE/2 , YOKO/2+1} , {TATE/2 , YOKO/2+2} , {TATE/2 , YOKO/2+3}};
```

```
        World world = new World(TATE , YOKO);

        for(int i = 0; i < init.length; i++) {
            world.set(init[i][0] , init[i][1]);
        }

        for(int i = 0; i < 12; i++) world.game();
        world.disp();

    }

}
class World{
    private boolean[][] cell;
    private int tate;
    private int yoko;

    public World(int tate , int yoko) {
        this.tate = tate;
        this.yoko = yoko;
        cell = new boolean[tate][yoko];
    }

    public void set(int j , int i) {
        cell[j][i] = true;
    }

    public void game() {
        boolean[][] next = new boolean[tate][yoko];
        for(int j = 0; j < tate; j++) {
            for(int i = 0; i < yoko;i++) {
                int counter = 0;
                for(int kj=-1;kj<=1;kj++) {
                    for(int ki=-1;ki<=1;ki++) {
                        if(      【1】      ) {
                            if(j+kj >= 0 && j+kj < tate && i+ki >= 0 && i+ki < yoko) {
                                if(cell[j+kj][i+ki])      【2】      ;
                            }
                        }
                    }
                }
                if(      【3】      ) {
                    if(counter<2 || counter > 3)
                        next[j][i] = false;
                    else
                        next[j][i] = cell[j][i];
                }
                else {
```

```
                        if(counter == 3) {
                            next[j][i] = true;
                        }
                        else
                            next[j][i] = cell[j][i];
                    }
                }
            }

            for(int j =0; j < tate;j++) {
                for(int i = 0; i < yoko;i++) {
                    cell[j][i] = next[j][i];
                }
            }
        }

        public void disp() {
            for(int j=0; j < tate; j++) {
                for(int i =0;i<yoko;i++) {
                    if(cell[j][i])
                        System.out.print("■");
                    else
                        System.out.print("□");
                }
                System.out.println();
            }
        }
}
```