オブジェクト指向言語・期末テスト問題用紙
（2024 年 08 月 01 日・ 08:50 ～ 10:20）

解答上、その他の注意事項

1. 問題は、問 Ⅰ～Ⅶ まである。うち、<u>問 Ⅰ～Ⅲ は中間テストの代替問題である。</u>

   - 中間テストの得点が 7 割に達しなかったものは、 代替問題を解答すれば、7 割を上限として良いほうの点数を採用する。
   - 正当な事情があって中間テストを欠席した者も代替問題を解答すること。（この場合は、上限は設けない。）

2. 解答用紙の右上の欄に学籍番号・名前を記入すること。

3. 解答欄を間違えないよう注意すること。

4. 解答中の文字 (特に a と d)がはっきりと区別できるよう注意すること。

5. 持ち込みは不可である。筆記用具・時計・学生証以外のものは、かばんの中などにしまうこと。

6. テストの配点は 70 点（うち中間テストの代替問題の問 Ⅰ～Ⅲの配点は 30 点）である。合格は「オブジェクト指向言語演習」の課題の得点を加点して、100 点満点中 60 点以上とする。

すべての問に対する補足

プログラムの空欄を埋める問題では、解答が長くなる可能性があるので、下の省略形（◯囲み文字）を用いても良い。（必ず ◯ で囲むこと。）

Ⓐ `ActionListener` ⓐA `addActionListener` ⒶⒺ `ActionEvent` Ⓚ `KeyListener`
ⓐK `addKeyListener` ⓀⒺ `KeyEvent` Ⓜ `MouseListener` ⓐM `addMouseListener`
ⓂⒺ `MouseEvent` ⓟl `System.out.println` ⓟf `System.out.printf`

また、参考のために問題用紙の末尾に授業配布プリントの LeftRightButton.java, LeftRightButton2.java, LeftRightButton3.java, Guruguru.java, Point.java, ColorPoint.java, Quadratic.kt, SequenceTest.kt のソースを掲載する。（GUI アプリケーションは `main` メソッドは省略している。）

I. 次の (1)〜(2) の Java に関する多肢選択問題に答えよ。解答は各問の指示する選択肢から選べ。ただし、特に指定しない限り、選ぶべき選択肢は必ずしも**一つとは限らない**。

(1) 文法上 Java のクラス名として使用できるものはどれか? すべて選べ。

(A). `HelloWorld` (B). `Rock'n'Roll` (C). `3_2_1_Go` (D). `A_B_C_`

(2) 次の Java に関する文章のうち、正しいものはどれか? すべて選べ。

(A). Java は仕様を簡潔に保つため、公開されて以来一度も言語仕様を変更していない。
(B). Java の配列は範囲外をアクセスすると、実行時に例外を発生する。
(C). Java は JavaScript に型宣言を仕様として追加したプログラミング言語である。
(D). Java では `null` のフィールドやメソッドにアクセスするというエラーはコンパイル時に検知される。
(E). Java の 1 つのクラスは 2 つ以上のインタフェースを実装 (implement) してもよい。
(F). Java 言語は、コンパイル時に型チェックを行わない、いわゆる動的な言語である。

II. 次の枠内の文章は `java.awt.Polygon` クラスの `contains`, `getBounds2D`, `intersects`, `translate` メソッド、 `java.awt.Graphics2D` クラスの `draw`, `fill` メソッド、 `java.lang.Math` クラスの `random` メソッドの Java API Reference からの抜粋である。 `java.awt.Polygon` クラスは `java.awt.Shape` インタフェースを実装している。（解くのに関係ない部分は割愛し、説明を簡単にするために改変していることがある。）

パッケージ `java.awt`

クラス `Polygon`

```
public class Polygon implements Shape, Serializable
```

`Polygon` クラスは、座標空間内の閉じられた 2 次元領域の記述をカプセル化します。

メソッドの詳細

contains

```
public boolean contains(int x, int y)
```

指定された座標がこの `Polygon` の内側にあるかどうかを判定します。

パラメーター:
`x` - テストされる指定された X 座標
`y` - テストされる指定された Y 座標
戻り値:
この `Polygon` に、指定された座標 `(x,y)` が含まれる場合は `true`、 それ以外の場合は `false`。

getBounds2D

```
public Rectangle2D getBounds2D()
```

高精度な `Polygon` のバウンディング・ボックス†を返します。

戻り値:
Polygon の高精度のバウンディング・ボックスである Rectangle2D のインスタンス。

†取り囲む長方形のこと

intersects

```
public boolean intersects(Rectangle2D r)
```

Polygon の内部が指定された Rectangle2D の内部と交差しているかどうかをテストします。

パラメーター:
r - 指定されたRectangle2D
戻り値:
Polygon の内部と指定された Rectangle2Dの内部が交差している場合は true、 それ以外の場合はfalse。

translate

```
public void translate(int deltaX, int deltaY)
```

Polygonの頂点を、X軸に沿ってdeltaX、Y軸に沿ってdeltaY 平行移動します。

パラメーター:
deltaX - X 軸に沿って移動する距離
deltaY - Y 軸に沿って移動する距離

パッケージ java.awt

クラス Graphics2D

```
public class Graphics2D extends Graphics
```

Graphics2D クラスは、Graphics クラスを拡張して、幾何学的図形、座標変換、色の管理、およびテキスト・レイアウトに対する、より高度な制御を提供します。

メソッドの詳細

draw

```
public void draw(Shape s)
```

現在の Graphics2D コンテキストの設定を使用して、Shape の輪郭をストロークで描画します。

パラメーター:
s - 描画される Shape

fill

```
public void fill(Shape s)
```

Graphics2D コンテキストの設定を使用して、Shape の内部を塗りつぶします。

パラメーター:
s - 塗りつぶされる Shape

パッケージ java.lang

クラス Math

public class Math

メソッドの詳細

random

public static double random()

0.0 以上で 1.0 未満の、正の double 値を返します。

戻り値:
0.0 以上 1.0 未満の疑似乱数の double 値。

下に示す ManyPolygons クラスは、これらのメソッドを使って、以下のような方法で幅 width、高さ height のウインドウに複数の Polygon を配置する。

- xMax, yMax を Polygon のバウンディング・ボックスのそれぞれ幅と高さとするとき、 0 以上 (width - xMax) 未満と 0 以上 (height - yMax) 未満の乱数を生成して、これを左上の点の座標とするように Polygon を配置する。
- ただし、これまでに配置した Polygon のバウンディング・ボックスと新しく配置しようとする Polygon が交差する (intersect) するときは、Polygon を追加せず、失敗を記録する。
- ある一定の数（ここでは 20）回連続して失敗した場合は、そこで Polygon の追加を終了する。

そして、配置された Polygon を以下のように描画する

- すべての Polygon は、水色 (CYAN) で内部を塗りつぶす。
- 加えて、(0.25 * width, 0.25 * height),(0.5 * width, 0.5 * height), (0.75 * width, 0.75 * height) という特別な点を内部に含む (contains) Polygon は、青色 (BLUE) で輪郭を描画する。

次のプログラムを見て、以下の問に答えよ。

ファイル名 ManyPolygons.java

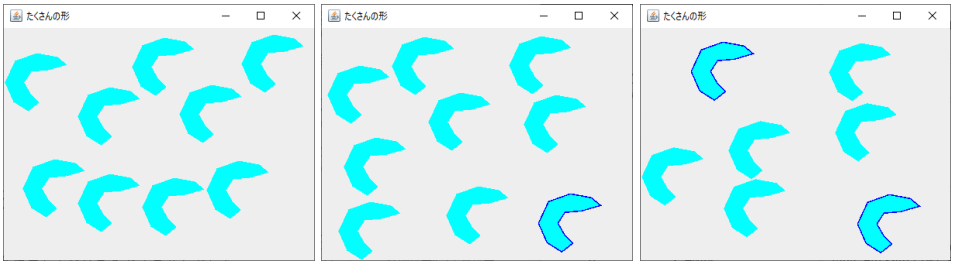
```
1 import java.awt.*;
2 import java.util.ArrayList;
3 import javax.swing.*;
4
5 public class ManyPolygons extends JPanel  {
```

```
    // 適当な多角形（13 角形）
    private static final int[] xpoints = {
        80, 68, 41, 13,  0, 11, 30, 44, 33, 25, 34, 56, 80 };
    private static final int[] ypoints = {
        15,  5,  0, 10, 38, 63, 75, 64, 52, 38, 24, 22, 15 };
    // xpoints, ypoints の最大値
    private static final int xMax = 80, yMax = 75;
    // ウインドウのサイズ
    private static final int width = 400, height = 300;
    private static final int maxFail = 20;
    ArrayList<Polygon> polygons = new ArrayList<>();

    private boolean mayCollide(Polygon p1) {
        for (Polygon p2: polygons) {
            if (                (1)                ) return true;
        }
        return false;
    }
    public ManyPolygons() {
        setPreferredSize(new Dimension(width, height));
        int fails = 0;

        while (true) {
            Polygon shape = new Polygon(xpoints, ypoints, 13);
            int x =                     (2)                    ;
            int y =                     (3)                    ;
                                (4)
            if (mayCollide(shape)) {
                fails++;
                if (fails >= maxFail) break;
                continue;
            }
            fails = 0;
            polygons.add(shape);
        }
    }

    public void paintComponent(Graphics g) {
        Graphics2D g2 = (Graphics2D)g;
        for (Polygon p: polygons) {
            g2.setColor(Color.CYAN);
            g2.fill(p);
            if (                    (5)                    
              ||                    (6)                    
              ||                    (7)                    ) {
                g2.setStroke(new BasicStroke(1.5f));
                g2.setColor(Color.BLUE);
                g2.draw(p);
            }
        }
    }
    /* main は省略する */
}
```



実行例 1　　実行例 2　　実行例 3

(1) の空欄に、`p1` が `p2` のバウンディング・ボックスと交差する (intersects) かどうかを判定する式が入る。（Polygon 同士が交差するかどうか判定するメソッドはない。）この式を答えよ。

(2) の空欄に 0 以上 `(width - xMax)` 未満の整数値の乱数を返す式が入る。この式を答えよ。
((3) は (2) とほとんど同じなので答えなくて良い。)

(4) の空欄に、`shape` を (x, y) だけ平行移動 (translate) する文が入る。この文を答えよ。

(5) の空欄に、`p` が座標 `(0.5 * width, 0.5 * height)` を含む (contains) かどうかを判定する式が入る。この式を答えよ。 ((6), (7) は (5) とほとんど同じなので答えなくて良い。)

III. 次の `BinaryCalc` クラスは二進数の逆ポーランド記法（後置記法）電卓の GUI アプリケーションである。数字のボタン [0]、[1] と演算子のボタン [+]、[−]、[×]、[÷]、クリアボタン [C]、エンターボタン [↵] を持ち、上部の JLabel に結果を二進数で表示する。例えば `10↵11×100+` と入力すると、（$2 \times 3 + 4 = 10$ なので） `1010` と表示され、`10↵11↵100+×` と入力すると、（$2 \times (3 + 4) = 14$ なので）`1110` と表示される。

また `BinaryCalc` クラスは `JPanel` を継承している。

ファイル名 BinaryCalc.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import java.util.Stack;
4  import javax.swing.*;
5
6  public class BinaryCalc                    (1)                    {
7      private JLabel display;
8      private int current = 0;
9      private Stack<Integer> stack;
10     private JButton btn0, btn1, btnP, btnM, btnT, btnD,
11                     btnE, btnC;
12     private boolean append = true, auto_e = false;
13
14     // 整数 → 二進数への変換
15     private String toBinary(int n) {
16         String ret = "";
17         if (n > 1) {
18             ret += toBinary(n / 2);
19         }
20         ret += n % 2;
21         return ret;
22     }
```

```
    public BinaryCalc() {
        setPreferredSize(new Dimension(150, 120));
        stack = new Stack<>();
        // 改ページの都合で updateDisplay の定義は下
        display = new JLabel(); updateDisplay();
        display.setOpaque(true);
        display.setBackground(Color.WHITE);
        btn0 = new JButton("0");
        btn1 = new JButton("1");
        btnP = new JButton("+"); // Plus
        btnM = new JButton("−"); // Minus
        btnT = new JButton("×"); // Timer
        btnD = new JButton("÷"); // Divide
        btnE = new JButton("↲"); // Enter
        btnC = new JButton("C"); // Clear
        add(display);
        for (JButton b : new JButton[] {
                btn0, btn1, btnP, btnM, btnT, btnD, btnE, btnC
        }) { add(b); b.addActionListener(this); }
    }

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        if (source == btn0) {
            if (append) {
                current *= 2;
            } else {
                if (auto_e) {
                    stack.push(current); auto_e = false;
                }
                current = 0; append = true;
            }
        } else if (source == btn1) {
            if (append) {
                current *= 2; current += 1;
            } else {
                if (auto_e) {
                    stack.push(current); auto_e = false;
                }
                current = 1; append = true;
            }
        } else if (source == btnC) {
            current = 0;           append = true;  auto_e = false;
        } else if (source == btnE) {
            stack.push(current);   append = false; auto_e = false;
        } else if (source == btnP) {
            int x = stack.pop();
            current = x + current; append = false; auto_e = true;
        } else if (source == btnM) {
            int x = stack.pop();
            current = x - current; append = false; auto_e = true;
        } else if (source == btnT) {
            int x = stack.pop();
            current = x * current; append = false; auto_e = true;
        } else if (source == btnD) {
            int x = stack.pop();
            current = x / current; append = false; auto_e = true;
        }
        // 改ページの都合で updateDisplay の定義は下
        updateDisplay();
    }
```

```
84      // current を二進数に変換して display に表示する
85      private void updateDisplay() {
86          int n = current;
87          String ret = "";
88          if (n < 0) {
89              ret += "-"; n *= -1;
90          }
91          ret += toBinary(n);
92          int len = ret.length();
93          if (len < 32) {
94              ret = " ".repeat(32 - len) + ret;
95          }
96          display.setText(ret);
97      }
98      /* main は省略する */
99  }
```

念のため、ここで使われている Stack クラスの API ドキュメントを掲載する。

パッケージ java.util

クラス Stack<E>

```
public class Stack<E> extends Vector<E>
```

Stack クラスは、オブジェクトの後入れ先出し (LIFO) スタックを表します。

メソッドの詳細

pop

```
public E pop()
```

スタックの先頭のオブジェクトを削除し、そのオブジェクトを関数の値として返します

戻り値:
　　このスタックの上部にあるオブジェクト
例外:
　　EmptyStackException - このスタックが空の場合

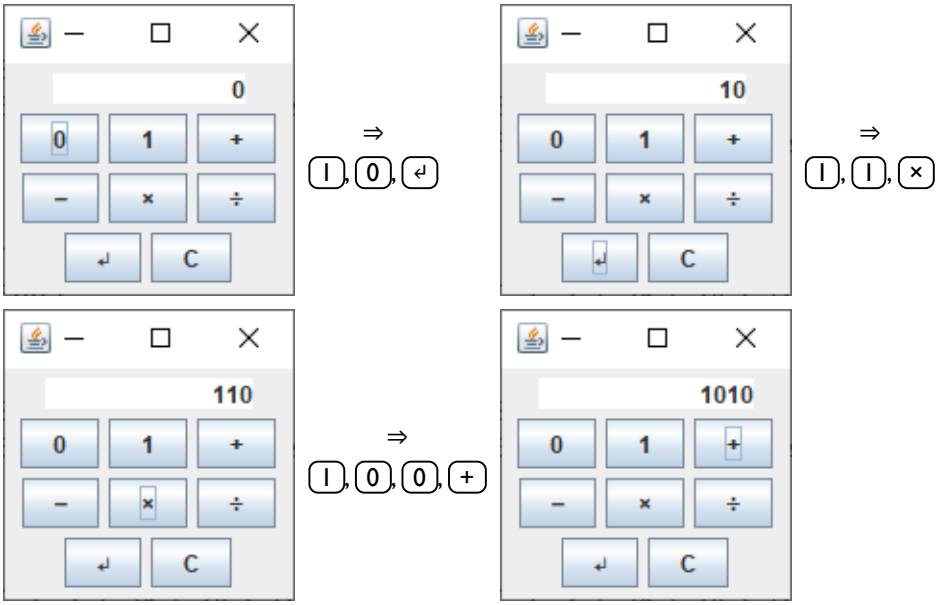
push

```
public E push(E item)
```

スタックの先頭にオブジェクトを入れます。

パラメーター:
　　item - スタックに入れるオブジェクト。
戻り値:
　　item 引数

実行例 （次ページ）:



- 上の (1) の空欄を埋めよ。

このプログラムを内部クラス、匿名クラス、ラムダ式を用いて次のように書き換える。

```
1  /* import は変わらないので省略する */
2  public class BinaryCalc2 (2) {
3      private JLabel display;
4      private int current = 0;
5      private Stack<Integer> stack;
6      private boolean append = true, auto_e = false;
7      /* toBinary, updateDisplay は変わらないので省略する */
8      public BinaryCalc2() {
9          setPreferredSize(new Dimension(150, 120));
10         stack = new Stack<>();
11         display = new JLabel(); updateDisplay();
12         display.setOpaque(true);
13         display.setBackground(Color.WHITE);
14         JButton btn0 = new JButton("0");
15         JButton btn1 = new JButton("1");
16         JButton btnP = new JButton("+"); // Plus
17         JButton btnM = new JButton("−"); // Minus
18         JButton btnT = new JButton("×"); // Times
19         JButton btnD = new JButton("÷"); // Divide
20         JButton btnE = new JButton("↵"); // Enter
21         JButton btnC = new JButton("C"); // Clear
22         add(display); add(btn0); add(btn1);
23         add(btnP); add(btnM); add(btnT); add(btnD);
24         add(btnE); add(btnC);
25         btn0.addActionListener( (3) );
26         btn1.addActionListener( (4) );
27         btnP.addActionListener( (5) );
28         btnM.addActionListener( (6) );
29         btnT.addActionListener( (7) );
30         btnD.addActionListener( (8) );
31         btnC.addActionListener( (9) );
32         btnE.addActionListener(ev -> {
33             stack.push(current); append = false; auto_e = false;
34         });
35     }
```

```
37      private class NumHandler implements ActionListener {
38          int n;
39
40          public NumHandler(int n0) {
41              n = n0;
42          }
43
44          public void actionPerformed(ActionEvent e) {
45            if (append) {
46                current *= 2; current += n;
47            } else {
48                if (auto_e) {
49                    stack.push(current); auto_e = false;
50                }
51                current = n; append = true;
52            }
53            updateDisplay();
54        }
55      }
56
57      private class OpHandler implements ActionListener {
58          public int calculate(int x, int y) {
59              return x + y;
60          }
61
62          public void actionPerformed(ActionEvent e) {
63              int x = stack.pop();
64              current = calculate(x, current);
65              append = false; auto_e = true;
66              updateDisplay();
67          }
68      }
69
70      /* main は省略する */
71  }
```

- (2) の空欄を埋めよ。なお `BinaryCalc2` クラスも `JPanel` を継承している。
- 数字ボタン [0]、[1] の対応はほとんど同じなので、内部クラス `NumHandler` を定義する。(3), (4) には `NumHandler` を使用した式が入る。 (3), (4) の空欄を埋めよ。
- 演算子のボタンの対応はほとんど同じなので、内部クラス `OpHandler` を定義する。 (5) の空欄には、内部クラス `OpHandler` を使用した式が、 (6) ～ (8) の空欄には、`OpHandler` と匿名クラスを使った式が入る。 (5), (6) の空欄を埋めよ。 ((7), (8) は (6) とほとんど同じなので答えなくて良い。)
- クリアボタンとエンターボタンの対応は独特なので、ラムダ式で記述する。 (9) の空欄に入るラムダ式を答えよ。

なお、(3) ～ (9) では `getSource` メソッドは使用しないこと。

IV. 次の (1) ～ (2) の Kotlin プログラムはどのように出力するか? 答えよ。なお、シーケンスの `take(n)` メソッドは先頭の *n* 個の要素からなる有限列を取り出す。

(1)

```
val foo: Sequence<Pair<Int, Int>> = sequence {
    var x = 1
    var y = 0;
    while (true) {
        yield(Pair(x, y))
        var t = x
        x = t - y
        y = t + y
        yield(Pair(x, y))
        t = x
        x = (t - y) / 2
        y = (t + y) / 2
    }
}

fun main() {
    println(foo.take(5).toList())
}
```

(2)

```
val bar: Sequence<Pair<Int, Int>> = sequence {
    val queue = ArrayDeque<Pair<Int, Int>>()
    queue.add(Pair(3, 1))
    while (true) {
        val (m, n)  = queue.removeFirst()
        yield(Pair(m, n))
        queue.add(Pair(2 * m - n, m))
        queue.add(Pair(2 * m + n, m))
        queue.add(Pair(m + 2 * n, n))
    }
}

fun main() {
    println(bar.take(10).toList())
}
```

念のため、ここで使われている `ArrayDeque` クラスの API ドキュメントを（Java API ドキュメント風の書き方で）掲載する。

パッケージ kotlin.collections

クラス ArrayDeque<E>

`public class ArrayDeque<E> extends AbstractMutableList<E>`

deque (double-ended queue) データ構造のリサイズ可能な配列による実装。

メソッドの詳細

removeFirst

```
public E removeFirst()
```

この deque から最初の要素を削除し、その削除した要素を返します。

戻り値:
この deque の最初の要素
例外:
`NoSuchElementException` - この deque が空の場合

add

```
public Boolean add(E element)
```

指定された要素をリストの最後に追加します。

パラメーター:
`element` - 追加するオブジェクト。
戻り値:
常に true を返す。

V. 次の (1) ～ (2) の多肢選択問題に答えよ。ただし、特に指定しない限り、選ぶべき選択肢は必ずしも**一つとは限らない**。

(1) 次の選択肢の中で関数型言語に分類される言語はどれか? もっともふさわしいものを一つ選べ。

(A). C (B). Java (C). Haskell (D). Prolog (E). Smalltalk

(2) 次の選択肢の中で、Java より登場が新しい（若い）言語はどれか? すべて選べ。

(A). Fortran (B). C# (C). C (D). C++ (E). Kotlin (F). Python

VI. 育成ゲーム用のクラス階層を設計するために、テスト用のいくつかのクラスを作成する。次に定義されるクラス `Fish` を継承して、

ファイル名 Fish.java

```java
public class Fish {
    protected int age;
    public void grow() { age++; }
    public void showName() { System.out.print("サカナ "); }
    public void growShowName() { showName(); grow(); }
}
```

3 つのサブクラス `Carp`, `Seriola`, `Mugil` を定義する。

ファイル名 Carp.java

```
public class Carp extends Fish {
    public String color;
    public Carp(String c) {  color = c;  }
    @Override
    public void showName() {
        if (age < 1000) {
            System.out.print(color + "鯉 ");
        } else {
            System.out.print(color + "龍 ");
        }
    }
}
```

ファイル名 Seriola.java

```
public class Seriola extends Fish {
    @Override
    public void showName() {
        if (age == 0) {
            System.out.print("ツバス ");
        } else if (age <= 2) {
            System.out.print("ハマチ ");
        } else {
            System.out.print("ブリ ");
        }
    }
}
```

ファイル名 Mugil.java

```
public class Mugil extends Fish {
    @Override
    public void showName() {
        if (age <= 2) {
            System.out.print("オボコ ");
        } else {
            System.out.print("ボラ ");
        }
    }
}
```

さらに main メソッドを持つクラス FishTest を次のように定義する。

```
public class FishTest {
    public static void main(String[] args) {
        Carp c = new Carp("緋");
        c.color = "金";
        Fish[] fish = { c, new Seriola(), new Mugil() };
        int i, j;
        for (i = 0; i < 3; i++) {
            for (j = 0; j < 3; j++) {
                fish[j].growShowName();
            }
            System.out.println(); // 改行を出力
        }
```

```
13          for (i = 0; i < 998; i++) {
14              for (j = 0; j < 3; j++) {
15                  fish[j].grow();
16              }
17          }
18          for (j = 0; j < 3; j++) {
19              fish[j].showName();
20          }
21          System.out.println(); // 改行を出力
22      }
23  }
```

(1) Carp クラスのフィールド color の値はコンストラクターでのみ与えることができるものとし、一旦作成した後は、クラスの外からは直接アクセスできないようにしたい。（例えば、FishTest.java の 4 行目はコンパイル時にエラーになるようにしたい。） Carp.java の何行めをどのように変更すれば良いか？（行の左端の数字がファイルの中での行数を表す。）

(2) FishTest.java の 4 行目をコメントアウトし、このクラスの main メソッドを実行するとき、出力はどうなるか？（ただし、Fish クラスおよびそのサブクラスのインスタンスが新しく生成されたときのフィールド age の初期値は 0 である。）

なお、解答用紙に記入するとき、空白の有無や数は気にしなくて良い。直前の行と同じ内容は「〃」と省略して良い。

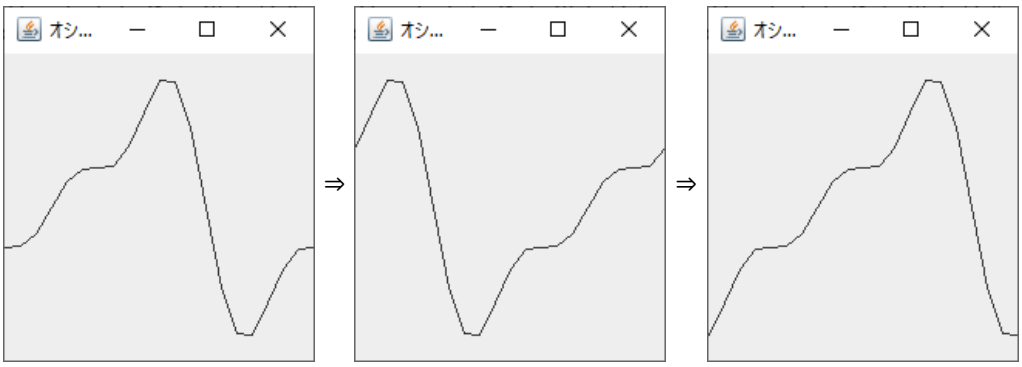
VII. 次の Oscilloscope は関数 f のグラフをオシロスコープ風のアニメーションとして表示する GUI アプリケーションである。“s” キーをタイプするとアニメーションを停止し、もう一度 “s” キーをタイプするとアニメーションを再開する。“a” (accelerate) キーをタイプすると、グラフの動きが速くなり 、“b” (break) キーをタイプすると、グラフの動きが遅くなる。 Oscilloscope クラスは JPanel を継承している。

ファイル名 Oscilloscope.java

```
1   import java.awt.*;
2   import java.awt.event.*;
3   import javax.swing.*;
4
5   public class Oscilloscope
6           (1)                     {
7       private volatile double t = 0, dt = 10;
8       private volatile Thread thread = null;
9
10      public Oscilloscope() {
11          setPreferredSize(new Dimension(200, 200));
12          thread = new Thread(this);
13          thread.start();
14          setFocusable(true);
15          addKeyListener(this);
16      }
17
18      private static double f(double x) {
19          return Math.sin(Math.PI * x / 100)
20               + Math.sin(2 * Math.PI * x / 100) / 2
21               + Math.sin(3 * Math.PI * x / 100) / 3;
22      }
```

```
23      @Override
24      public void paintComponent(Graphics g) {
25          super.paintComponent(g);
26          double a = 60;
27          for (int x0 = 0; x0 < 200; x0 += 10) {
28              double y0 = a * f(x0 + t);
29              int x1 = x0 + 10;
30              double y1 = a * f(x1 + t);
31              g.drawLine(x0, 100 + (int) y0, x1, 100 + (int) y1);
32          }
33      }
34
35      public void run() {
36          Thread thisThread = Thread.currentThread();
37          for (;                    (2)                    ; t += dt) {
38              repaint(); // paintComponent を間接的に呼出す
39              try {
40                  Thread.sleep(30); // 30 ミリ秒お休み
41              } catch (InterruptedException e) {}
42          }
43      }
44
45      public void keyTyped(KeyEvent e) {
46          char c = e.getKeyChar();
47          switch (c) {
48              case 's':
49                  if (thread == null) {
50                      thread = new Thread(this);
51                      thread.start();
52                  } else {
53                                (3)                ;
54                  }
55                  break;
56              case 'a':  dt += 2; break;
57              case 'b':  dt -= 2; break;
58          }
59      }
60      public void keyPressed(KeyEvent e) {}
61      public void keyReleased(KeyEvent e) {}
62      /* main は省略する */
63  }
```

実行例:



(1) ～ (3) の空欄を埋めてプログラムを完成せよ。

以下に参考のために授業配布プリントの LeftRightButton.java, LeftRightButton2.java, LeftRightButton3.java, Guruguru.java, Point.java, ColorPoint.java, Quadratic.kt, SequeceTest.kt のソースを掲載する。

ファイル LeftRightButton.java

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

public class LeftRightButton extends JPanel
                            implements ActionListener {
    private int x = 20;
    private JButton lBtn, rBtn;

    public LeftRightButton() {
        setPreferredSize(new Dimension(200, 70));
        lBtn = new JButton("Left");
        rBtn = new JButton("Right");
        lBtn.addActionListener(this);
        rBtn.addActionListener(this);
        setLayout(new FlowLayout());
        add(lBtn); add(rBtn);
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("HELLO WORLD!", x, 55);
    }

    public void actionPerformed(ActionEvent e) {
        Object source = e.getSource();
        if (source == lBtn) {       // lBtnが押された
            x -= 10;
        }
        else if (source == rBtn) { // rBtnが押された
            x += 10;
        }
        repaint();
    }

    public static void main(String[] args) {/* 省略 */}
}
```

ファイル LeftRightButton2.java

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

public class LeftRightButton2 extends JPanel {
    private int x = 20;

    public LeftRightButton2() {
        setPreferredSize(new Dimension(200, 70));
        JButton lBtn = new JButton("Left");
        JButton rBtn = new JButton("Right");
        lBtn.addActionListener(new LeftListener());
        rBtn.addActionListener(new RightListener());
        setLayout(new FlowLayout());
        add(lBtn); add(rBtn);
    }

    private class LeftListener
            implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            x -= 10; repaint();
```

```
        }
    }

    private class RightListener
            implements ActionListener {
        public void actionPerformed(ActionEvent e) {
            x += 10; repaint();
        }
    }

    @Override
    public void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawString("HELLO WORLD!", x, 55);
    }

    public static void main(String[] args) {/* 省略 */}
}
```

ファイル LeftRightButton3.java

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

public class LeftRightButton3 extends JPanel {
   private int x = 20;
   public LeftRightButton3() {
      setPreferredSize(new Dimension(200, 70));
      JButton lBtn = new JButton("Left");
      JButton rBtn = new JButton("Right");
      lBtn.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            x -= 10; repaint();
         }
      });
      rBtn.addActionListener(new ActionListener() {
         public void actionPerformed(ActionEvent e) {
            x += 10; repaint();
         }
      });
      add(lBtn); add(rBtn);
   }

   @Override
   public void paintComponent(Graphics g) {
      super.paintComponent(g);
      g.drawString("HELLO WORLD!", x, 55);
   }

   public static void main(String[] args) {/* 省略 */}
}
```

ファイル Guruguru.java

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

public class Guruguru extends JPanel
                      implements Runnable {
  private int r = 50;
  private volatile int x = 110, y = 70 ;
  private double theta = 0; // 角度
  private volatile Thread thread = null;

  public Guruguru() {
    setPreferredSize(new Dimension(200, 180));
    JButton startBtn = new JButton("start");
    startBtn.addActionListener(e -> startThread());
    JButton stopBtn = new JButton("stop");
```

```
    stopBtn.addActionListener(e -> stopThread());
    setLayout(new FlowLayout());
    add(startBtn); add(stopBtn);
    startThread();
  }

  private void startThread() {
    if (thread == null) {
      thread = new Thread(this);
      thread.start();
    }
  }

  private void stopThread() { thread = null; }

  @Override
  public void paintComponent(Graphics g) {
    // スーパークラスの paintComponent を呼び出す
    super.paintComponent(g);
    // 全体を背景色で塗りつぶす。
    g.drawString("Hello, World!", x, y);
  }

  public void run() {
    Thread thisThread = Thread.currentThread();
    for (; thread == thisThread; theta += 0.02) {
      x =  60 + (int)(r * Math.cos(theta));
      y = 100 - (int)(r * Math.sin(theta));
      repaint(); // paintComponent を間接的に呼出す
      try {
        Thread.sleep(30); // 30 ミリ秒お休み
      } catch (InterruptedException e) {}
    }
  }

  public static void main(String[] args) {/* 省略 */}
}
```

ファイル Point.java

```
public class Point {
    // フィールド(メンバー変数)
    public int x, y;

    // メソッド（メンバー関数）
    public void move(int dx, int dy) { x += dx; y += dy; }

    public double distance() {
        return Math.sqrt(x * x + y * y);
    }

    public void print() {
        System.out.printf("(%d, %d)", x, y);
    }

    public void info() {
        System.out.printf("x: %d, y: %d", x, y);
    }

    public void moveAndPrint(int dx, int dy) {
        print(); move(dx, dy); print();
    }

    // コンストラクター
    public Point(int x0, int y0) { x = x0; y = y0; }
}
```

ファイル ColorPoint.java

```java
public class ColorPoint extends Point {
    public String[] cs = {
        "black", "red", "green", "yellow",
        "blue", "magenta", "cyan", "white" };
    public String color;

    @Override
    public void print() {
        System.out.printf("<font color='%s'>", getColor()); // 色の指定
        System.out.printf("(%d, %d)", x, y); // super.print(); でも可
        System.out.print("</font>");                        // 色を戻す
    }

    public void setColor(String c) {
        int i;
        for (i = 0; i < cs.length; i++) {
            if (c.equals(cs[i])) {
                color = c; return;
            }
        } // 対応する色がなかったら何もしない。
    }

    public ColorPoint(int x, int y, String c) {
        super(x, y);
        setColor(c);
        if (color == null) color = "black";
    }

    public String getColor() { return color; }
}
```

ファイル Quadratic.kt

```kotlin
import kotlin.math.*

fun quadratic(a: Double, b: Double, c: Double): Pair<Double, Double> {
    val d = b * b - 4 * a * c
    val sq = sqrt(d)
    return Pair((-b + sq) / (2 * a), (-b - sq) / (2 * a))
}

fun main() {
    val a = 1.0; val b = -1.0; val c = -1.0
    val (x1, x2) = quadratic(a, b, c)
    println("方程式の解は、$x1、$x2 です。")
}
```

ファイル SequenceTest.kt

```kotlin
val nums = generateSequence(1) { x -> x + 3 }

val fibs = sequence {
    var a = 1; var b = 1
    while (true) {
        yield(a)
        val c = a
        a = b; b += c
    }
}

fun main() {
    println(nums.take(10).toList())
    println(fibs.take(10).toList())
}
```

（計算用紙）

オブジェクト指向言語・期末テスト解答用紙（2024 年 08 月 01 日）

学籍番号		氏名	

I. (3 × 2)

(1)		(2)	

II. (3 × 4)

(1)	
(2)	
(4)	
(5)	

III. (1, 1, 1, 1, 2, 3, 3)

(1)	
(2)	
(3)	
(4)	
(5)	
(6)	
(9)	

IV. (4 × 2)

(1)	
(2)	

V. (4 × 2)

(1)		(2)	

VI. (3, 9)

(1)	
(2)	

VII. (4 × 3)

(1)	
(2)	
(3)	

授業・テストの感想