オブジェクト指向言語・期末テスト問題用紙
（2025 年 07 月 31 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、問 Ⅰ～Ⅲ は中間テストの代替問題である。

   - 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 MouseTest.java, LeftRightButton.java, LeftRightButton3.java, LeftRightButton4.java, Guruguru.java, Point.java, ColorPoint.java, Quadratic.kt, SequenceTest.kt のソースを掲載する。（GUI アプリケーションの `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**。

`int` 型の変数 `x` の値が 3 のとき、次のうち、エラーとならず、「3 の 4 倍は 12 です。」と出力して改行する文はどれか? すべて選べ。

(A). `System.out.println(x & " の 4 倍は " & (x * 4) & " です。");`
(B). `System.out.println(x, " の 4 倍は ", x * 4, " です。");`
(C). `System.out.println(x + " の 4 倍は " + x * 4 + " です。");`
(D). `System.out.println("${x} の 4 倍は ${x * 4} です。");`
(E). `System.out.printf("%d の 4 倍は %d です。%n", x, x * 4);`

(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 は静的型付け言語であり、コンパイル時に型の整合性がチェックされる。
(B). Java のコンパイラーは、C 言語のソースファイルもコンパイルできる必要がある。
(C). Java の言語仕様は JavaScript の言語仕様の上位互換になるように定められている。
(D). Java の中間言語の形式はコンパイル時の機種に依存しているため、他の機種で実行することはできない。
(E). Java の配列は範囲外をアクセスしてもエラーや例外にならず、ただ変な動作が起こる。
(F). Java のクラスが実装 (implement) できるインタフェースは複数あっても良い。

II. 次の枠内の文章は `java.util.Scanner` クラスの `hasNextLine`, `nextLine`, `hasNext`, `next`, `close` メソッド、 `java.lang.Double` クラスの `parseDouble` メソッド、 `java.lang.String` クラスの `join` メソッドの Java API Reference からの抜粋である。（問題を解くのに関係ない部分は割愛し、また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.util`

クラス Scanner

```
public class Scanner extends Object implements Iterator,
Closeable
```

正規表現を使用してプリミティブ型および文字列の構文解析が可能な、単純なテキスト・スキャナーです。 `Scanner` は、区切り文字のパターンを使用して入力をトークンに分割します。デフォルトでは区切り文字は空白文字です。 結果のトークンは、様々な `next` メソッドを使用して様々なタイプの値に変換できます。

コンストラクターの詳細

Scanner

```
public Scanner(InputStream source)
```

指定された入力ストリームからスキャンされる値を生成する新しい `Scanner` を構築します。

パラメータ:
`source` - スキャン対象の入力ストリーム

Scanner

```
public Scanner(String source)
```

指定された文字列からスキャンされる値を生成する新しい `Scanner` を構築します。

パラメータ:
`source` - スキャンする文字列

メソッドの詳細

close

```
public void close()
```

このスキャナをクローズします。

hasNext

```
public boolean hasNext()
```

このスキャナが入力内にまだトークンを保持する場合は `true` を返します。

戻り値:
このスキャナがまだトークンを保持する場合にのみ `true`

next

```
public String next()
```

このスキャナから次の完全なトークンを見つけて返します。

戻り値:
次のトークン

hasNextLine

```
public boolean hasNextLine()
```

このスキャナの入力にまだ行がある場合は `true` を返します。

戻り値:
残りの入力に行セパレータがある場合、または入力に他の残りの文字がある場合、`true`

nextLine

```
public String nextLine()
```

スキャナを現在行の先に進めて、スキップした入力を返します。 このメソッドは、最後の行区切り文字を除く、現在行の残りを返します。 位置は、次の行の最初に設定されます。

戻り値:
スキップされた行

パッケージ `java.lang`

クラス Double

```
public final class Double extends Number implements
Comparable<Double>, Constable, ConstantDesc
```

Double クラスは、プリミティブ型 doubleの値をオブジェクトにラップします。Double 型のオブジェクトには、型が double の単一フィールドが含まれます。

さらにこのクラスは、double を String に、 String を double に変換する各種メソッドや、double の処理時に役立つ定数およびメソッドも提供します。

メソッドの詳細

parseDouble

```
public static double parseDouble(String s) throws
NumberFormatException
```

指定された String が表す値に初期化された新しい double 値を返します。

パラメーター:
　　s - 解析される文字列。
戻り値:
　　文字列引数で表される double 値。
throws:
　　NullPointerException - 文字列が null の場合
　　NumberFormatException - 文字列が解析可能な double を含まない場合

パッケージ java.lang

クラス String

```
public class String
```

メソッドの詳細

join

```
public static String join(String delim, ArrayList<String> elem)
```

指定された delim のコピーを使用して結合された ArrayList 要素のコピーからなる新しい String を返します。

パラメータ:
　　delim - 各要素を区切る区切り文字
　　elem - 結合する要素。
戻り値:
　　delim で区切られた elem からなる新しい String
throws:
　　NullPointerException - delim または elem が null である場合

下に示す SimpleCalc クラスは、これらのメソッドを使って、標準入力から行ごとに空白で区切られたいくつかの実数値を読み取り、その合計を計算して結果を出力する。空行が入力された時点で終了する。

このプログラムの実行例は次のようになる。 *斜字体*の部分がユーザーからの入力でそれ以外がプログラムの出力である。改行は ↲ で明示されている。

```
1 3 10 2↲
1 + 3 + 10 + 2 = 16.0↲
2.3 4.56 1.86 -0.2↲
2.3 + 4.56 + 1.86 + -0.2 = 8.52↲
↲
```

次のソースプログラムを見て、以下の問に答えよ。

ファイル名 SimpleCalc.java

```
 1 import java.util.Scanner;
 2 import java.util.ArrayList;
 3
 4 public class SimpleCalc {
 5    public static void main(String[] args) {
 6      Scanner scanner = new Scanner(System.in);
 7      while (          (1)          ) {
 8         String line =          (2)          ;
 9         Scanner lineScanner = new Scanner(line);
10         ArrayList<String> tokens = new ArrayList<>();
11         double sum = 0;
12         while (          (3)          ) {
13            String token =          (4)          ;
14            tokens.add(token);
15            try {
16               double number =          (5)          ;
17               sum += number;
18            } catch (NumberFormatException e) {}
19         }
20         (6)          ;
21         if (tokens.isEmpty()) break;
22         System.out.print(          (7)          );
23         System.out.println(" = " + sum);
24       }
25       (8)          ;
26    }
27 }
```

(1) の空欄には、scanner に次の行が存在するかどうかを判定する式が入る。この式を答えよ。

(2) の空欄には、scanner から現在行を読み取る式が入る。この式を答えよ。

(3) の空欄には、lineScanner に次のトークンが存在するかどうかを判定する式が入る。この式を答えよ。

(4) の空欄には、lineScanner から次のトークンを読み取る式が入る。この式を答えよ。

(5) の空欄に、token を double 型に変換する式が入る。この式を答えよ。

(6) の空欄には、lineScanner を閉じる式が入る。この式を答えよ。

(7) の空欄には、tokens の要素を区切り文字 " + " で結合した文字列を表す式が入る。この式を答えよ。

(8) の空欄には、scanner を閉じる式が入る。この式を答えよ。

III. 次の ToyTimer クラスは簡易タイマーの GUI アプリケーションである。マウスをクリックした位置に時計の針が表示され、1 秒ごとに針が 6 度反時計回りに回転する。また、マウスをクリックすると針の位置がその位置に移動する。

また ToyTimer クラスは JPanel を継承している。

ファイル名 ToyTimer.java

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  public class ToyTimer
6          (1)                                  {
7      private int seconds = 0;
8
9      public ToyTimer() {
10         setPreferredSize(new Dimension(200, 200));
11         addMouseListener(this);
12         Timer timer = new Timer(1000, this);
13         timer.start();
14     }
15
16     @Override
17     public void paintComponent(Graphics g) {
18         super.paintComponent(g);
19         g.setColor(Color.BLUE);
20         g.drawOval(20, 20, 160, 160);
21         double theta = seconds * Math.PI / 30;
22         // 真上が 0 度で時計回り方向が正
23         g.drawLine(100, 100,
24                    100 + (int)(80 * Math.sin(theta)),
25                    100 - (int)(80 * Math.cos(theta)));
26         g.drawString("Seconds: " + seconds, 10, 20);
27     }
28
29     public void actionPerformed(ActionEvent e) {
30         if (seconds > 0) seconds--;
31         repaint();
32     }
33
34     public void mouseClicked(MouseEvent e) {
35         int x = e.getX();
36         int y = e.getY();
37         // 真上が 0 度で時計回り方向が正
38         double theta = Math.atan2(x - 100, 100 - y);
39         seconds = (int)(theta / Math.PI * 30);
40         if (seconds < 0) seconds += 60;
41         repaint();
42     }
43
44     public void mousePressed(MouseEvent e) {}
45     public void mouseReleased(MouseEvent e) {}
46     public void mouseEntered(MouseEvent e) {}
47     public void mouseExited(MouseEvent e) {}
48     /* main は省略する */
49 }
```

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

パッケージ javax.swing

クラス Timer

```
public class Timer extends Object implements Serializable
```

指定された間隔で、1つ以上の ActionEvent をトリガーします。 たとえば、アニメーション・オブジェクトは、フレームを描画するトリガーとして Timerを使用します。

タイマーの設定には、Timer オブジェクトの生成、オブジェクトへの 1 つ以上のアクション・リスナーの登録、および start メソッドを使用したタイマーの起動が含まれます。 たとえば、次に示すコードでは、Timer コンストラクタへの最初の引数によって指定されたように、アクション・イベントが 1 秒間に 1 回生成されて起動されます。 Timer コンストラクタへの 2 番目の引数では、タイマーのアクション・イベントを受信するリスナーを指定しています。

```
int delay = 1000; //milliseconds
ActionListener taskPerformer = new ActionListener() {
    public void actionPerformed(ActionEvent evt) {
        //...Perform a task...
    }
};
new Timer(delay, taskPerformer).start();
```

Timer を構築する際は、遅延パラメータと ActionListener の両方を指定します。遅延パラメータは、初期遅延と、イベントがトリガーされる間の遅延の両方の設定に使用します。単位はミリ秒です。 タイマーは、開始後、登録されたリスナーに対する最初の ActionEvent がトリガーされるまでの初期遅延を待機します。 この最初のイベントの発生後は、タイマーが停止するまで、イベント間遅延が経過するたびにイベントをトリガーする処理を継続します。

コンストラクターの詳細

Timer

```
public Timer(int delay, ActionListener listener)
```

Timer を作成し、初期遅延とイベント間遅延を delay ミリ秒に初期化します。listener が null 以外の場合は、タイマーのアクション・リスナーとして登録されます。

パラメータ:
delay - 初期遅延とイベント間遅延を表す値(ミリ秒)
listener - 初期のリスナー。null の場合もあり

メソッドの詳細

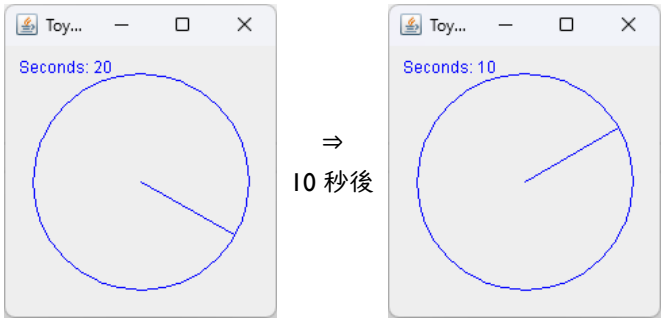
start

```
public void start()
```

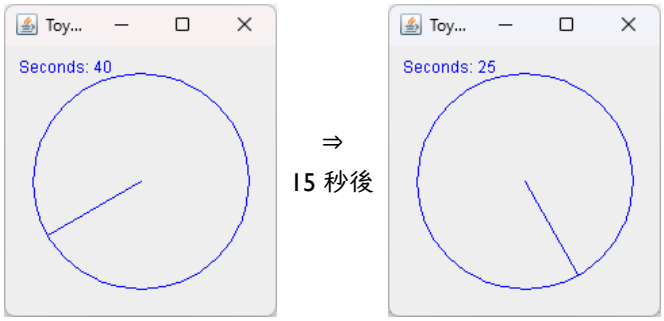
Timer を起動し、リスナーへのアクション・イベントの送信を開始します。

実行例（次ページ）：

右下のほうをクリック



左下のほうをクリック



(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
3  public class ToyTimer2
4          (2)                          {
5      private int seconds = 0;
6
7      public ToyTimer2() {
8          setPreferredSize(new Dimension(200, 200));
9          addMouseListener(          (3)                );
10         Timer timer = new Timer(1000,
11                 (4)                );
12         timer.start();
13     }
14
15     /* paintComponent, main はほぼ同じなので省略する */
16 }
```

(2) の空欄を埋めよ。なお ToyTimer2 クラスも JPanel を継承している。

(3) には MouseAdapter を使用した匿名クラスの式が入る。(3)の空欄を埋めよ。なお、MouseAdapter は次のような定義済みのクラスである。（一部、問題と関係ないところは省略している。）

```
1  package java.awt.event;
2  public class MouseAdapter implements MouseListener {
3      public void mouseClicked(MouseEvent e) {}
4      public void mousePressed(MouseEvent e) {}
5      public void mouseReleased(MouseEvent e) {}
6      public void mouseEntered(MouseEvent e) {}
7      public void mouseExited(MouseEvent e) {}
8  }
```

ただし、ToyTimer.java と同じ部分は、`/* 元の AB ～ YZ 行目と同じ */` のように省略して書いて良い。

(4) の空欄に入るラムダ式を答えよ。ただし、ToyTimer.java と同じ部分は、`/* 元の AB ～ YZ 行目と同じ */` のように省略して書いて良い。

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。また `toList()` メソッドは、出力するためにシーケンスをリストに変換する。

Kotlin はリストを `[` ～ `]` を使ってコンマ区切りで出力する。例えば、`print(list(2, 3, 5))` は `[2, 3, 5]` と出力する。

(1)

```kotlin
val r2d: Sequence<Int> = sequence {
    var x = 1
    var y = 0
    while (true) {
        yield(x)
        val tempX = -y
        y = x
        x = tempX
    }
}

fun main() {
    println(r2d.take(8).toList())
}
```

(2)

```kotlin
val r3d: Sequence<Int> = sequence {
    var x = 3
    var y = 0
    var z = 0
    while (true) {
        yield(x)
        val tempX = (x * 2 + y * 2 - z) / 3
        val tempY = (y * 2 + z * 2 - x) / 3
        z         = (z * 2 + x * 2 - y) / 3
        x = tempX
        y = tempY
    }
}

fun main() {
    println(r3d.take(8).toList())
}
```

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) 次の選択肢の中でオブジェクト指向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). Fortran (C). Haskell (D). Prolog (E). Smalltalk

(2) 次の選択肢の中で、動的型付けに分類される言語はどれか? 二つ選べ。

(A). Java (B). JavaScript (C). C (D). C++ (E). Python

VI. 物体の運動をシミュレーションするために、いくつかのパッケージにいくつかのクラスを作成する。次に定義されるクラス `basic.P` を継承して、

ファイル名 basic/P.java

```java
package basic;

public class P {
    public int x;
    public String name;

    public P(String n, int x0) {
        x = x0;
        name = n;
    }
    public void update() {}
    public void info() {
        System.out.print(name + ": " + x);
    }
}
```

3 つのサブクラス `basic.U`, `advanced.S`, `advanced.A` を定義する。

ファイル名 basic/U.java

```java
package basic;

public class U extends P {
    public int v;

    public U(int x0, int v0) {
        super("U", x0);
        v = v0;
    }

    @Override
    public void update() {
        x += v;
    }
}
```

ファイル名 advanced/S.java

```java
package advanced;
import basic.*;

public class S extends P {
    private int r;

    public S(int x0, int r0) {
        super("S", x0);
        r = r0;
    }

    @Override
    public void update() {
        x *= r;
    }
}
```

ファイル名 advanced/A.java

```java
package advanced;
import basic.*;

public class A extends U {
    private int a;

    public A(int x0, int v0, int a0) {
        super(x0, v0);
        name = "A";
        a = a0;
    }

    @Override
    public void update() {
        super.update();
        v += a;
    }
}
```

さらに `main` メソッドを持つクラス `Main` を匿名パッケージに次のように定義する。

```java
import advanced.*;
import basic.*;
public class Main {
   public static void main(String[] args) {
      P[] ps = { new U(0, 2), new S(1, -2), new A(2, 0, 1)};
      ps[0].name = "X";
      for (int i = 0; i < 4; i++) {
         for (P p : ps) {
            p.info();
            p.update();
            System.out.print(", ");
         }
         System.out.println();
      }
   }
}
```

(1) basic.P クラスのフィールド name の値は、一旦作成した後は、package 外のクラスからは直接アクセスできないようにしたい。（例えば、Main.java の 6 行目はコンパイル時にエラーになるようにしたい。）ただし、advanced/A.java の 9 行目のように、サブクラスからは package が異なってもアクセスできるようにしたい。 basic/P.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。）なお、basic.P クラスのもう一つのフィールド x や basic.U クラスのフィールド v も同じように変更するべきだが、この問ではその点は触れなくてよい。

(2) Main.java の 6 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？

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。

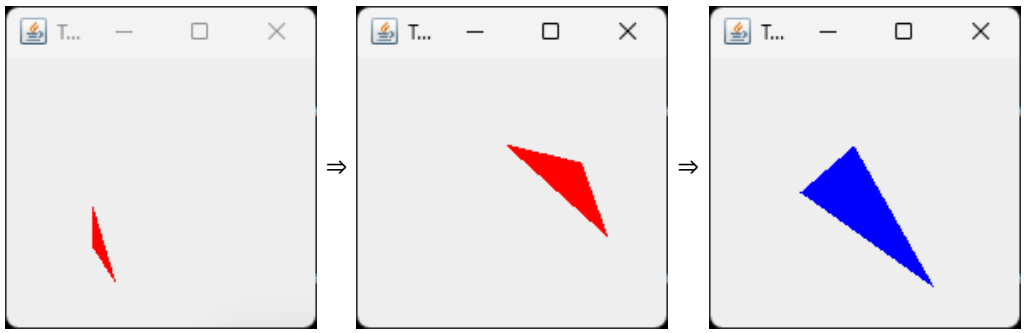
VII. 次の TriangleAnime は 3 つの速さの異なる頂点が円周上を動く三角形のアニメーションを表示する GUI アプリケーションである。マウスをクリックすると、アニメーションが停止し、もう一度クリックすると再開する。

ファイル名 TriangleAnime.java

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  public class TriangleAnime extends JPanel
6          (1)                                 {
7      private int[] xs = {0, 0, 0};
8      private int[] ys = {0, 0, 0};
9      private double t = 0;
10     private volatile Thread thread;
11
12     public TriangleAnime() {
13         setPreferredSize(new Dimension(400, 400));
14         addMouseListener(this);
15         startThread();
16     }
17
18     private void startThread() {
19         thread = new Thread(this);
20         thread.start();
21     }
22
23     public void mouseClicked(MouseEvent e) {
24         if (thread != null) {
25             thread = null;
26         } else {
27             startThread();
28         }
29     }
30
31     public void mousePressed(MouseEvent e) {}
32     public void mouseReleased(MouseEvent e) {}
33     public void mouseEntered(MouseEvent e) {}
34     public void mouseExited(MouseEvent e) {}
35
```

```
36      @Override
37      protected void paintComponent(Graphics g) {
38          super.paintComponent(g);
39          if ((xs[2] - xs[0]) * (ys[1] - ys[0])
40                  - (ys[2] - ys[0]) * (xs[1] - xs[0]) < 0) {
41              g.setColor(Color.RED);
42          } else {
43              g.setColor(Color.BLUE);
44          }
45          g.fillPolygon(xs, ys, 3);
46      }
47
48      public void run() {
49          Thread currentThread = Thread.currentThread();
50          while (                    (2)                    ) {
51              xs[0] = (int) (100 + 50 * Math.cos(9 * t));
52              ys[0] = (int) (100 + 50 * Math.sin(9 * t));
53              xs[1] = (int) (100 + 50 * Math.cos(11 * t));
54              ys[1] = (int) (100 + 50 * Math.sin(11 * t));
55              xs[2] = (int) (100 + 50 * Math.cos(13 * t));
56              ys[2] = (int) (100 + 50 * Math.sin(13 * t));
57              t += 0.005;
58                          (3)                          ;
59              try {
60                  Thread.sleep(25);
61              } catch (InterruptedException e) {
62                  Thread.currentThread().interrupt();
63              }
64          }
65      }
66      /* main は省略する */
67  }
```

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, LeftRightButton3.java, LeftRightButton4.java, Guruguru.java, Point.java, ColorPoint.java, Quadratic.kt, SequeceTest.kt のソースを掲載する。

ファイル MouseTest.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;   /* 1 */

public class MouseTest extends JPanel
             implements MouseListener /* 2 */ {
    private int x = 50, y = 20;

    public MouseTest() {
        setPreferredSize(new Dimension(150, 150));
        addMouseListener(this);   /* 3 */
    }

    /* 4 */
    public void mouseClicked(MouseEvent e) {
        x = e.getX();
        y = e.getY();
        repaint();
        return;
    }

    public void mousePressed(MouseEvent e) {}  /* 5 */
    public void mouseReleased(MouseEvent e) {} /* 5 */
    public void mouseEntered(MouseEvent e) {}  /* 5 */
    public void mouseExited(MouseEvent e) {}   /* 5 */

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

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

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 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 LeftRightButton4.java

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

public class LeftRightButton4 extends JPanel {
    private int x = 20;

    public LeftRightButton4() {
        setPreferredSize(new Dimension(200, 70));
        JButton lBtn = new JButton("Left");
        JButton rBtn = new JButton("Right");
        lBtn.addActionListener(e -> {
            x -= 10;
            repaint();
        });
        rBtn.addActionListener(e -> {
            x += 10;
            repaint();
        });
        setLayout(new FlowLayout());
        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);
    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

```java
public class Point {
  public int x;
  public int y;

  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();
  }

  public Point(int x0, int y0) {
    x = x0; y = y0;
  }
}
```

ファイル ColorPoint.java

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

```
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

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

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

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

（計算用紙）

（計算用紙）

オブジェクト指向言語・期末テスト解答用紙（2025 年 07 月 31 日）

学籍番号		氏名	

I. (2 × 3)

(1)		(2)	

II. (8 × 2)

(1)	
(2)	
(3)	
(4)	
(5)	
(6)	
(7)	
(8)	

III. (1 + 1 + 3 + 3)

(1)	
(2)	
(3)	
(4)	

IV. (2 × 4)

(1)	
(2)	

V. (2 × 4)

(1)		(2)	

VI. (3 + 9)

(1)	
(2)	

VII. (3 × 4)

(1)	
(2)	
(3)	

授業・テストの感想