オブジェクト指向言語・期末テスト問題用紙
（2026 年 07 月 30 日・ 08:50 ～ 10:20）

解答上、その他の注意事項

1. 問題は、問 Ⅰ～Ⅵ まである。うち、問 Ⅰ～Ⅱ 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。）
   - 中間テストで 7 割以上あった場合は、問 Ⅰ～Ⅱ は解かなくてよい。
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～Ⅱ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` Ⓐ`E` `ActionEvent` Ⓚ `KeyListener`
Ⓐ`K` `addKeyListener` Ⓚ`E` `KeyEvent` Ⓜ `MouseListener` Ⓐ`M` `addMouseListener`
Ⓜ`E` `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 KeyTest.java, LeftRightButton.java, LeftRightButton4.java, Guruguru.java, GuruguruTh.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。また、改行の位置などを変更している可能性はある。）

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**。

Java のクラス名として文法上許されない（コンパイル時にエラーになる）ものをすべて選べ。

(A). `Y20260730`　(B). `2026Test`　(C). `MyClass`　(D). `Hey'Jude`　(E). `Plag_n_`

(2) 次の Java に関する文章のうち、正しいものをすべて選べ。

(A). Java の配列は範囲外をアクセスすると実行時に例外が発生する。
(F). Java のカプセル化は、メソッドやフィールドの名前の慣習として表現され、コンパイル時には違反をチェックできない。
(B). C++ 言語のコンパイラーは、Java 言語のソースもコンパイルできる必要がある。
(C). Java 言語をブラウザーで実行するために、ECMA で仕様を定めたのが ECMAScript、いわゆる JavaScript である。
(D). Java のコンパイラーはプログラムを実行する機種ごとに異なる中間言語を生成する。
(E). Java のクラスは、（直接の）スーパークラスを、たかだか一つしか指定することができない。

II. 次の枠内の文章は `java.time.LocalDate` クラスの `now`, `of`, `getYear`, `getMonthValue`, `getDayOfMonth`, `plusDays`, `toEpochDay`, `toString` 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.time`

クラス LocalDate

```
public final class LocalDate extends Object
```

`2007-12-03` のようなローカルな（タイムゾーンのない）日付です。

`LocalDate`は、日付 (年-月-日として表されることが多い) を表すイミュータブルな日付オブジェクトです。他の日付フィールド (その年の何日めか、曜日、その年の何週めかなど) にもアクセスできます。たとえば「2007年10月2日」という値を `LocalDate` に格納できます。

メソッドの詳細

now

```
public static LocalDate now()
```

デフォルトのタイムゾーンのシステム・クロックから現在の日付を取得します。

戻り値:
　　システムクロックとデフォルトのタイムゾーンを使用した現在の日付、null ではない。

of

```
public static LocalDate of(int year, int month, int dayOfMonth)
```

年、月、および日から LocalDate のインスタンスを取得します。 これは、指定された年、月、および日を使って LocalDate を返します。日は年と月に対して有効である必要があり、そうでない場合は、例外がスローされます。

パラメーター:
year - 表される年。
month - 表される月 (1 ～ 12)。
dayOfMonth - 表される日 (1 ～ 31)。
戻り値:
null ではないローカルな日付
スロー:
DateTimeException - いずれかのフィールドの値が範囲外である場合、または日が月に対して無効である場合

getYear

```
public int getYear()
```

"年"フィールドを取得します。

このメソッドは、年を表す int 値を返します。

戻り値:
年。

getMonthValue

```
public int getMonthValue()
```

1 から 12 の範囲の"月"フィールドを取得します。

このメソッドは、月を 1 ～ 12 の int として返します。

戻り値:
月 (1 ～ 12)。

getDayOfMonth

```
public int getDayOfMonth()
```

"日" (day-of-month) フィールドを取得します。

戻り値:
日 (1 ～ 31)。

plusDays

```
public LocalDate plusDays(long daysToAdd)
```

指定された日数を加えた新しい LocalDate を返します。

このメソッドは、指定された量を日フィールドに加算し、結果が有効であるように、必要に応じて月および年フィールドを増分します。たとえば、2008-12-31に1日を加算すると、2009-01-01という結果になります。

パラメーター:
`daysToAdd` - 加算する日数。負の値の場合もある。
戻り値:
指定された日数を加えた新しい `LocalDate`、null ではない。

toEpochDay

```
public long toEpochDay()
```

この日付をエポック日に変換します。

エポック日は日の単純な増分カウントで、1970-1-1 からの日数（1970-1-1のエポック日は 0）です。

引用者注: 例えば 1972-1-1のエポック日は (365 + 365 =) 730、 1973-1-1 のエポック日は (365 + 365 + 366 =) 1096 です。

toString

```
public String toString()
```

この日付を `YYYY-MM-DD` 形式の文字列に変換します。

戻り値:
この日付の文字列表現。null ではない。

下に示す `DateCalculator` クラスは、これらのメソッドを使って、プログラム起動した日の X 日後が何年何月何日かを計算し、また、逆に、入力した年・月・日がプログラム起動した日から何日後かを計算する GUI アプリケーションである。なお、エラー処理は行っていないので、ユーザーが不正な値を入力した場合は、例外が発生してプログラムが終了する。

また `DateCalculator` は `JPanel` を継承する。

実行例:

プログラム起動時の画面

5000 日後を入力した画面

2099-12-31 を入力した画面

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 DateCalculator.java

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

class DateCalculator
        (1)                    {
  LocalDate today =                    (2)                    ;
  JTextField periodTF, yearTF, monthTF, dateTF;

  public DateCalculator() {
    add(new JLabel(                    (3)                    ));
    add(new JLabel("の"));
    periodTF = new JTextField("0", 5);
    add(periodTF);
    add(new JLabel("日後は"));
    yearTF = new JTextField(5);
    add(yearTF);
    add(new JLabel("-"));
    monthTF = new JTextField(2);
    add(monthTF);
    add(new JLabel("-"));
    dateTF = new JTextField(2);
    add(dateTF);
    add(new JLabel("です"));
    setYMD(today);
    periodTF.addActionListener(this);
    yearTF.addActionListener(this);
    monthTF.addActionListener(this);
    dateTF.addActionListener(this);
  }

  void setYMD(LocalDate date) {
    yearTF.setText(""  +                    (4)                    );
    monthTF.setText("" +                    (5)                    );
    dateTF.setText(""  +                    (6)                    );
  }

  public void actionPerformed(ActionEvent e) {
    Object source = e.getSource();
    if (source == periodTF) {
      int period = Integer.parseInt(periodTF.getText());
      LocalDate newDate =                    (7)                    ;
      setYMD(newDate);
    } else {
      int year  = Integer.parseInt(yearTF.getText());
      int month = Integer.parseInt(monthTF.getText());
      int date  = Integer.parseInt(dateTF.getText());
      LocalDate newDate =                    (8)                    ;
      long diff =                    (9)                    ;
      periodTF.setText("" + diff);
    }
  }
}
// main は掲載を省略する
```

(1) の空欄を埋めて DateCalculator クラスのスーパークラスや実装するインタフェースを指定せよ。

(2) の空欄には、現在の日付を取得する式が入る。この式を答えよ。

(3) の空欄には today から YYYY-MM-DD 形式の文字列を取得する式が入る。この式を答えよ。

(4) の空欄には date から年を取得する式が入る。この式を答えよ。

(5) の空欄には date から 1 ～ 12 の月を取得する式が入る。この式を答えよ。

(6) の空欄には date から 1 ～ 31 の日を取得する式が入る。この式を答えよ。

(7) の空欄には today を基準に period 日後の日付を返す式が入る。この式を答えよ。

(8) の空欄には year 年 month 月 date 日を表す LocalDate のインスタンスを返す式が入る。この式を答えよ。

(9) の空欄には newDate と today のエポック日を使って、 newDate が today から何日後かを計算する式が入る。この式を答えよ。

このプログラムをラムダ式を用いて次の DateCalculatorL クラスのように書き換える。

DateCalculatorL も JPanel を継承する。

ファイル名 DateCalculatorL.java

```
1  // import は DateCalculator.java と同じなので省略する
2
3  class DateCalculatorL
4          (10)          {
5    // フィールドは DateCalculator.java と同じなので省略する
6
7    public DateCalculatorL() {
8      // DateCalculator.java の 11-25 行目と同じなので省略する
9      periodTF.addActionListener(          (11)          );
10     ActionListener listener =          (12)          ;
11     yearTF.addActionListener(listener);
12     monthTF.addActionListener(listener);
13     dateTF.addActionListener(listener);
14   }
15
16   // setYMD メソッドは DateCalculator.java と同じなので省略する
17 }
18 // main は掲載を省略する
```

(10) の空欄を埋めて DateCalculatorL クラスのスーパークラスや実装するインタフェースを指定せよ。

(11) の空欄に入るラムダ式を答えよ。ただし、DateCalculator.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。

(12) の空欄に入るラムダ式を答えよ。ただし、DateCalculator.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。

III. 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
fun hanoi(n: Int) : Sequence<Int>  = sequence {
    if (n > 0) {
        for (i in hanoi(n - 1)) {
            yield(i)
        }
        yield(n)
        for (i in hanoi(n - 1)) {
            yield(i)
        }
    }
}

fun main() {
    print(hanoi(4).take(10).toList())
}
```

(2)

```kotlin
fun collatz(n: Int): Sequence<Int> = sequence {
    var x = n
    while (x != 1) {
        yield(x)
        if (x % 2 == 0) {
            x = x / 2
        } else {
            x = 3 * x + 1
        }
    }
    yield(1)
}

fun main() {
    print(collatz(11).take(10).toList())
}
```

IV. 次の (1) ～ (2) の多肢選択問題に答えよ。

(1) 次の選択肢の中で論理型言語に分類される言語はどれか? 最もふさわしいものを一つ選べ。

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

(2) 次の選択肢の中で、最も新しく公開された言語はどれか? 一つ選べ。

(A). C (B). C++ (C). C# (D). Java (E). COBOL

V. 育成ゲームのプロトタイプ実装として、いくつかのパッケージにいくつかのクラスを作成する。まず、次に定義されるクラス `base.Animal` を継承して、

ファイル名 base/Animal.java

```java
package base;

public class Animal {
    protected int age;
    public Animal() { age = 0; }
    public void say() {
        System.out.print("...");
        age++;
    }
}
```

3 つのサブクラス `animal.Chicken`, `animal.Gundi`, `animal.Quokka` を定義する。

ファイル名 animal/Chicken.java

```java
package animal;
import base.*;

public class Chicken extends Animal {
    public boolean female; // false -- 雄, true -- 雌
    public Chicken(boolean f) { super(); female = f; }
    @Override
    public void say() {
        if (age++ < 1) {
            System.out.print("Piyo ");
        } else if (female) {
            System.out.print("Kokko ");
        } else {
            System.out.print("Kokekokkooo ");
        }
    }
}
```

ファイル名 animal/Gundi.java

```java
package animal;
import base.*;

public class Gundi extends Animal {
    public Gundi() { super(); }
    @Override
    public void say() {
        if (age++ < 1) {
            System.out.print("Chruru ");
        } else {
            System.out.print("ChiChi ");
        }
    }
}
```

ファイル名 animal/Quokka.java

```java
package animal;
import base.*;

public class Quokka extends Animal {
    public Quokka() { super(); }
    @Override
    public void say() {
        System.out.print("Qyuu " );
        age++;
    }
}
```

さらに `main` メソッドを匿名パッケージに次のように定義する。

ファイル名 Main.java

```java
import animal.*;
import base.*;

void main() {
    Chicken c = new Chicken(false);
    c.female = true;
    Animal[] animals = { new Gundi(), new Quokka(), c };

    for (int i = 0; i < 3; i++) {
        for (Animal a: animals) {
            a.say();
        }
        System.out.println();
    }
}
```

(1) `animal.Chicken` クラスのフィールド `female` の値は、一旦作成した後は他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 `c.female = true;` はコンパイル時にエラーになるようにしたい。） animal/Chicken.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) Main.java の 6 行目 `c.female = true;` をコメントアウトし、この `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。

VI. 次の SankakuThread クラスは円が直角三角形上を動くアニメーションを表示する GUI アプリケーションである。スペースキーをタイプすると、アニメーションが停止し、もう一度タイプすると再開する。

ファイル名 SankakuThread.java

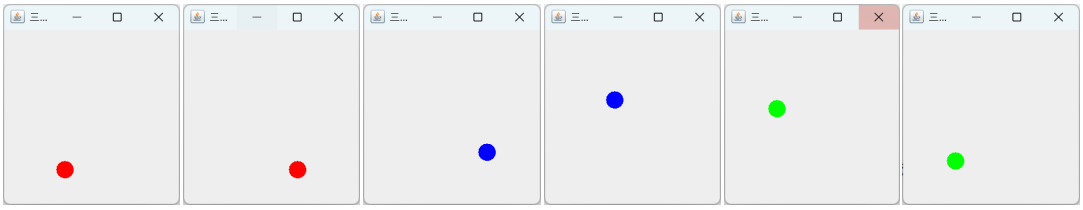
```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class SankakuThread extends JPanel
                    (1)                    {
    private volatile Thread thread;
    private volatile int x = 50, y = 150;
    private volatile Color color = Color.RED;
    private int i = 0;
    public SankakuThread() {
        setPreferredSize(new Dimension(200, 200));
        thread = new Thread(this); thread.start();
        setFocusable(true); addKeyListener(this);
    }

    public void run() {
        Thread myThread = Thread.currentThread();
        while (true) {
            color = Color.RED;
            for ( ; i < 10; i++) {
                if (                    (2)                    )
                    return;
                x = 50 + 10 * i;  y = 150;
                repaint();
                try { Thread.sleep(50); }
                catch (InterruptedException e) {}
            }
            color = Color.BLUE;
            for ( ; i < 20; i++) {
                if (                    (2)                    )
                    return;
                x = 150 - 10 * (i - 10);  y = x;
                repaint();
                try { Thread.sleep(50); }
                catch (InterruptedException e) {}
            }
            color = Color.GREEN;
            for ( ; i < 30; i++) {
                if (                    (2)                    )
                    return;
                x = 50;  y = 50 + 10 * (i - 20);
                repaint();
                try { Thread.sleep(50); }
                catch (InterruptedException e) {}
            }
            i = 0;
        }
    }
```

```
51      @Override
52      public void paintComponent(Graphics g) {
53          super.paintComponent(g);
54          g.setColor(color);
55          g.fillOval(x, y, 20, 20);
56      }
57
58      public void keyTyped(KeyEvent e) {
59          int key = e.getKeyChar();
60          if (key == ' ') {
61              if (thread == null) {
62                  thread = new Thread(this);
63                  thread.start();
64              } else {
65                  thread = null;
66              }
67          }
68      }
69      public void keyPressed(KeyEvent e) {}
70      public void keyReleased(KeyEvent e) {}
71  }
72  /* main は掲載を省略する */
```

実行例:



●の色は赤　●の色は赤　●の色は青　●の色は青　●の色は緑　●の色は緑

(1) の空欄を埋めて SankakuThread クラスが実装するインタフェースを指定せよ。

(2) の空欄（3ヵ所の内容は共通）を埋めて、スペースキーをタイプしたときにスレッドの実行を停止するようにせよ。

このプログラムを javax.swing.Timer を使って、 SankakuThread とほぼ同じ 50 msec ごとに 1 フレーム更新するプログラム SankakuTimer に書き換える。

```
1   // import は SankakuThread.java と同じなので省略する
2
3   class SankakuTimer extends JPanel
4           (3)                          {
5       private Timer timer;
6       private int x = 50, y = 150;
7       private Color color = Color.RED;
8       private int i = 0;
9
10      public SankakuTimer() {
11          setPreferredSize(new Dimension(200, 200));
12          setFocusable(true); addKeyListener(this);
13          timer = new Timer(50, this); timer.start();
14      }
15      // paintComponent メソッドは同じなので省略する
```

```
16
17      public void keyTyped(KeyEvent e) {
18          int key = e.getKeyChar();
19          if (key == ' ') {
20              if (timer.isRunning()) {
21                  timer.stop();
22              } else {
23                  timer.start();
24              }
25          }
26      }
27      public void keyPressed(KeyEvent e) {}
28      public void keyReleased(KeyEvent e) {}
29
30      public void actionPerformed(ActionEvent e) {
31                          (4)
32          repaint();
33          i++;
34          if (i >= 30) {
35              i = 0;
36          }
37      }
38  }
39  /* main は掲載を省略する */
```

(3) の空欄を埋めて SankakuTimer クラスが実装するインタフェースを指定せよ。

(4) の空欄を埋めて、SankakuThread と同じ軌跡を円が移動するようにせよ。ただし、SankakuThread.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。

なお念のため javax.swing.Timer クラスの isRunning メソッドの Java API Reference の文書を次に示す。

パッケージ javax.swing

クラス Timer

public class Timer extends Object

指定された間隔でアクションイベントを発生させます。

メソッドの詳細

isRunning

public boolean isRunning()

Timer が実行中の場合は true を返します。

戻り値:
Timer が実行されている場合は true、 それ以外の場合は false。

以下に参考のために授業配布プリントの KeyTest.java, LeftRightButton.java, LeftRightButton4.java, Guruguru.java, GuruguruTh.java, Point.java, ColorPoint.java, Quadratic.kt, SequeceTest.kt のソースを掲載する。

ファイル KeyTest.java

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

public class KeyTest extends JPanel implements KeyListener {
   private int x = 50, y = 20;
   public KeyTest() {
      setPreferredSize(new Dimension(150, 150));
      setFocusable(true); addKeyListener(this);
   }

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

   public void keyTyped(KeyEvent e) {
      int k = e.getKeyChar();
      if (k == 'u') y -= 10; else if (k == 'd') y += 10;
      repaint();
   }
   public void keyReleased(KeyEvent e) {}
   public void keyPressed(KeyEvent e) {}
}
void main() {/* 省略 */}
```

ファイル LeftRightButton.java

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

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) x -= 10; else if (source == rBtn) x += 10;
        repaint();
    }
}
void main() {/* 省略 */}
```

ファイル LeftRightButton4.java

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

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);
    }
}
void main() {/* 省略 */}
```

ファイル Guruguru.java

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

class Guruguru extends JPanel {
  private int r = 50;
  private int x = 110, y = 70 ;
  private double theta = 0; // 角度

  public Guruguru() {
    setPreferredSize(new Dimension(200, 180));
    Timer timer = new Timer(30, e -> {
      theta += 0.02;
      x =  60 + (int)(r * Math.cos(theta));
      y = 100 - (int)(r * Math.sin(theta));
      repaint();
    });
    timer.start();
    JButton startBtn = new JButton("start");
    startBtn.addActionListener(e -> timer.start());
    JButton stopBtn = new JButton("stop");
    stopBtn.addActionListener(e -> timer.stop());
    setLayout(new FlowLayout());
    add(startBtn); add(stopBtn);
  }

  @Override
  public void paintComponent(Graphics g) {
    super.paintComponent(g);
    g.drawString("Hello, World!", x, y);
  }
}
void main() {/* 省略 */}
```

ファイル GuruguruTh.java

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

public class GuruguruTh 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 GuruguruTh() {
    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) {}
    }
  }
}
void main() { /* 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) {
        for (int 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
    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（2026 年 07 月 30 日）

学籍番号		氏名	

I. (2 × 3)

(1)		(2)	

II. (1 + 8 × 2 + 1 + 2 × 3)

(1)	
(2)	
(3)	
(4)	
(5)	
(6)	
(7)	
(8)	
(9)	
(10)	
(11)	
(12)	

中間言語の代替問題はここまで

III. (2 × 4)

(1)	
(2)	

IV. (2 × 4)

(1)		(2)	

V. (3 + 7)

(1)	
(2)	

VI. (3 × 3 + 5)

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

授業・テストの感想