オブジェクト指向言語・期末テスト問題用紙

（'10 年 7 月 30 日・ 10:30 〜 12:00）

解答上、その他の注意事項

I. 問題は、問 I〜VII までである。

II. 解答用紙の右上の欄に学籍番号・名前を記入すること。

III. 解答欄を間違えないよう注意すること。

IV. 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。

V. 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。

VI. テストの配点は 80 点である。合格はレポートの得点を加点して、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。例えば `this==null` と書く代わりに, Ⓣ==Ⓝと書いて良い。（必ずで◯囲むこと。）

Ⓐ `ActionListener` Ⓐ' `addActionListener` Ⓒ `class` Ⓓ `actionPerformed`
Ⓔ `ActionEvent` Ⓖ `getSource` Ⓘ `implements` Ⓙ `JApplet` Ⓚ `KeyListener`
Ⓚ' `addKeyListener` Ⓜ `MouseListener` Ⓜ' `addMouseListener` Ⓝ `null`
Ⓟ `public` Ⓠ `equals` Ⓡ `Runnable` Ⓢ `System.out.println` Ⓣ `this`
Ⓥ `void` Ⓦ `new` Ⓧ `extends`

また、参考のために問題用紙の末尾に授業プリントの `KeyTest.java`, `UpDownButton.java`, `UpDownButton3.java`, `BubbleSort1.java`, `BubbleSort2.java`, `Point.java`, `ColorPoint.java` のソースを掲載する。

I. 次の各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 **1 つとは限らない**。

(i) 次のうち Java のクラスの名前として（文法的に）許されないのは、どれか?

(A). `2_1`　(B). `Foo777`　(C). `Project-X`　(D). `Bar__`

(ii) 次の Java に関する文章のうち正しいものはどれか?

(A). Java の文法は C 言語に似ており、そのため、Java のコンパイラは C 言語のソースファイルをコンパイルすることも可能となっている。

(B). Java はポインタをプログラマに提供しないことで、安全性を確保している。

(C). Java はゴミ集めの機能を標準で持っていないので、メモリが少なくても高速に動作する。

(D). Java と JavaScript の違いは主に実行方式（Java — 中間言語方式、JavaScript — インタプリタ方式）であり、両者の文法はまったく同じである。

(iii) 要素の型が `Color` 型であるような、`ArrayList` 型（サイズ変更可能な配列の型）の変数 `a` を宣言したい。正しい書き方を 1 つ、以下の選択肢から選べ。

(A). `Color<ArrayList> a = new Color<ArrayList>();`

(B). `ArrayList.Color<> a = new ArrayList.Color<>();`

(C). `ArrayList<Color> a = new ArrayList<Color>();`

(D). `Color<> a = new Color<>();`

(iv) 1 から 5 までの数とそれに 10 を足した数を次のように出力したい。

```
n が 1 のとき n+10 は 11
n が 2 のとき n+10 は 12
n が 3 のとき n+10 は 13
n が 4 のとき n+10 は 14
n が 5 のとき n+10 は 15
```

次のプログラム（の一部）:

```
int n;
for (n=1; n<=5; n++) {
    ?                                   ;
}
```

の空欄 `?` にふさわしい式を下の選択肢の中から選べ。

(A). `System.out.println("n が "{n}" のとき n+10 は "{n+10})`

(B). `System.out.println("n が ¥"n¥" のとき n+10 は ¥"n+10¥"")`

(C). `System.out.println("n が +n+ のとき n+10 は +(n+10)+")`

(D). `System.out.println("n が "+n+" のとき n+10 は "+(n+10))`

II. 次のプログラムは、コマンドライン引数として与えられた整数の和を計算する。

```
public class SumArgs {
  public static void main(String[] args) {
    int k, s = 0;
    for (k=0; k<      (i)      ; k++) {
      int a =      (ii)      (args[k]);
      s += a;
    }
    System.out.printf("コマンドライン引数の和は %d です。%n", s);
  }
}
```

実行例は以下のようになる。

```
% java SumArgs 1 2 3 4 5
コマンドライン引数の和は 15 です。
% java SumArgs 1 2 4 8 16
コマンドライン引数の和は 31 です。
```

(i)〜(ii) の空欄にふさわしいものを以下の選択肢から 1 つずつ 選べ。

(i) の選択肢

(A). `length(args)` (B). `#args` (C). `args.length` (D). `sizeof(args)`

(ii) の選択肢

(A). `Integer.parseInt` (B). `String.valueOf`

(C). `toInteger` (D). `Double.parseDouble`

III. 次の文章は String クラス（java.lang.String クラス）の compareTo メソッドの説明の Java™ API 仕様からの抜粋である。

> public int **compareTo**(String anotherString)
>
> 2 つの文字列を辞書的に比較します。比較は文字列内のそれぞれの文字の Unicode 値に基づいて行われます。この String オブジェクトによって表される文字シーケンスが、引数文字列によって表される文字シーケンスと辞書的に比較されます。この String オブジェクトが辞書的に引数文字列より前にある場合は、結果は負の整数になります。この String オブジェクトが辞書的に引数文字列の後ろにある場合、結果は正の整数になります。文字列が等しい場合、結果は 0 になります。（後略）
>
> **パラメータ:**
>
> anotherString – 比較対象の String
>
> **戻り値:**
>
> 引数文字列がこの文字列に等しい場合は、値 0。この文字列が文字列引数より辞書的に小さい場合は、0 より小さい値。この文字列が文字列引数より辞書的に大きい場合は、0 より大きい値

このメソッドを使用しテストするプログラムを次のように作成する。

ファイル名: CompareToTest.java

```
public class CompareToTest {
  public static void main(String[] args) {
    String str1 = args[0];
    String str2 = args[1];

    ┌──────────────────────────────────────┐
    │                  ?                   │
    └──────────────────────────────────────┘
    if (result > 0) {
      System.out.println(str1+" > "+str2);
    } else if (result < 0) {
      System.out.println(str1+" < "+str2);
    } else {
      System.out.println(str1+" = "+str2);
    }
  }
}
```

このプログラムは、コマンドラインパラメータを 2 つ受け取り、その 2 つの文字列を比較して、>, <, =のいずれかを判定する。実行例は次のようになる。

```
prompt> java CompareToTest baz bar
baz > bar
prompt> java CompareToTest bar bar
bar = bar
prompt> java CompareToTest bar baz
bar < baz
```

上のプログラムの空欄を埋めよ。

IV. 次のプログラム（ColorPallete クラス）は、“C”, “M”, “Y” という 3 つのボタンと、色見本を表示し、ボタンを押せば、文字の色が変わるというアプレットである。

以下の空欄 (i) 〜 (iii) を埋めて、プログラムを完成させよ。

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

public class ColorPallete (i) {
    int r = 255, g = 255,  b = 255;
    JButton cBtn, mBtn, yBtn;

    @Override
    public void (ii)() {
        cBtn = new JButton("C");
        mBtn = new JButton("M");
        yBtn = new JButton("Y");

        cBtn.addActionListener(this);
        mBtn.addActionListener(this);
        yBtn.addActionListener(this);
        setLayout(new FlowLayout());
        add(cBtn); add(mBtn); add(yBtn);
    }

    public void (iii)(ActionEvent aEvt) {
        Object pushed = aEvt.getSource();
        if (pushed == cBtn) {
            r-=4;
            if (r<0) r+=256;
        } else if (pushed == mBtn) {
            g-=4;
            if (g<0) g+=256;
        } else if (pushed == yBtn) {
            b-=4;
            if (b<0) b+=256;
        }
        repaint();
    }

    @Override
    public void paint(Graphics gr) {
        super.paint(gr);

        String str = String.format("%02x%02x%02x", r, g, b);
        gr.setFont(new Font("monospaced", Font.BOLD, 40));
        gr.setColor(new Color(r, g, b));
        gr.drawString(str, 10, 100);
    }
}
```

次ページにつづく。

さらに、同じ動作をするプログラムを匿名（無名）クラスを利用して ColorPallete2 クラスとして実装する。以下の空欄 (iv) 〜 (v) を埋めて、プログラムを完成させよ。

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

public class ColorPallete2 (iv) {
    int r = 255, g = 255,  b = 255;

    @Override
    public void (ii)() {
        JButton cBtn  = new JButton("C");
        cBtn.addActionListener(
                (v)
        );
        JButton mBtn  = new JButton("M");
        mBtn.addActionListener(
                （他の問のヒントになるので省略）
        );
        JButton yBtn  = new JButton("Y");
        yBtn.addActionListener(
                （他の問のヒントになるので省略）
        );
        setLayout(new FlowLayout());
        add(cBtn); add(mBtn); add(yBtn);
    }

    /* paint メソッドは ColorPallete クラスと同一なので省略 */
}
```

V. つぎに定義されるクラス Animal を継承して、3 つのクラス Dog, Cat, Chicken を作成する。

ファイル: Animal.java

```
public class Animal {
  public int age;

  public Animal() {   // 注 1
    age = 0;
  }

  public void sound() {
    System.out.print("???, ");
    age++;
  }
}
```

ファイル: Dog.java

```
public class Dog        (i)        {
  public Dog() {   // 注 1
    super();
  }

  public void sound() {
    if (age<2) {
      System.out.print("Kyan, ");
    } else {
      System.out.print("Wan, ");
    }
    age++;
  }
}
```

ファイル: Cat.java

```
class Cat       (i)        {
  public Cat() {   // 注 1
    super();
  }

  public void sound() {
    System.out.print("Nyaa, " );
    age++;
  }
}
```

注 1: これらのコンストラクタは、実際には定義する必要はない。(デフォルトで定義されるコンストラクタと同等である。)

ファイル: Chicken.java

```
public class Chicken [ (i) ] {                                          // 01
  public boolean sex; // false -- male, true -- female                  // 02
                                                                        // 03
  public Chicken(boolean s) {                                           // 04
    super();                                                            // 05
    sex = s;                                                            // 06
  }                                                                     // 07
                                                                        // 08
  public void sound() {                                                 // 09
    if (age<1) {                                                        // 10
      System.out.print("Piyo, ");                                       // 11
    } else if (sex) {                                                   // 12
      System.out.print("Kokko, ");                                      // 13
    } else {                                                            // 14
      System.out.print("Kokekokkooo,");                                // 15
    }                                                                   // 16
    age++;                                                              // 17
  }                                                                     // 18
}
```

また、AnimalTest クラスは、これらのクラスのテスト用の main メソッドを持つ。

ファイル: AnimalTest.java

```
public class AnimalTest {                                               // 01
  public static void main(String[] args) {                              // 02
    Animal[] animals = new Animal[3];                                   // 03
    animals[0] = new Dog();                                             // 04
    animals[1] = new Cat();                                             // 05
    Chicken c = new Chicken(false);                                     // 06
    c.sex = true;                                                       // 07
    animals[2] = c;                                                     // 08
                                                                        // 09
    int i, j;                                                           // 10
    for(i=0; i<3; i++) {                                                // 11
      for(j=0; j<3; j++) {                                              // 12
        animals[j].sound();                                             // 13
      }                                                                 // 14
      System.out.print("¥n");                                           // 15
    }                                                                   // 16
  }                                                                     // 17
}                                                                       // 18
```

(i) の空白（3 箇所に共通）を埋めて、クラスの定義を完成させよ。

(ii) Chicken クラスのフィールド sex の値はコンストラクタでのみ与えることができるものとし、一旦作成した後は、他のオブジェクトのメソッドからは変更できないようにしたい。（例えば、AnimalTest クラスの 7 行目はコンパイル時にエラーになるようにしたい。）

Chicken クラスの定義の何行めをどのように変更すれば良いか？（プログラム中の行の末尾の数字がクラスの中での行数を表す。）

(iii) AnimalTest クラスの 7 行目をコメントアウトし、このプログラム（AnimalTest クラスの main メソッド）を実行するとき、出力はどうなるか？（空白は気にしなくて良い。）

VI. 下のプログラムは、キーボードから打ち込まれた 'h'（左）, 'j'（下）, 'k'（上）, 'l'（右）キーに従って、向きを変え、時間とともにのびていく折れ線のアニメーションを描画する Java アプレットである。

ファイル: TronModoki.java

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

public class TronModoki (i) {
```

```
  ArrayList<Character> keys = new ArrayList<Character>();
  public char last = 'k' ;
  private Thread myThread = null;
```

```
  @Override
  public void start() {
    if (myThread==null) {
      myThread = (ii) ;
      myThread.start();
    }
  }
```

```
  @Override
  public void stop() {
    myThread = null;
  }
```

```
  @Override
  public void init() {
      (iii)
  }
```

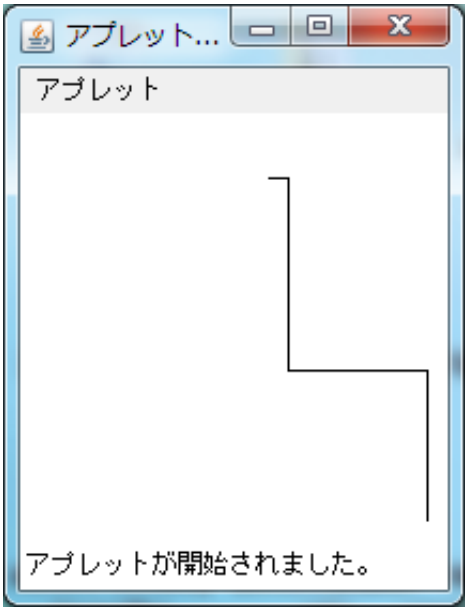
```
  @Override
  public void paint(Graphics g) {
    int x = 190, y = 190;
    final int step = 5;
    for (char k : keys) {
      int x1 = x, y1 =y;
      switch(k) {
      case 'h': x1-=step; break;
      case 'j': y1+=step; break;
      case 'k': y1-=step; break;
      case 'l': x1+=step; break;
      default: break;
      }
      g.drawLine(x, y, x1, y1);
      x = x1; y = y1;
    }
  }
```

```
  public void run() {
    while (myThread==Thread.currentThread()) {
      keys.add(last);
      repaint();
      try {
        Thread.sleep(300);
      } catch (InterruptedException e) {}
    }
  }
```

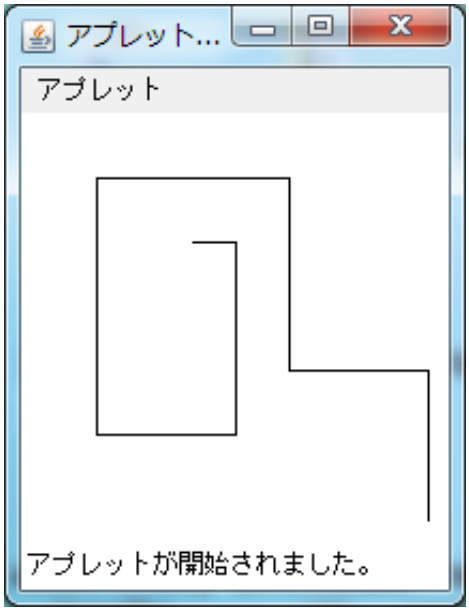
```
  public void keyPressed(KeyEvent e) {}
  public void keyReleased(KeyEvent e) {}

  public void keyTyped(KeyEvent e) {
    char c = e.getKeyChar();
    if ("hjkl".indexOf(c) >= 0) {
      last = c;
    }
  }
```

```
}
```

次にスクリーンショットを示す。



(1) 'h', 'k', 'h' と打ったところ



(2) さらに、いくつかのキーを打ったところ

(i)〜(iii) の空欄を埋めてプログラムを完成させよ。

VII. map, filter を次のように定義された Scala の関数とするとき、

```
def map[A,B](f : A => B, xs : List[A]) : List[B] = xs match {
  case Nil     => Nil
  case y :: ys => f(y) :: map(f, ys)
}

def filter[T](p : T => Boolean, xs : List[T]) : List[T] = xs match {
  case Nil     => Nil
  case y :: ys => if (p(y)) y :: filter(p, ys) else filter(p, ys)
}
```

次の式の値を以下の選択肢から選べ。

(i) `map((x:  Int) => 2*x, List(1, 2, 3))`

(ii) `filter((x:  Int) => x%2==1, List(3, 4, 5))`

選択肢（共通）

(A). `List(1, 2, 3)`　(B). `List(2, 4, 6)`　(C). `List(3, 4, 5)`　(D). `List(3, 5)`
(E). `List(4)`　(F). `4`　(G). `8`　(H). `12`　(I). `15`　(J). `30`　(K). `60`

以下に参考のために授業配布プリントのKeyTest.java, UpDownButton.java, UpDownButton3.java, BubbleSort1.java, BubbleSort2.java, Point.java, ColorPoint.java のソースを掲載する。

KeyTest.java

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

public class KeyTest extends JApplet implements KeyListener {
  int x=50, y=20;

  @Override
  public void init() {
    addKeyListener(this);
  }

  @Override
  public void paint(Graphics g) {
    super.paint(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) {}
}
```

UpDownButton.java

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

public class UpDownButton extends JApplet implements ActionListener {
    int x=20;
    JButton left, right;

    @Override
    public void init() {
        left  = new JButton("Left");
        right = new JButton("Right");
        left.addActionListener(this);
        right.addActionListener(this);
        setLayout(new FlowLayout());
        add(left); add(right);
    }

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

    public void actionPerformed(ActionEvent e) {
        if (e.getSource() == left) { // Left が押された
            x-=10;
        } else if (e.getSource() == right) { // Right が押された
            x+=10;
        }
        repaint();
    }
}
```

UpDownButton3.java

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

public class UpDownButton3 extends JApplet {
    int x=20;

    @Override
    public void init() {
        JButton left  = new JButton("Left");
        JButton right = new JButton("Right");
        left.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                x-=10;
                repaint();
            }
        });
        right.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                x+=10;
                repaint();
            }
        });
        setLayout(new FlowLayout());
        add(left); add(right);
    }

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

BubbleSort1.java

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

public class BubbleSort1 extends JApplet implements Runnable {
  int[] args = {10, 3, 46, 7, 23, 34, 8, 12, 4, 45, 44, 52};
  Color[] cs ={Color.RED, Color.ORANGE, Color.GREEN, Color.BLUE};
  Thread thread=null;

  @Override
  public void start() {
    if (thread == null) {
      thread = new Thread(this);
      thread.start();
    }
  }

  @Override
  public void stop() {
    thread = null;
  }

  @Override
  public void paint(Graphics g) {
    int i;
    super.paint(g);
    for(i=0; i<args.length; i++) {
      g.setColor(cs[args[i]%cs.length]);
      g.fillRect(0, i*10, args[i]*5, 10);
    }
  }

  public void run() {
    int i, j;
    Thread thisThread = Thread.currentThread();
    for (i=0; i<args.length-1; i++) {
      for (j=args.length-1; thread == thisThread && j>i; j--) {
        if (args[j-1]>args[j]) { // スワップする。
          int tmp=args[j-1]; args[j-1]=args[j]; args[j]=tmp;
        }
        repaint();
        try { // repaint の後でしばらく止まる
          Thread.sleep(500);
        } catch (InterruptedException e) {}
      }
    }
  }
}
```

BubbleSort2.java

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

public class BubbleSort2 extends JApplet implements Runnable, ActionListener {
  int[] args = { 10, 3, 46, 7, 23, 34, 8, 12, 4, 45, 44, 52};
  Color[] cs ={Color.RED, Color.ORANGE, Color.GREEN, Color.BLUE};
  Thread thread=null;
  private boolean threadSuspended=true;

  @Override
  public void init() {
    JButton step = new JButton("Step");
    step.addActionListener(this);
    setLayout(new FlowLayout());
    add(step);
  }

  // start, stop, paint メソッドは BubbleSort1.java と同一なので省略する。

  public synchronized void actionPerformed(ActionEvent e) {
    threadSuspended=false;
    notify();
  }

  public void run() {
    int i, j;

    for (i=0; i<args.length-1; i++) {
      for (j=args.length-1; j>i; j--) {
        if (args[j-1]>args[j]) { // スワップする。
          int tmp=args[j-1]; args[j-1]=args[j]; args[j]=tmp;
        }
        repaint();
        try {  // repaint の後で止まる
          synchronized(this) {
             while (threadSuspended) {
                wait();
             }
             threadSuspended=true;
          }
        } catch (InterruptedException e) {}
      }
    }
  }
}
```

Point.java

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

  public void move(int dx, int dy) {
    x += dx;  y += dy;
  }

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

ColorPoint.java

```
public class ColorPoint extends Point {
  private String[] cs = {"black", "red", "green", "yellow",
                         "blue", "magenta", "cyan", "white"};
  private int color;     // 0-黒 1-赤 2-緑 3-黄 4-青 5-紫 6-水 7-白

  @Override
  public void print() {
    System.out.printf("<font color='%s'>", getColor());  // 色の指定
    super.print();
    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 = i; 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);
  }

  public String getColor() {
    return cs[color];
  }
}
```

オブジェクト指向言語・期末テスト解答用紙（'10 年 7 月 30 日）

学籍番号		氏名	

I. （4 × 4）

(i).		(ii).		(iii).		(iv).	

II. （4 × 2）

(i).		(ii).	

III. (4 × 1)

IV. （4 × 5）

(i).	
(ii).	(iii).
(iv).	
(v).	

V. （4 × 3）

(i).	
(ii).	

(iii).	

VI. (4 × 3)

(i).	
(ii).	
(iii).	

VII. (4 × 2)

(i).		(ii).	

授業・テストの感想

……

……

……

……

……

……

……

……

……

……