オブジェクト指向言語・期末テスト問題用紙

（'12 年 7 月 27 日・ 10:30 〜 12:00）

解答上、その他の注意事項

I. 問題は、問 I〜VI まである。

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 **一つとは限らない**。

(i) 次のうち Java のクラスの名前として使うことができるのは、どれか?

(A). `2Apples` (B). `Vitamin-C` (C). `Peach21` (D). `Pine_Apple`

(ii) 次の Java に関する文章のうち正しいものはどれか?

(A). Java はゴミ集めを標準で持っていないので、少ないメモリでも高速に動作する。

(B). Java は中間言語方式を取り、機種依存性を避けながら効率の良い実行を目指している。

(C). Java は、オブジェクト指向という概念を最初に導入したプログラミング言語として知られている。

(D). JavaScript は Java のサブセットであり、JavaScript のプログラムは Java プログラムとしても実行することができる。

(iii) 次のプログラムは、コマンドライン引数の中の 7 の倍数の個数をカウントする。

```
public class CountMultipleSeven {
  public static void main(String[] args) {
    int i, count=0;
    for(i=0; i<args.length; i++) {
      int a = [ ? ];
      if (a%7==0) {
        count++;
      }
    }
    System.out.printf("7の倍数は␣%d個です。%n", count);
  }
}
```

実行例は以下のようになる。

```
% java CountMultipleSeven 10 20 30 40 50
7 の倍数は 0 個です。
% java CountMultipleSeven 14 21 35 38
7 の倍数は 3 個です。
```

空欄[?]にふさわしい式を以下の選択肢から一つ選べ。

(A). `String.valueOf(args+i)` (B). `String.valueOf(args[i])`

(C). `Integer.parseInt(args+i)` (D). `Integer.parseInt(args[i])`

(iv) 次のプログラム（の一部）の出力を以下の選択肢から選べ。

```
int[][] xss = {{0}, {1, 2}, {3, 4, 5}};
System.out.printf("%d␣%d␣%d␣", xss.length, xss[0].length, xss[1][1]);
```

(A). 3 0 0 (B). 3 1 2 (C). 1 1 0 (D). 1 3 2

(v) Javaで1から5までの数とその3で割った余りの数を次のように出力したい。

```
n が 1 のとき n%3 は 1
n が 2 のとき n%3 は 2
n が 3 のとき n%3 は 0
n が 4 のとき n%3 は 1
n が 5 のとき n%3 は 2
```

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%3 は+n%3+")`

(B). `System.out.println("nが{n}のとき n%3は{n%3}")`

(C). `System.out.println("nが "+n+" のとき n%3 は"+(n%3))`

(D). `System.out.println("nが"+"n"+"のとき n%3 は"+"n%3")`

(vi) 0による除算を行うと `ArithmeticException` という種類の例外が発生する。次のプログラム（の一部）の出力を以下の選択肢から選べ。

```
int i;
for(i=-3; i<=3; i++) {
  try {
    System.out.printf("%d␣", 6/i);
  } catch (ArithmeticException e) {
    System.out.printf("(/␣by␣0)␣");
  }
}
```

(A). `-2 -3 -6`

(B). `-2 -3 -6 (/ by 0)`

(C). `-2 -3 -6 6 3 2`

(D). `-2 -3 -6 (/ by 0) 6 3 2`

II. myPackage.Chome クラス（myPackage パッケージの Chome クラス）、 myPackage.ChomeTest クラス（myPackage パッケージの ChomeTest クラス）、および ChomeTest クラス（無名パッケージの ChomeTest クラス）をそれぞれ次のように定義する

パス: myPackage/Chome.java

```
package myPackage;

public class Chome {
  String a;
  private String b;
  public  String c;

  public Chome(String x, String y, String z) {
    a = x; b = y; c = z;
  }
}
```

パス: myPackage/ChomeTest.java

```
package myPackage;

public class ChomeTest {
  public static void main(String[] args) {
    Chome chome = new Chome("abc", "def", "ghi");
    System.out.println(chome.a);   // い
    System.out.println(chome.b);   // ろ
    System.out.println(chome.c);   // は
  }
}
```

パス: ./ChomeTest.java

```
import myPackage.Chome;

public class ChomeTest {
  public static void main(String[] args) {
    Chome chome = new Chome("rst", "uvw", "xyz");
    System.out.println(chome.a);   // に
    System.out.println(chome.b);   // ほ
    System.out.println(chome.c);   // へ
  }
}
```

い〜へ、の行で（コンパイル時に）エラーにならない行には○を、エラーになる行には×をつけよ。

III. 次の文章は java.util.ArrayList クラスの isEmpty メソッドと remove メソッドの説明の Java™ API 仕様からの抜粋である。

java.util
クラス **ArrayList<E>**
...
public boolean **isEmpty**()
リストに要素がないかどうかを判定します。

戻り値:

リストに要素がない場合は true、そうでない場合は false

public E **remove**(int index)
リスト内の指定された位置から要素を削除します。そして，後続の要素を左側に移動し、それぞれのインデックスから 1 を減算します。

パラメータ:

index – 削除される要素のインデックス

戻り値:

リストから削除した要素

例外:

IndexOutOfBoundsException – インデックスが範囲外の場合 (index < 0 || index >= size())

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。また、実行例は右のようになる。

ファイル名: QueueTest.java

```
import java.util.ArrayList;

public class QueueTest {
    public static void main(String[] args) {
        ArrayList<String> queue = new ArrayList<String>();

        queue.add("");
        while(![ (i) ]) {
            String str = [ (ii) ];
            System.out.printf("%s%n", str);
            if (str.length()<=3) {
                queue.add(str+"a");
                queue.add(str+"b");
                queue.add(str+"c");
            }
        }
    }
}
```

実行例

```
a
b
c
aa
ab
ac
ba
bb
bc
ca
cb
cc
aaa
... （略） ...
ccbc
ccca
cccb
cccc
```

このプログラムは、空文字列（""）から始めて、ArrayList をキューとして利用して、'a', 'b', 'c' の 3 種類の文字からなる、長さ 4 までの文字列を次々と生成する。

上のプログラムの空欄 (i) に queue に要素がないかどうか判定する式、空欄 (ii) に queue の最初の（インデックス 0 の）要素を削除し、その削除した要素を返す式、をそれぞれ埋めよ。

IV. 以下はキーボードのキーを打つと画面上の文字列（“Hello!”）の色が変化する簡単なアプレットのプログラムである。“r” キーが押されると赤色、“g” キーが押されると緑色、“b” キーが押されると青色、その他のキーが押されると黒色になる。(i)〜(iii) の空欄を埋めて、プログラムを完成させよ。

ファイル名: KeyColor.java

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  public class KeyColor (i) {
6      private Color textColor;
7
8      @Override
9      public void init() {
10         setFocusable(true);
11         (ii)
12     }
13
14     @Override
15     public void paint(Graphics g) {
16         super.paint(g);
17         (iii)
18         g.drawString("Hello!", 50, 50);
19     }
20
21     private static Color keyToColor(int k) {
22         switch (k) {
23         case 'r': return Color.RED;
24         case 'g': return Color.GREEN;
25         case 'b': return Color.BLUE;
26         default:  return Color.BLACK;
27         }
28     }
29
30     public void keyPressed(KeyEvent e) {}
31     public void keyReleased(KeyEvent e) {}
32     public void keyTyped(KeyEvent e) {
33         textColor = keyToColor(e.getKeyChar());
34         repaint();
35     }
36 }
```

さらに、同じ動作をするプログラムを匿名（無名）クラスを KeyListener として利用して KeyColor2 クラスとして実装する。(iv)〜(v) の空欄を埋めて、プログラムを完成させよ。節約のため、必要なら「KeyColor.java の○行めから△行めと同じ」というような書き方をしても良い。

ファイル名: KeyColor2.java

```
1  import javax.swing.*;
2  import java.awt.*;
3  import java.awt.event.*;
4
5  public class KeyColor2 [          (iv)          ] {
6      private Color textColor;
7
8      @Override
9      public void init() {
10         setFocusable(true);
11         [          (v)          ]
12     }
13
14     /* paint, keyToColor は KeyColor と同じなので省略 */
15 }
```

V. 次に定義されるクラス Fish を継承して、

ファイル名: Fish.java

```
1  public class Fish {
2    public int age;
3
4    public void grow() {
5      age++;
6    }
7
8    public void showName() {
9      System.out.print("サカナ␣");
10   }
11
12   public void growShowName() {
13     showName();
14     grow();
15   }
16 }
```

3 つのクラス Seriola, Mugil, Carp を定義する。

ファイル名: Seriola.java

```
1  public class Seriola [     ♠     ]{
2    @Override
3    public void showName() {
4      if (age==0) {
5        System.out.print("ツバス␣");
```

```
    } else if (age==1) {
      System.out.print("ハマチ␣");
    } else {
      System.out.print("ブリ␣");
    }
  }
}
```

ファイル名: Mugil.java

```
public class Mugil [♠]{
  @Override
  public void showName() {
    if (age==0) {
      System.out.print("オボコ␣");
    } else if (age<=3) {
      System.out.print("ボラ␣");
    } else {
      System.out.print("トド␣");
    }
  }
}
```

ファイル名: Carp.java

```
public class Carp [♠]{
  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␣");
    }
  }
}
```

また、FishTest 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。
ファイル名: FishTest.java

```
public class FishTest {
  public static void main(String[] args) {
    Fish[] fish = new Fish[3];
    fish[0] = new Seriola();
    fish[1] = new Mugil();
    Carp c  = new Carp("緋");
    c.color = "緑";
```

```
 8      fish[2] = c;
 9
10      int i, j;
11      for (i=0; i<3; i++) {
12        for (j=0; j<3; j++) {
13          fish[j].growShowName();
14        }
15        System.out.println();    // 改行を出力
16      }
17
18      for (i=0; i<998; i++) {
19        for (j=0; j<3; j++) {
20          fish[j].grow();
21        }
22      }
23      for (j=0; j<3; j++) {
24        fish[j].showName();
25      }
26      System.out.println();    // 改行を出力
27    }
28  }
```

(i) ♠の空白（3 箇所で共通）を埋めて、クラスの定義を完成させよ。

(ii) Carp クラスのフィールド color 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、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 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。）

Carp クラスの定義の何行めをどのように変更すれば良いか？（行の左端の数字がクラスの中での行数を表す。）

(iii) FishTest クラスの 7 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。）

VI. 下のプログラムは、円が時間の経過とともに大きくなったり小さくなったりするアニメーションを表示する Java アプレットである。

ファイル名: BallThread.java

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

public class BallThread [        (i)        ] {
    int r = 50;
    int n = 0;
    Thread theThread = null;

    @Override
    public void start() {
```

```
        if (theThread==null) {
            theThread = new Thread(this);
            theThread.start();
        }
    }

    @Override
    public void stop() {
            (ii)
    }

    public void run() {
        Thread thread0 = Thread.currentThread();
        while(          (iii)          ) {
            r = 50+(int)(50*Math.sin(Math.PI*n/50));
            repaint();
            try {
                Thread.sleep(300);
            } catch (InterruptedException e) {}
            n++;
        }
    }

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        g.fillOval(100-r, 100-r, 2*r, 2*r);
    }
}
```

(i)〜(iii) の空欄を埋めてプログラムを完成させよ。

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 lBtn, rBtn;

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

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

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 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();
      }
    });
    setLayout(new FlowLayout());
    add(lBtn); add(rBtn);
  }

  @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) {}
      }
    }
    thread=null;
  }
}
```

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 String color;

  @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 = 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; }
}
```

オブジェクト指向言語・期末テスト解答用紙（'12 年 7 月 27 日）

学籍番号		氏名	

I. （3 × 6）

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

II. （6 つ正解で 6 点, 5 つ正解で 4 点, 4 つ正解で 2 点）

い	ろ	は	に	ほ	へ

III. （5 × 2）

(i).	
(ii).	

IV. （4, 4, 4, 4, 6）

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

V. （4 × 3）

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

VI. （4 × 3）

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

授業・テストの感想

……

……

……

……

……

……

……

……

……

……

……