オブジェクト指向言語・期末テスト問題用紙

（2016 年 07 月 29 日・ 10:30 〜 12:00）

解答上、その他の注意事項

I. 問題は、問 I〜V 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。（必ず◯で囲むこと。）

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

また、参考のために問題用紙の末尾に授業プリントの Factorial.java, MouseDraw.java, UpDownButton.java, UpDownButton4.java, BubbleSort1.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 のクラス名として使うことができるのは、どれか?

(A). `Foo_Bar` (B). `G'day` (C). `999` (D). `Golgo13`

(ii) 次の文章のうち正しいものはどれか?

(A). Java では public なクラスの名前と、そのクラスを定義しているファイルの名前は一致させなければならない。

(B). Java はゴミ集めを標準で持っていないので、プログラマーが自前でメモリー管理を行う必要がある。

(C). Java から、型宣言（とコンパイル時の型チェック）を取り去った言語を JavaScript と呼ぶ。

(D). Java の文法は C 言語に似せて設計されており、そのため、Java のコンパイラーは 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。

(iii) 次のうち論理型言語に分類される言語はどれか?一つを選べ。

(A). Java (B). Python (C). Prolog (D). Lisp

II. 次の枠内の文章は `java.awt.event.MouseEvent` クラスの `getClickCount` メソッドと、`java.awt.event.InputEvent` クラスの `getWhen` メソッド、および `java.lang.System` クラスの `currentTimeMillis` メソッドの説明の Java™ API 仕様からの抜粋（問題を解くのに関係ない部分は割愛）である。`java.awt.event.InputEvent` クラスは `java.awt.event.MouseEvent` クラスのスーパークラスである。なお、`java.lang` パッケージのクラスは `import` なしでクラス名を使用できる。

java.awt.event
クラス **MouseEvent**
...
public int **getClickCount**()
このイベントに関連したマウスクリック数を返します。
戻り値:
クリック数を表す整数値
...

java.awt.event
クラス **InputEvent**
...
public long **getWhen**()
このイベントが発生したタイムスタンプと協定世界時の UTC 1970 年 1 月 1 日深夜零時との差をミリ秒単位で返します。
...

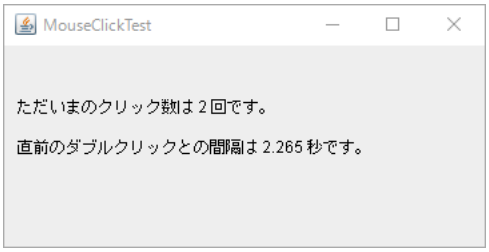
java.lang
クラス **System**
...
public static long **currentTimeMillis**()
　ミリ秒で表される現在の時間を返します。...
戻り値:
　ミリ秒で測定した、現在時刻と協定世界時の UTC 1970 年 1 月 1 日深夜零時との差。
...

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: MouseEventTest.java

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

public class MouseEventTest extends JApplet implements MouseListener {
    int num = 0;
    long curr, prev = 0;

    @Override
    public void init() {
        prev = [ (i) ]  // 現時点の時刻を記録する。
        addMouseListener(this);
    }

    public void mouseClicked(MouseEvent e) {
        num  = [ (ii) ];          // クリック数を記録する
        curr = [ (iii) ];         // マウスイベントの時刻を記録する
        repaint();
    }
    /* mousePressed, mouseReleased, mouseEntered, mouseExited は掲載を省略する。*/

    @Override
    public void paint(Graphics g) {
        super.paint(g);
        g.drawString([ (iv) ], 10, 50);
        if (num == 2) {
            double gap = (curr - prev) / 1000.0;          // 間隔を秒単位に
            g.drawString([ (v) ],
                         10, 80);
            prev = curr;
        }
    }
    /* main メソッドは掲載を省略する */
}
```

このプログラムはマウスのクリック数（ダブルクリックなら 2）と、ダブルクリックの場合は直前のダブルクリックからの経過時間（最初のダブルクリックだけは、プログラム起動からの経過時間）を秒単位で表示する。実行例は次のようになる。



スクリーンショット

(i) 空欄 (i) には現在時刻と UTC 1970 年 1 月 1 日 0 時 0 分との差をミリ秒で表す式が入る。この空欄を埋めよ。

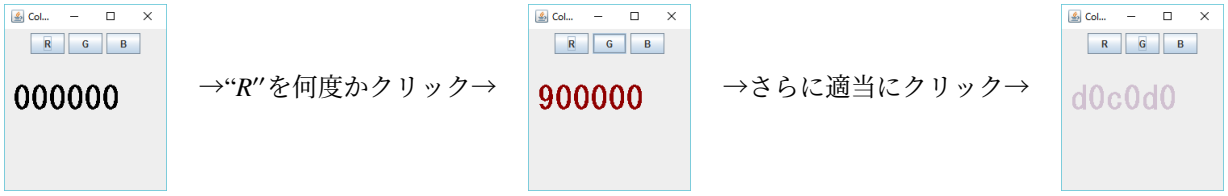
(ii) 空欄 (ii) には、このマウスイベントのマウスクリック数を表す式が入る。この空欄を埋めよ。

(iii) 空欄 (iii) には、このマウスイベントの時刻と UTC 1970 年 1 月 1 日 0 時 0 分との差をミリ秒で表す式が入る。この空欄を埋めよ。

(iv) 空欄 (iv) には、フィールド `num` の値を「ただいまのクリック回数は 2 回です。」のような文字列（`String` 型）にする式が入る。この空欄を埋めよ。

(v) 空欄 (v) には、変数 `gap` の値を「直前のダブルクリックとの間隔は 2.327 秒です。」のような文字列（`String` 型）にする式が入る。（書式指定をする必要はない。）この空欄を埋めよ。

III. 次のプログラム（ColorPallete クラス）は、“R”, “G”, “B” 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。



スクリーンショット

(i)〜(ii) の空欄を埋めて、プログラムを完成させよ。

ファイル名: ColorPallete.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3  import javax.swing.*;
4
5  public class ColorPallete [      (i)      ] {
6      int r = 0, g = 0,  b = 0;
7      JButton rBtn, gBtn, bBtn;
8
9      @Override
10     public void init() {
11         rBtn = new JButton("R");
12         gBtn = new JButton("G");
13         bBtn = new JButton("B");
14
15         rBtn.addActionListener(this);
16         gBtn.addActionListener(this);
17         bBtn.addActionListener(this);
18         setLayout(new FlowLayout());
19         add(rBtn); add(gBtn); add(bBtn);
20     }
21
22     public void actionPerformed(ActionEvent e) {
23         Object pushed = [      (ii)      ];
24         if (pushed == rBtn) {
25             r += 4;
26         } else if (pushed == gBtn) {
27             g += 4;
28         } else if (pushed == bBtn) {
29             b += 4;
30         }
31         repaint();
32     }
33
```

```
34      @Override
35      public void paint(Graphics gr) {
36          super.paint(gr);
37
38          String str = String.format("%02x%02x%02x", r, g, b);
39          gr.setFont(new Font("monospaced", Font.BOLD, 40));
40          gr.setColor(new Color(r, g, b));
41          gr.drawString(str, 10, 100);
42      }
43      /* main メソッドは掲載を省略する */
44  }
```

また、同じ動作をするプログラムを、ラムダ式を利用して、ColorPallete2 クラスとして実装する。(iii)〜(iv) の空欄を埋めて、プログラムを完成させよ。

ファイル名: ColorPallete2.java

```
1   // import は ColorPallete.java と同一なので省略する
2
3   public class ColorPallete2 [ (iii) ] {
4       int r = 0, g = 0,  b = 0;
5       @Override
6       public void init() {
7           JButton rBtn  = new JButton("R");
8           JButton gBtn  = new JButton("G");
9           JButton bBtn  = new JButton("B");
10          rBtn.addActionListener( [ (iv)（ラムダ式） ] );
11          gBtn.addActionListener( [ (iv) のヒントになるので省略する ] );
12          bBtn.addActionListener( [ (iv) のヒントになるので省略する ] );
13          setLayout(new FlowLayout());
14          add(rBtn); add(gBtn); add(bBtn);
15      }
16
17      /* paintメソッドは ColorPallete クラスと同一なので省略する */
18  }
```

IV. ゲーム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した。まず、character.basic.Player クラスは味方キャラクターを表すクラスである。さらに、敵キャラクターのためにいくつかのクラスがある。character.basic.Enemy クラスはすべての敵キャラクターのスーパークラスとなるクラスである。character.basic.GenericEnemy クラスと character.extended.Boss クラスは character.basic.Enemy クラスを継承し、さらに、character.extended.Alice クラスと character.extended.Grace クラスは、character.basic.GenericEnemy クラスを継承する。なお、これらのクラス名には特に意味はない。

character.basic.Enemy クラスは attack メソッドを持ち、character.basic.GenericEnemy クラスは updateDamage メソッドを持つ。

ファイル名: character/basic/Player.java

```
1  package character.basic;
2
3  public class Player {
4      private int hp;
5
6      public Player(int initHp) {
7          hp = initHp;
8      }
9
10     public void damage(int damage) {
11         hp -= damage;
12         System.out.println(damage + "  のダメージ、残り␣HP␣=␣" + hp);
13     }
14 }
```

ファイル名: character/basic/Enemy.java

```
1  package character.basic;
2
3  public class Enemy {
4      public void attack(Player p) {}
5  }
```

ファイル名: character/basic/GenericEnemy.java

```
1  package character.basic;
2
3  public class GenericEnemy [ (i-1) ] {
4      [ (ii) ] String name;
5      [ (iii) ] int damage;
6
7      public GenericEnemy(String n, int d) {
8          name = n;
9          damage = d;
10     }
11
```

```
12      @Override
13      public void attack(Player p) {
14          System.out.print(name + "␣の攻撃:␣");
15          p.damage(damage);
16          updateDamage();
17      }
18
19      public void updateDamage() {}
20  }
```

ファイル名: character/extended/Alice.java

```
1   package character.extended;
2
3   import character.basic.GenericEnemy;
4
5   public class Alice (i-2) {
6       private int init;
7
8       public Alice(int d) {
9           super("Alice", d);
10          init = d;
11      }
12
13      @Override
14      public void updateDamage() {
15          damage += init;   // エラーにならない
16          // name += "!";   // エラーになるのでコメントアウト
17      }
18  }
```

ファイル名: character/extended/Grace.java

```
1   package character.extended;
2
3   import character.basic.GenericEnemy;
4
5   public class Grace (i-3) {
6       private int ratio;
7
8       public Grace(int d, int r) {
9           super("Grace", d);
10          ratio = r;
11      }
12
13      @Override
14      public void updateDamage() {
15          damage *= ratio;  // エラーにならない
16          // name += "?";   // エラーになるのでコメントアウト
17      }
```

```
}
```

ファイル名: character/extended/Boss.java

```
package character.extended;

import character.basic.*;

public class Boss [ (i-4) ] {
    protected int count;

    public Boss() {
        count = 0;
    }

    @Override
    public void attack(Player p) {
        System.out.print("Boss␣の攻撃:␣");
        if (++count >= 3) {
            p.damage(1000);
        } else {
            p.damage(0);
        }
    }
}
```

さらに、次のような main メソッドを持つテスト用プログラム character/test/Main.java を定義する。

ファイル名: character/test/Main.java

```
package character.test;

import java.util.ArrayList;
import character.basic.*;
import character.extended.*;

public class Main {
    public static void main(String[] args) {
        ArrayList<Enemy> enemies = [ (iv) ]; // 空の ArrayList
        Player p = new Player(3000);

        Alice a = new Alice(100);
        // a.damage = 0;          // エラーになるのでコメントアウト
        // a.name = "Agatha";     // エラーになるのでコメントアウト
        enemies.add(a);
        Grace g = new Grace(100, 2);
        // g.damage = 3000000;    // エラーになるのでコメントアウト
        // g.name = "George";     // エラーになるのでコメントアウト
        enemies.add(g);
```

```
20          Boss b  = new Boss();
21          enemies.add(b);
22
23          for (int i = 0; i < 3; i++) {
24              for (int j = 0; j < enemies.size(); j++) {
25                  Enemy e = enemies.get(j);
26                  e.attack(p);;
27              }
28          }
29      }
30  }
```

(i)〜(iv) の空欄を埋めてこれらのクラスの定義を完成させよ。(ii), (iii) の解答は、ソースプログラム中の「エラーにならない」「エラーになるのでコメントアウト」などコメントを参考にし、以下の選択肢 (A)〜(D) から選べ。

(A) `public`　　(B) `private`　　(C) `protected`　　(D)　　（修飾子なし）

(iv) は空の `ArrayList` を生成する式である。

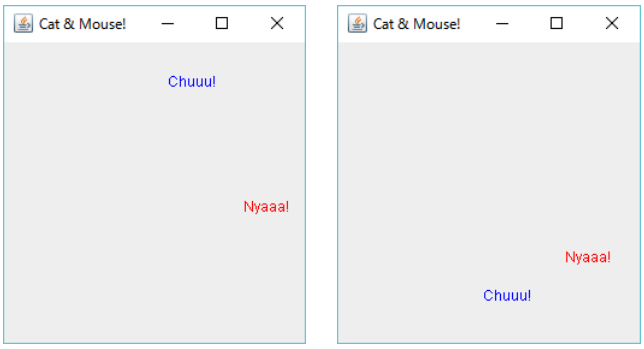
(v) `character.test.Main` クラスの `main` メソッドを実行するとき、出力はどうなるか？（解答用紙に記入するとき、空白の有無や数は気にしなくて良い。）

V. 下のプログラムは、2 つの文字列がある規則に従った動き方をするアニメーションを描画する Java アプレットである。(どのような動き方をするかは、この問題の解答と直接関係はない。)

ファイル: CatMouse.java

```
1  import javax.swing.*;
2  import java.awt.*;
3
4  public class CatMouse extends JApplet implements Runnable {
5      int r = 100, catX = 0, catY = 0, mouseX = 0, mouseY = 0;
6      double theta = 0;
7      Thread thread = null;
8
9      public void start() {
10         if (thread == null) {
11             thread = [          (i)          ];
12             thread.start();
13         }
14     }
15
16     public void stop() {
17         if (thread != null) {
18             thread = null;
19         }
20     }
21
22     public void run() {
23         Thread me = Thread.currentThread();
24         while ([       (ii)       ]) {
25             mouseX = 120 + (int)(r * Math.cos(5 * (theta + 0.15)));
26             mouseY = 120 - (int)(r * Math.sin(8 * (theta + 0.15)));
27             catX   = 120 + (int)(r * Math.cos(5 * theta));
28             catY   = 120 - (int)(r * Math.sin(8 * theta));
29             repaint();
30             try {
31                 Thread.sleep(50);
32             } catch (InterruptedException e) {}
33             theta += 0.02;
34         }
35     }
36
37     public void paint(Graphics g) {
38         super.paint(g);
39         g.setColor(Color.red);
40         g.drawString("Nyaaa!", catX, catY);
41         g.setColor(Color.blue);
42         g.drawString("Chuuu!", mouseX, mouseY);
43     }
44     /* main メソッドは掲載割愛する */
45 }
```

次にスクリーンショットを示す。



(i)〜(ii) 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 Factorial.java, MouseDraw.java, UpDownButton.java, UpDownButton4.java, BubbleSort1.java, Point.java, ColorPoint.java のソースを掲載する。(main メソッドは省略している。)

Factorial.java

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

public class Factorial extends JApplet implements ActionListener {
  JTextField input;
  JLabel output;

  @Override
  public void init() {
    input=new JTextField("0", 8);
    output=new JLabel("␣␣␣1");
    input.addActionListener(this);
    setLayout(new FlowLayout());
    add(input);  add(new JLabel("の階乗は"));
    add(output); add(new JLabel("です。"));
  }

  static int factorial(int n) { // factorial -- 階乗のこと
    int r = 1;
    for (; n>0; n--) {
      r *= n;
    }
    return r;
  }

  public void actionPerformed(ActionEvent e) {
    try {
      int n = Integer.parseInt(input.getText());
      output.setText("␣␣"+factorial(n));
    } catch (NumberFormatException ex) {
      input.setText("数値!");
    }
  }
}
```

MouseDraw.java

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

public class MouseDraw extends JApplet implements MouseListener {
  ArrayList<int[]> points;

  @Override
  public void init() {
    points = new ArrayList<>();
    addMouseListener(this);
  }

  public void mouseClicked(MouseEvent e) {
    points.add(new int[] { e.getX(), e.getY() });
    repaint();
  }

  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
```

```
  public void mousePressed(MouseEvent e) {}
  public void mouseReleased(MouseEvent e) {}

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

    int i, n = points.size();
    for (i=1; i<n; i++) {
      int[] p0 = points.get(i-1);
      int[] p1 = points.get(i);
      g.drawLine(p0[0], p0[1], p1[0], p1[1]);
    }
  }
}
```

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();
  }
}
```

UpDownButton4.java

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

public class UpDownButton4 extends JApplet {
  int x=20;

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

Point.java

```
public class Point {
  public int x, 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 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", ..., "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（2016 年 07 月 29 日）

学籍番号		氏名	

I. （3 × 3）

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

II. （6, 6, 6, 3, 3）

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

III. （4 × 4）

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

IV. （4, 4, 4, 4, 7）

(i-1).		(i-2).	
(i-3).		(i-4).	
(ii).		(iii).	
(iv).			

（裏面に続く）

(v).	

V. （4, 4）

(i).	
(ii).	

授業・テストの感想

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……