オブジェクト指向言語・期末テスト問題用紙

（2017 年 07 月 28 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配布プリントの KeyTest.java, LeftRightButton2.java, LeftRightButton4.java, Guruguru.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). `UTF-8` (B). `ShiftJIS` (C). `EUC_JP` (D). `8859_1`

(ii) 次の文章のうち正しいものはどれか?

(A). Java はネットワークに関する機能を言語に標準のライブラリーとして持っていないので、ネットワークを使ったプログラムは機種に依存する。

(B). Web ブラウザー上でインタプリター方式で実行する Java プログラムのことを JavaScript と呼ぶ。

(C). Java は並行に複数の処理を行なうためのスレッドを言語に標準の機構として持っている。

(D). Java はメモリを自動的に回収するゴミ集めという技術を、初めて取り入れたプログラミング言語である。

(iii) 次のうち関数型言語に分類される言語はどれか?一つを選べ。

(A). Java (B). Haskell (C). C (D). Prolog

II. 次の枠内の文章は java.math.BigInteger クラスのいくつかのメソッドといくつかのクラスフィールドの説明の 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.math
クラス **BigInteger**

変更が不可能な、任意精度の整数です。...

`public static final BigInteger` **ZERO**
BigInteger 定数 0 です。

`public static final BigInteger` **ONE**
BigInteger 定数 1 です。

`public BigInteger` **multiply**`(BigInteger val)`
値が (this * val) である BigInteger を返します。
パラメータ:
val - この BigInteger で乗算する値。
戻り値:
this * val

`public BigInteger` **subtract**`(BigInteger val)`
値が (this - val) である BigInteger を返します。
パラメータ:
val - この BigInteger から減算する値。
戻り値:
this - val

`public boolean` **equals**`(Object x)`
この BigInteger と指定された Object が等しいかどうかを比較します。
パラメータ:
x - この BigInteger と比較する Object。
戻り値:
指定された Object が、この BigInteger と値が等しい BigInteger である場合にだけ true。

`public static BigInteger` **valueOf**`(long val)`
値が指定された long の値と等しい BigInteger を返します。...
パラメータ:
val - 返される BigInteger の値。
戻り値:
指定値を使った BigInteger

補足すると、java.math.BigInteger は、（メモリーが足りる限り）無限に大きな桁数を表現できる整数型である。（一方 int は有限精度なので、22 億くらいの整数までしか表すことができない。）BigInteger は int や long とは異なる型なので、「0」や「123」のようなリテラルを直接代入することはできないし、「*」や「-」、「==」のような、通常の演算子を使用することもできない。演算には上記の BigInteger クラスのメソッドを利用する必要がある。

これらのメソッドやクラスフィールドを使用するプログラムを次のように作成する。

ファイル名: BigFactorial.java

```
import java.math.BigInteger;
import java.util.Scanner;

public class BigFactorial {
  static BigInteger factorial(BigInteger n) {
    if ( (i) ) {
      return (ii);
    } else {
      return (iii);
    }
  }

  public static void main(String[] args) {
    System.err.print("Please␣input␣a␣non-negative␣integer:␣");
    long i = new Scanner(System.in).nextLong();
    System.out.printf("Its␣factorial␣is␣%s.%n",
                      factorial( (iv) ));
  }
}
```

このプログラムは入力された long 型の整数 (i) の階乗を計算する。実行例は次のようになる。

```
Please input a non-negative integer:  50
Its factorial is 30414093201713378043612608166064768844377641568960512000000000000.
```

(i) 空欄 (i) には n が（BigInteger の）0 と等しいときに真、等しくないときに偽を返す条件式が入る、この空欄を埋めよ。

(ii) 空欄 (ii) には、（BigInteger の）1 を表す式が入る。この空欄を埋めよ。

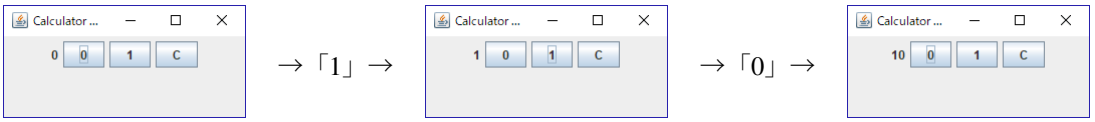
(iii) 空欄 (iii) には、再帰的に階乗を計算する式が入る。この空欄を埋めよ。

(iv) 空欄 (iv) には、long 型の i に対応する BigInteger 型の値を返す式が入る。この空欄を埋めよ。

III. 電卓の動きの一部を模倣するために、次のプログラム（CalculatorTest クラス）を GUI アプリケーションとして作成する。といっても電卓を極端に簡単化して、

- 入力された数字列を表示するためのラベル（digits）が一つある。
- 数字のボタンは「0」と「1」しかない。数字のボタンが押されると、
  - ラベルに表示されている数字列が「0」一字のみのときは、押された数字に置き換わる。
  - ラベルに表示されている数字列が「0」以外のときは、押された数字が右端に追加される。
- 数字以外のボタンはクリアボタン「C」しかない。クリアボタンが押されると、ラベルに表示される数字列が「0」にリセットされる。

というだけのものとする。



スクリーンショット

ファイル名: CalculatorTest.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3
4  import javax.swing.*;
5
6  public class CalculatorTest extends JPanel {
7    JLabel digits;
8
9    private class NumAction [          (i)          ] {
10     String num;
11
12     NumAction(String n) {
13       num = n;
14     }
15
16     public void actionPerformed(ActionEvent e) {
17       String current = digits.getText();
18       if (current.equals("0")) {
19         digits.setText(num);
20       } else {
21         digits.setText(current + num);
22       }
23     }
24   }
25
26   public CalculatorTest() {
27     setPreferredSize(new Dimension(240, 80));
```

```
28      digits = new JLabel("0");
29
30      JButton bt0 = new JButton("0");
31      bt0.addActionListener(          (ii)          );
32      JButton bt1 = new JButton("1");
33      bt1.addActionListener(          (iii)         );
34      JButton clear = new JButton("C");
35      clear.addActionListener(                (iv)                );
36      add(digits);  add(bt0); add(bt1); add(clear);
37    }
38
39    public static void main(String[] args) {
40      SwingUtilities.invokeLater(() -> {
41        JFrame frame = new JFrame("Calculator␣Test!");
42        frame.add(new CalculatorTest());
43        frame.pack();
44        frame.setVisible(true);
45        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
46      });
47    }
48  }
```

(i) NumAction クラスは数字ボタンが押されたときの処理を実装するための内部クラスである。どの数字ボタンでも、処理はほとんど共通のため、コンストラクターの引数で数字をパラメーターとして渡すようにしている。空欄 (i) を埋めて、NumAction クラスの定義を完成させよ。

(ii) 空欄 (ii), (iii) には NumAction クラスのコンストラクターを呼び出す式が入る。これらの空欄を埋めよ。

(iii) 空欄 (iv) には、ラベル（digits）の表示を「0」にリセットするような ラムダ式 が入る。この空欄を埋めよ。

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、basic.GameCharacter クラスはすべてのキャラクターのスーパークラスである。material.Material クラスは、basic.GameCharacter クラスを継承し、“物質” キャラクターのスーパークラスとなるクラスである。material.Stone クラスは material.Material クラスを継承する。animal.Animal クラスは、やはり basic.GameCharacter クラスを継承し、“動物” キャラクターのスーパークラスとなるクラスである。さらに、animal.Cat クラスと animal.Mouse クラスは、animal.Animal 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。

basic.GameCharacter クラスは、exp フィールドと getName メソッド、info メソッド、rankUp メソッドを持ち、 animal.Animal クラスは、さらに growth フィールドと setGrowth メソッド、cry メソッドを持つ。

ファイル名: basic/GameCharacter.java

```
1  package basic;
2
3  public class GameCharacter {
4    protected String getName() { return ""; }
5    (i)  int exp;
6    public String info() {
7      return getName();
8    }
9    public void rankUp() { /* do nothing */ }
10 }
```

ファイル名: material/Material.java

```
1  package material;
2
3  import basic.GameCharacter;
4
5  public class Material (ii) {
6    @Override
7    public String info() {
8      return getName() + ":␣" + exp;  // エラーにならない
9    }
10 }
```

ファイル名: material/Stone.java

```
1  package material;
2
3  public class Stone (iii) {
4    public Stone() {
5      exp = 2000;  // エラーにならない
6    }
7
8    @Override
9    protected String getName() {
10     return "Stone";
```

```
11    }
12  }
```

ファイル名: animal/Animal.java

```
1   package animal;
2
3   import basic.GameCharacter;
4
5   public class Animal (ii) {
6     (iv) int growth;
7
8     protected void setGrowth(int g) {
9       if (g > 0) {
10        growth = g;
11      } else {
12        growth = 0;
13      }
14    }
15
16    protected String cry() {
17      return "";
18    }
19
20    @Override
21    public void rankUp() {
22      exp += growth;  // エラーにならない
23    }
24
25    @Override
26    public String info() {
27      return getName() + ":␣" + exp + "␣" + cry();  // エラーにならない
28    }
29  }
```

ファイル名: animal/Cat.java

```
1   package animal;
2
3   public class Cat (v) {
4     public Cat() {
5       exp = 100;  // エラーにならない
6       setGrowth(20);
7     }
8
9     @Override
10    protected String getName() {
11      return "Cat";
12    }
13
```

```
14    @Override
15    protected String cry() {
16      if (exp < 128) {  // エラーにならない
17        return "Myuu";
18      } else {
19        return "Nyaa";
20      }
21    }
22  }
```

ファイル名: animal/Mouse.java

```
1   package animal;
2
3   public class Mouse (v) {
4     public Mouse() {
5       exp = 0;   // エラーにならない
6       // growth = -20;  // 先頭の//を取る　とエラーになる
7       setGrowth(50);
8     }
9
10    @Override
11    protected String getName() {
12      return "Mouse";
13    }
14
15    @Override
16    protected String cry() {
17      return "Chuu";
18    }
19  }
```

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: test/Main.java を定義する。

ファイル名: test/Main.java

```
1   package test;
2
3   import animal.*;
4   import basic.*;
5   import material.*;
6
7   public class Main {
8     public static void main(String[] args) {
9       GameCharacter[] characters = {
10          new Stone(), new Mouse(), new Cat()
11      };
12
13      for (int i = 0; i < 3; i++) {
14        for (GameCharacter c: characters) {
15          c.rankUp();
```

```
16            // c.exp = 0; // 先頭の//を取る    とエラーになる
17            System.out.println(c.info());
18          }
19        }
20      }
21  }
```

(i)〜(v) の空欄を埋めてこれらのクラスの定義を完成させよ。(i), (iv) の解答は、ソースプログラム中の「エラーにならない」「**先頭の//を取る**　とエラーになる」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）

(vii) `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。）

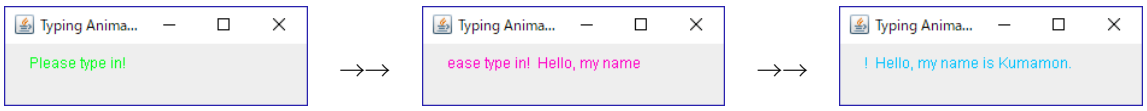
V. 下のプログラムは、キーボードから打ち込まれた文字を画面に表示し、文字の色が変化するアニメーションを表示する Java GUI アプリケーションである。ただし、画面に表示される文字を、最後に打ち込んだ 30 文字になるように制限して、古い文字は消えていくようになっている。さらに、空白を入力すると、色の変化がとまり、空白以外の文字を入力すると色の変化が再開するようになっている。キーイベントには、匿名クラスで対処する。

ファイル: TypingAnimation.java

```
1  import java.awt.*;
2  import java.awt.event.*;
3
4  import javax.swing.*;
5
6  public class TypingAnimation extends JPanel (i) {
7    private static final long serialVersionUID = 1L;
8
9    private String text = "Please␣type␣in!␣";
10   private float t = 0;
11   private Thread mythrd;
12
13   public void startThread() {
14     if (mythrd == null) { // 念のためチェック
15       mythrd = new Thread(this);
16       mythrd.start();
17     }
18   }
19
20   public void stopThread() {
21     mythrd = null;
22   }
23
24   public TypingAnimation() {
25     setPreferredSize(new Dimension(250, 50));
26     setFocusable(true);
27     addKeyListener(new KeyListener() {
28       public void keyPressed(KeyEvent e) {}
29       public void keyReleased(KeyEvent e) {}
30       public void keyTyped(KeyEvent e) {
31         char c = e.getKeyChar();
32         text += c;
33         if (text.length() > 30) {
34           text = text.substring(text.length() - 30);
35         }
36         if (c == '␣') {
37           repaint();
38           stopThread();
39         } else {
40           startThread();
41         }
42       }
```

```
43      });
44      startThread();
45    }
46
47    @Override
48    public void paintComponent(Graphics g) {
49      super.paintComponent(g);
50      g.setColor(Color.getHSBColor(t, (float)1.0, (float)1.0));
51      g.drawString(text, 20, 20);
52    }
53
54    public void run() {
55      Thread me = Thread.currentThread();
56      while (          (ii)          ) {
57        repaint();  // paintComponent を間接的に呼出す
58        try {
59          Thread.sleep(200); // 200 ミリ秒お休み
60        } catch (InterruptedException e) {}
61        t += 0.01;
62      }
63    }
64
65    public static void main(String[] args) {
66      SwingUtilities.invokeLater(() -> {
67        JFrame frame = new JFrame("Typing Animation");
68        frame.add(new TypingAnimation());
69        frame.pack();
70        frame.setVisible(true);
71        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
72      });
73    }
74  }
```

次にスクリーンショットを示す。



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。

以下に参考のために授業配布プリントの KeyTest.java, LeftRightButton2.java, LeftRightButton4.java, Guruguru.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 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;
    }
    System.err.printf("key␣=␣%d%n", k);
    repaint();
  }
  public void keyReleased(KeyEvent e) {}
  public void keyPressed(KeyEvent e) {}

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
      JFrame frame = new JFrame("Key␣Test");
      frame.add(new KeyTest());
      frame.pack();
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    });
  }
}
```

LeftRightButton2.java

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

public class LeftRightButton2 extends JPanel {
  private int x = 20;

  public LeftRightButton2() {
    setPreferredSize(new Dimension(200, 70));
    JButton lBtn = new JButton("Left");
    JButton rBtn = new JButton("Right");
    lBtn.addActionListener(new LeftListener());
    rBtn.addActionListener(new RightListener());
    setLayout(new FlowLayout());
    add(lBtn); add(rBtn);
  }
```

```
  private class LeftListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      x -= 10; repaint();
    }
  }

  private class RightListener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
      x += 10; repaint();
    }
  }

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

  public static void main(String[] args) {
    SwingUtilities.invokeLater(() -> {
      JFrame frame = new JFrame("ボタン");
      frame.add(new LeftRightButton2());
      frame.pack();
      frame.setVisible(true);;
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    });
  }
}
```

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) {
    SwingUtilities.invokeLater(() -> {
      JFrame frame = new JFrame("ボタン");
      frame.add(new LeftRightButton4());
      frame.pack();
      frame.setVisible(true);;
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    });
```

```
  }
}
```

Guruguru.java

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

public class Guruguru extends JPanel implements Runnable {
  private int r = 50, 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) {
    super.paintComponent(g); // スーパークラスの paintComponent を呼び出す
    // 全体を背景色で塗りつぶす。
    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) {
    SwingUtilities.invokeLater(() -> {
      JFrame frame = new JFrame("ぐるぐる!");
      frame.add(new Guruguru());
      frame.pack();
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    });
  }
}
```

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（2017 年 07 月 28 日）

学籍番号		氏名	

I. （3 × 3）

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

II. （6 × 4）

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

III. （4 × 4）

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

IV. （4, 3, 3, 4, 3, 6）

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

（裏面に続く）

(vi).	

V. （4, 4）

(i).	
(ii).	

授業・テストの感想

……

……

……

……

……

……

……

……

……

……

……

……

……

……

……