学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/4/21

問 1　大きさ n の整数配列 data 中のデータを小さい順に並べる泡立ち法のアルゴリズムを示しなさい。また、要素の比較が何回行われるかを示しなさい。さらに、$n = 4$ の場合に、比較回数が正しいことを確認しなさい。

解答例

Algorithm 1 泡立ち法

```
for j = n − 1 to 1 do
  for i = 0 to j − 1 do
    if data[i] > data[i + 1] then
      t = data[i]
      data[i]=data[i + 1]
      data[i + 1] = t
    end if
  end for
end for
```

また、比較の回数は

$$\sum_{j=n-1}^{1} j = \frac{n\,(n-1)}{2}$$

である。

$n = 4$ の場合は、初回の比較が 3 回、次が 2 回、最後が 1 回の合計 6 回の比較が必要となる。上式に代入すると

$$\frac{4 \times 3}{2} = 6$$

となり、正しいことがわかる。

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/4/28

問 1　`Comparable` インターフェースの拡張であるクラステンプレート `T` に対応した `MergeSort` クラスには、`Student` クラスが出てこないことを確認しなさい。

問 2　`MergeSort` クラス中の `less()` メソッドをここに記載しなさい。

解答例

```
1    private boolean less(int i, int j) {
2        return (list.get(i).compareTo(list.get(j)) < 0);
3    }
```

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/5/12

問 1　抽象クラス `AbstractSort` の拡張として、泡立ち法によるクラス `BubbleSort` を実装しなさい。

解答例

ソースコード 1　`BubbleSort` クラス

```
1  public class BubbleSort<T extends Comparable<T>>
2          extends AbstractSort<T>{
3
4      public BubbleSort(List<T> list) {
5          super(list);
6      }
7
8      /**
9       * 整列の実行
10      *
11      * @return 整列済みのリスト
12      */
13     @Override
14     public List<T> sort() {
15         int n = list.size();
16         for (int i = n - 1; i > 1; i--) {
17             for (int j = 0; j < i; j++) {
18                 if (!less(j, j + 1)) {
19                     swap(j, j + 1);
20                 }
21             }
22         }
23         return list;
24     }
25
26 }
```

問 2　抽象クラス `AbstractSort` の拡張として、セレクションソートを実行するクラス `SelectionSort` を実装しなさい。

解答例

ソースコード 2　`SelectionSort` クラス

```
1  public class SelectionSort<T extends Comparable<T>>
2          extends AbstractSort<T> {
3
4      public SelectionSort(List<T> list) {
5          super(list);
6      }
7
8      @Override
9      public List<T> sort() {
10         int n = list.size();
11         for(int i=0;i<n-1;i++){
12             int m = findMinimum(i,n);
13             if(i!=m)swap(i,m);
14         }
15         return list;
16     }
17
18     private int findMinimum(int from, int to) {
19         int m = from;
20         T min = list.get(m);
21         for(int i=m;i<to;i++){
22             T t = list.get(i);
23             if(t.compareTo(min)<0){
24                 min = t;
25                 m = i;
26             }
27         }
28         return m;
29     }
30 }
```

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/5/19

問 1　`example0/SampleRunnable.java` について考える。

1. スレッドの開始

   ```
   new Thread(new SampleRunnable(1)).start();
   ```

   によって、どのメソッドが実行されるか？
2. どのメソッド内でどの変数の値が何になると停止するか？

解答例

1. `run()` メソッドを実行する。`run()` メソッド内の `while` ループにより、`update()` を繰り返し実行する。
2. `update()` メソッドは、変数 `c` を一つずつ増やす。`c=10` となると、`runnning=false` となり、`run()` メソッドは終了する。

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/5/26

問 1　以下のプログラム中のメソッド `sumAll` は、引数で渡されたリスト `input` の各要素に対して、`function` で定義された計算を行い、その和を返す。呼び出し側 (18 行目) の第二引数に Lambda 式を記入し、線形和、二乗和が計算できることを確かめなさい。

```
1  package minitest;
2
3  import java.util.Arrays;
4  import java.util.List;
5  import java.util.function.DoubleFunction;
6
7  public class MiniTest {
8
9      public static double sumAll(
10             List<Double> input, DoubleFunction<Double> function) {
11         double sum = input.stream().map(d -> function.apply(d)).
12                 reduce(0., (accumulator, _item) -> accumulator + _item);
13         return sum;
14     }
15
16     public static void main(String[] args) {
17         Double d[]={1.,2.,3.};
18         double r = sumAll(Arrays.asList(d), );
19         System.out.println(r);
20     }
21
22 }
```

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/6/2

問 1　ソースコード 1 は、正規表現を用いて、”0”と”1”からなる文字列から偶数個連続する”0”を選び列挙するプログラムである。ここで k の値を 0 から 2 まで変化させ、結果を観察し、何が起こっているかを記述しなさい。

ソースコード 1　正規表現

```
1  import java.util.regex.Matcher;
2  import java.util.regex.Pattern;
3
4  public class Simple {
5
6      public static void main(String[] args) {
7          String input = "1100100100001";
8          Pattern p = Pattern.compile("1((00)+)1");//正規表現の定義
9          Matcher m = p.matcher(input);
10         while (m.find()) {
11             System.out.println(m.group());
12             k++;
13         }
14     }
15 }
```

解答例　初めに、出力結果を示す。

```
1001
100001
```

正規表現 1((00)+)1 は、0 が偶数個連続した前後に 1 がある文字列である。match(0) は、入力文字列の先頭から 2 文字目で始まるパターンに相当している。match(1) は、入力文字列の先頭から 8 文字目で始まるパターンに相当している。

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/6/9

問 1　講義ではテキストファイルを BufferedReader/BufferedWriter を用いて読み書きした。しかし、バイナリファイルの場合は、FileInputStream/FileOutputStream を使って、一バイト毎に読み書きする必要があります。つまり FileInputStream から `read()` メソッドを用いて一バイト読み、FileOutputStream へ `write(int)` メソッドを用いて一バイト書く形式です。

配布ファイルの `fileCopy/BinaryFileCopy.java` はバイナリファイルをコピーするプログラムです。実際のコピーを行っているメソッド `copyData()` を完成させ、動作を確認しなさい。

解答例

```
1     public int copyData() throws IOException {
2         int n = 0;
3         int b;
4         //一バイト毎にコピー
5         while ((b = in.read()) != -1) {
6             n++;
7             out.write(b);
8         }
9         in.close();
10        out.close();
11        return n;
12    }
```

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/6/16

問 1　講義で示した例題のなかで、ReadURL クラス中の readHeaders() メソッドを考える。正規表現を定義する際に、複数行 (MULTILINE) と大文字小文字の違いを無視 (CASE_INSENSITIVE) が指定されている。この理由を述べよ。

解答例　HTML 中では、ヘッダの開始<h1>とヘッダの終了</h1>が必ずしも同一行とは限らない。つまり、例えば

```
<h1>
最初のヘッダ
</h1>
```

のような記法が許されている。このため、複数行 (MULTILINE) にわたって正規表現によるパターン探索を行う必要がある。

また、ヘッダは<h1>という小文字でも、<H1>という大文字でも良い。従って、正規表現によるパターン検索においても、大文字小文字の差を無視する CASE_INSENSITIVE が必要となる。

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/6/30

問 1　`guiWithoutAction` にある `selectColors` というメニューを参考に新しいメニューを作成しなさい。初めに、列挙型 (`enum`) として、線の種類 `LineTypes` を定義、3 つの種類 (`SOLID`、`DOTTED`、`DASHED`) を定義しなさい。さらに、これを新しいメニュー `selectLineTypes` に追加しなさい。

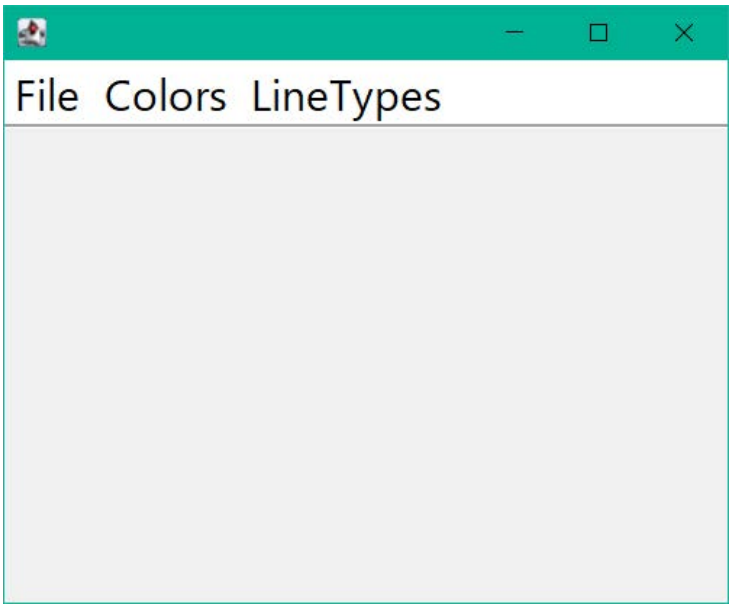
解答例　`guiWithoutAction` に課題のメニューを追加したクラスを `MainMiniTest` として作成した。

ソースコード 1　`MainMiniTest.java`

```
1  public class MainMiniTest extends javax.swing.JFrame {
2
3      public enum Colors {//色を列挙型として定義
4          ORANGE(Color.ORANGE), YELLOW(Color.YELLOW), GREEN(Color.
               GREEN);
5          private final Color color;
6
7          Colors(Color color) {
8              this.color = color;
9          }
10
11         public Color getColor() {
12             return color;
13         }
14     }
15
16     public enum LineTypes{
17         SOLID,DOTTED,DASHED
18     }
19
20     /**
21      * Creates new form MainWithJMenu
22      */
23     public MainMiniTest() {
24         initComponents();
```

```
25            Font font = new Font("MS␣UI␣Gothic", 0, 24);
26            for (Colors m : Colors.values()) {
27                JMenuItem item = new JMenuItem(m.toString());
28                item.setFont(font);
29                selectColors.add(item);
30            }
31            for (LineTypes lt:LineTypes.values()){
32                JMenuItem item = new JMenuItem(lt.toString());
33                item.setFont(font);
34                selectLineTypes.add(item);
35            }
36        }
37
38        ... 途中省略
39
40         // Variables declaration - do not modify
41        private javax.swing.JMenuItem exit;
42        private javax.swing.JMenu fileMenu;
43        private javax.swing.JMenuBar menuBar;
44        private javax.swing.JMenu selectColors;
45        private javax.swing.JMenu selectLineTypes;
46        // End of variables declaration
47    }
```

実行すると図のような GUI を表示する。



学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/7/7

問 1　fileChoose/fileChooseMain.java で、テキストを表示するフォントを変更する機能を追加しよう。JTextArea クラスのインスタンス textArea で使用しているフォントは、

```
Font font = textArea.getFont();
```

で取得することができる。フォントは、名前、スタイル、サイズの三つの要素で記述する。

```
String fontName = font.getFontName();
int fontStyle = font.getStyle();
int fontSize = font.getSize();
```

フォントのスタイルには、Font.PLAIN、Font.BOLD、Font.ITALIC の 3 種類がある。フォントを設定するためには

```
textArea.setFont(new Font(fontName,fontStyle,fontSize));
```

とする。

fileChoose/fileChooseMain.java にフォントスタイルを変更するメニューを追加し、動作を確認しなさい。

解答例

1. はじめに、メニューバーに新たなメニューとして selectFontStyle を追加する。
2. 三つのフォントスタイルに対応する列挙型 FontStyle を定義する。これに対応して、selectFontStyle に、各フォントスタイルのエントリーを追加し、対応する動作 setFontStyle() を登録する。
3. setFontStyle() は、引数で指定したフォントスタイルを、テキストエリアに設定する。

ソースコード 1　`FileChooseMainMiniTest.java`

```
1  public class FileChooseMainMiniTest extends javax.swing.JFrame {
2
3      private File dir;
4      private String filename = null;
5      private final String applicationName;
6
7      //
8      public enum FontStyle {
9          PLAIN(Font.PLAIN), BOLD(Font.BOLD), ITALIC(Font.ITALIC);
10         int fontStyle;
11
12         FontStyle(int f) {
13             fontStyle = f;
14         }
15
16         int getStyle() {
17             return fontStyle;
18         }
19     }
20
21     /**
22      * Creates new form FileChooseMain
23      */
24     public FileChooseMainMiniTest() {
25         initComponents();
26         for(FontStyle t:FontStyle.values()){
27             JMenuItem item = new JMenuItem(t.toString());
28             selectFontStyle.add(item);
29             item.addActionListener(e->setFontStyle(t.getStyle()));
30         }
31         applicationName = FileChooseMainMiniTest.class.getSimpleName();
32         setTitle(applicationName);
33     }
34
35     private void setFontStyle(int fontType){
36         Font font = textArea.getFont();
37         textArea.setFont(new Font(font.getFontName(),fontType,font.getSize()));
38     }
```

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/7/14

問 1　`ColorFrame` クラスのコンストラクタ内で、`ChangeEvent` を受け取った `colorPanel` の動作は、以下のように記載している。

```
1        colorPanel.addChangeListener((ChangeEvent e) -> {
2            Color c = colorPanel.getColor();
3            System.out.println(c.toString());
4        });
```

この部分を書き直し、`Colors` クラスから色を取得してた色を `colorPanel` へ設定する処理を追記し、動作させなさい。

解答例　`colorPanel` と `showColor` に色を設定する例を示す。

```
1        colorPanel.addChangeListener((ChangeEvent e) -> {
2            Color c = colorPanel.getColor();
3            colorPanel.setBackground(c);
4            showColor.setBackground(c);
5        });
```

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/7/21

問 1　講義で扱った shapeSamples について、多角形を追加して描こう。多角形を描くのには、java.awt.geom.Path2D.Double をいる。java.awt.geom.Path2D.Double の使い方は、以下の通りである。

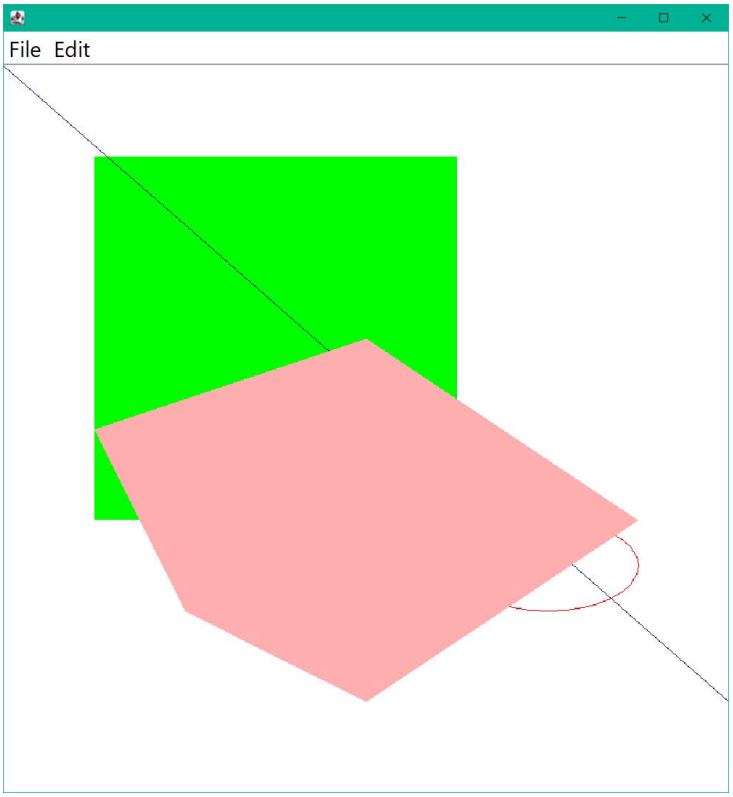
- コンストラクタ Path2D.Double()：新たなパスを開始する
- moveTo(double x,double y)：倍精度で指定された座標に移動して点をパスの最初の点とすする。
- lineTo(double x,double y)：現在の座標から倍精度で新しく指定された座標まで直線を描画して点をパスに追加
- closePath()：最後の moveTo の座標まで直線を描画して現在のサブパスを閉じる

最後にこのパスを描く (draw()) する、または塗りつぶす (fill())。

解答例 drawSamples() メソッドに追加した、多角形を描画する部分を示す。

```
1        //多角形
2        g.setColor(Color.PINK);
3        Path2D.Double path = new Path2D.Double();
4        double x[]={100.,400.,700.,400.,200.};
5        double y[]={400.,300.,500.,700.,600.};
6        path.moveTo(x[0],y[0]);
7        for(int i=1;i<x.length;i++){
8            path.lineTo(x[i], y[i]);
9        }
10       path.closePath();
11       g.fill(path);
```

実行すると、図のような 5 角形となる。


File Edit

学籍番号									氏名	

学籍番号と氏名は丁寧に記載すること

「オブジェクト指向プログラミング特論」ミニテスト

2020/7/28

問 1　講義では、Euler 閉路を列挙列挙した。全ての頂点を一度ずつ経由して始点に戻る閉路を Hamilton 閉路という。Hamilton 閉路を列挙する再帰的アルゴリズムを Algorithm 1 に示す。r は探索の始点、L は経由した頂点のリストを表している。

配布しているこのアルゴリズムを実装するクラス `Hamilton` を完成させなさい。また、例である `sample/Graph3` を実行し、動作を確認しなさい。

Algorithm 1 Hamilton 閉路を求める再帰的アルゴリズム

procedure ENUMERATE(v, L)
  for all $a \in \delta v$ **do**
    $w = \partial a \setminus \{v\}$　　▷ a の v と反対側の頂点
    if $w = r \wedge |L| = |V|$ **then**
      見つけた閉路を保存
    else
      if $w \notin L$ **then**
        $L' = L \cup \{w\}$
        ENUMERATE(w, L')
      end if
    end if
  end for
end procedure

解答例　`enumerateSub()` の実装を示す。

```
1    private void enumerateSub(Node v, List<Node> path) {
2        List<Arc> arcs = g.getArcs(v);
3        if (arcs != null) {
4            for (Arc a : arcs) {
```

```
                Node w = a.getAnotherEnd(v);
                if (w == start && path.size() == N) {
                    curcuitList.add(path);
                } else {
                    if (!path.contains(w)) {
                        List<Node> newPath = Utils.createList();
                        newPath.addAll(path);
                        newPath.add(w);
                        enumerateSub(w, newPath);
                    }
                }
            }
        }
    }
```

実行すると、以下の 4 つの閉路を見つけることができた。

```
[0, 1, 3, 4, 2]
[0, 1, 4, 3, 2]
[0, 2, 3, 4, 1]
[0, 2, 4, 3, 1]
```