



グラフを表すデータ構造  
JAVAでの実装

# なぜJAVAを使うか

- グラフの実装
  - 頂点、弧及びその関連を記述する
  - 頂点の数、弧の数を柔軟に変える必要あり
  - グラフ探索など、リンクをたどる必要あり
- オブジェクト指向言語が向いている
  - オブジェクト数の柔軟な変更
  - 再帰的関数・メソッド



- リストなどの豊富なライブラリ
  - java.util.Vectorなど
- 使い易い開発環境
  - プロジェクト管理、クラス管理
    - GUIによる支援
  - プログラミング支援
    - メソッド名補完、javadoc
- グラフィックスを容易に作成可能
  - javax.swingコンポーネント
- OSに依らずにプログラミング可能
- 豊富な文献



# JAVA開発環境

- NetBeans

- <http://www.netbeans.org/>
- <http://www.netbeans.jp/>
- GUI開発が容易
- UMLとの連携ができる

- Eclipse

- <http://www.eclipse.org/>

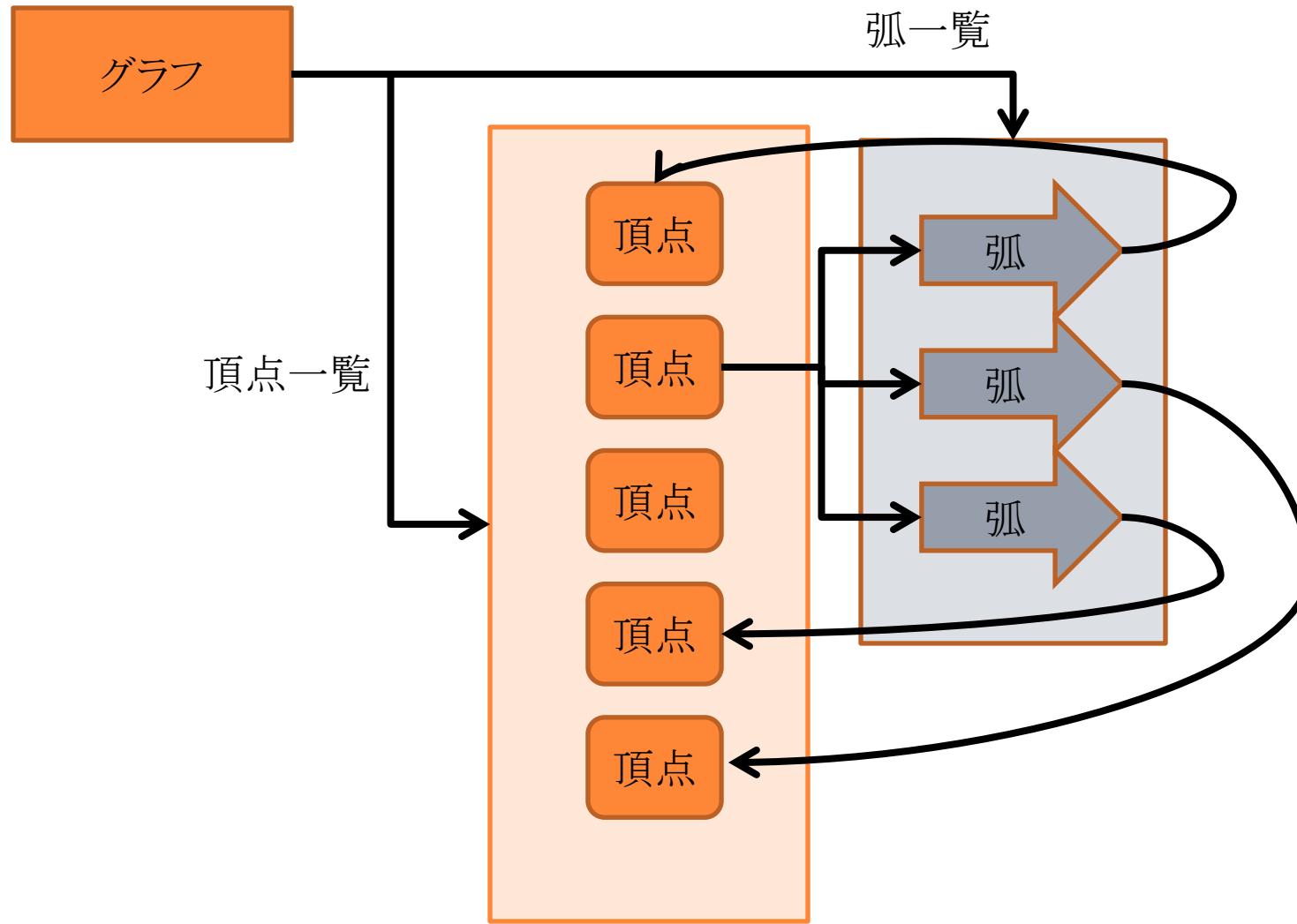


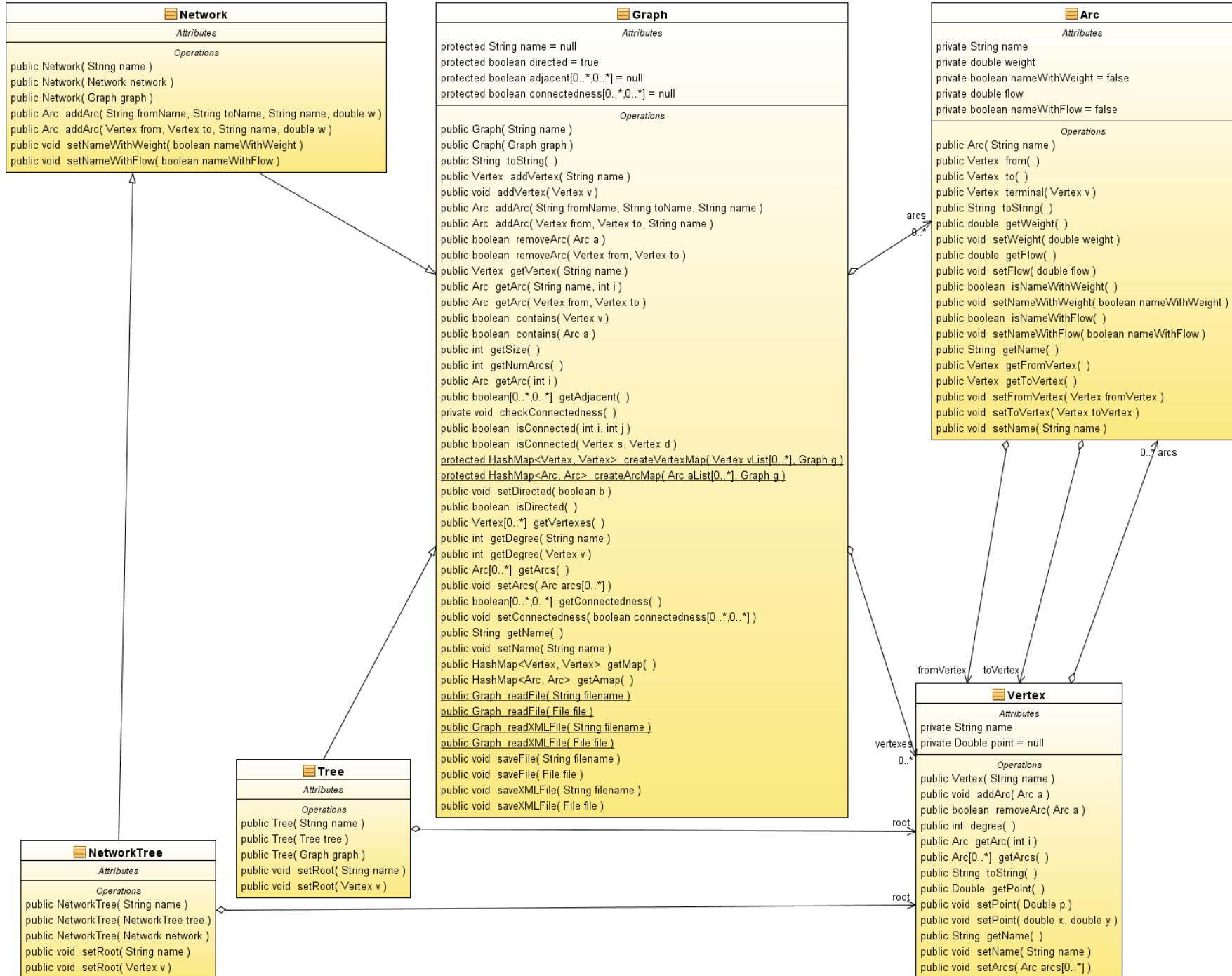
# グラフの構造を記述するクラス

- パッケージgraphLib
- graphLib.graph
  - グラフ全体
- graphLib.Vertex
  - 頂点
  - 頂点を始点とする弧
  - 頂点を2次元面に表示するための座標
- graphLib.Arc
  - 弧の始点と終点



# クラスの関係





## GRAPHクラス: フィールド

```
/** グラフのラベル*/
protected String name = null;
/** 頂点の一覧 */
protected Vector<Vertex> vertexes = null;
/** 弧の一覧 */
protected Vector<Arc> arcs = null;
/** 有向グラフであるか */
protected boolean directed = true;
/**隣接行列 */
protected boolean adjacent[][] = null;
protected boolean connectedness[][] = null;
private HashMap<Vertex, Vertex> map = null;
private HashMap<Arc, Arc> amap = null;
```

# GRAPHクラス: 主要メソッド

public Graph(String name)	コンストラクタ
public Graph(final Graph graph)	コピー・コンストラクタ
public Vertex addVertex(String name)	頂点追加
public void addVertex(Vertex v)	頂点追加
public Arc addArc(String fromName, String toName, String name)	弧追加
public Arc addArc(Vertex from, Vertex to, String name)	弧追加
public void setDirected(boolean b)	有向・無向を設定
public Vector<Vertex> getVertices()	頂点一覧を取得
public Vector<Arc> getArcs()	弧一覧を取得



## VERTEXクラス:フィールド

```
/** 頂点のラベル */
private String name;
/** 頂点を始点とする弧のリスト */
private Vector<Arc> arcs = null;
/** 頂点の位置座標 */
private Point2D.Double point = null;
```

# VERTEXクラス: 主要メソッド

public Vertex(String name)	コンストラクタ
public void addArc(Arc a)	頂点を始点とする弧を追加
public int degree()	次数を返す
public Vector<Arc> getArcs()	頂点を始点とする弧の一覧を取得
public Point2D.Double getPoint()	頂点の座標を取得
public void setPoint(Point2D.Double p)	頂点の座標を設定



## ARCクラス:フィールド

```
/** 始点 */
private Vertex fromVertex = null;
/** 終点 */
private Vertex toVertex = null;
/** 弧のラベル */
private String name;
/** 弧の重み */
private double weight;
private boolean nameWithWeight = false;
/** 弧を流れる流量 */
private double flow;
private boolean nameWithFlow = false;
```



## ARCクラス: 主要メソッド

public Arc(String name)	コンストラクタ
public Vertex from()	弧の始点を返す
public Vertex to()	弧の終点を返す
public Vertex terminal(Vertex v)	弧の反対側の頂点を返す



# グラフの定義

- graphLib.Graphを拡張

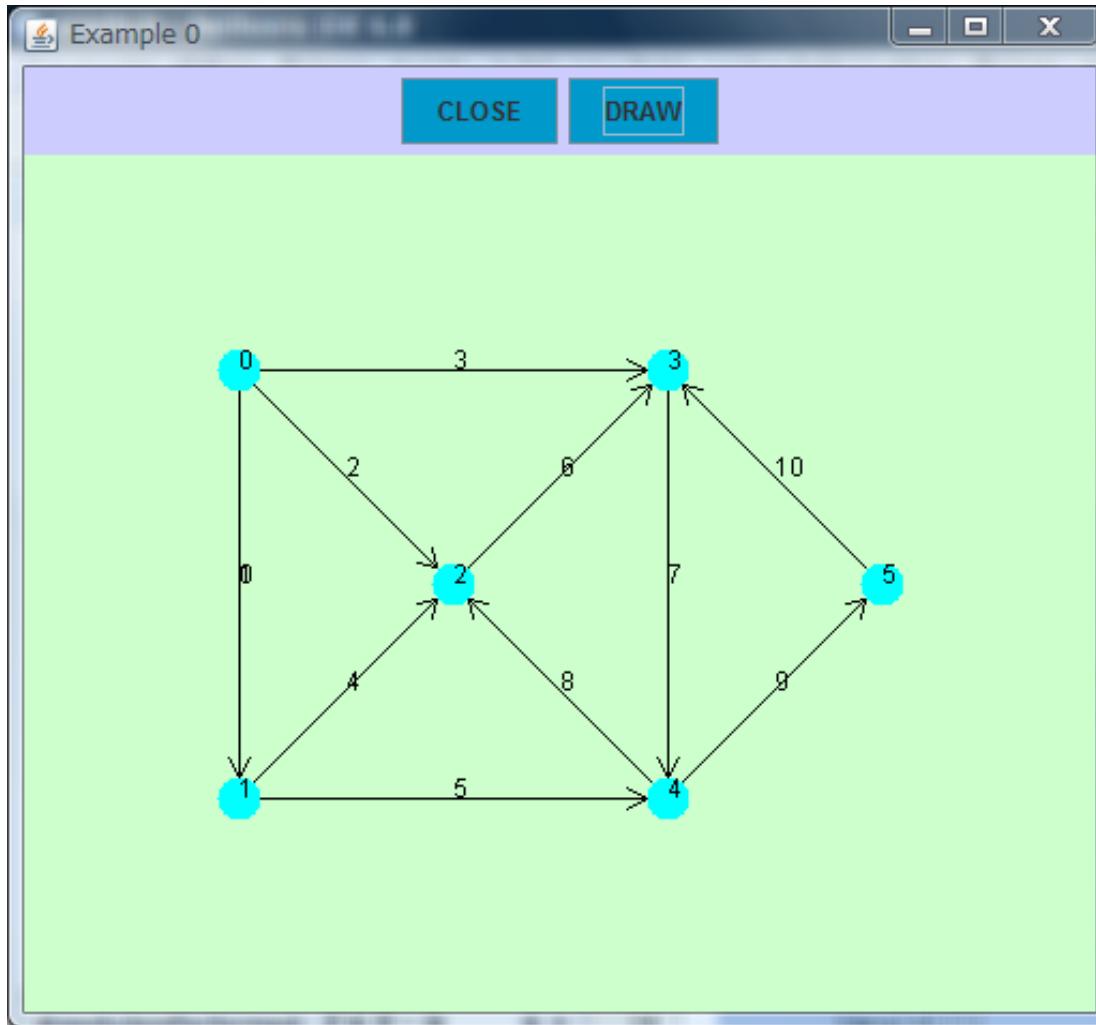
```
public class Graph0 extends Graph {  
  
    public Graph0(String name) {  
        super(name);  
        int n = 6;  
        //頂点の生成  
        graphLib.Vertex vList[] = new graphLib.Vertex[n];  
        for (int i = 0; i < n; i++) {  
            vList[i] = new graphLib.Vertex(String.valueOf(i));  
            addVertex(vList[i]);  
        }  
        //頂点の表示座標を設定  
        double d = 100.;  
        vList[0].setPoint(d, d);  
        ....  
        int k = 0;  
        addArc(vList[0], vList[1], String.valueOf(k)); k++;  
        addArc(vList[0], vList[1], String.valueOf(k)); k++;  
    }  
}
```



# グラフ表示のメインプログラム

```
/**  
 * 表示の開始  
 * @param evt 表示開始のボタンイベント  
 */  
private void actionPerformed(java.awt.event.ActionEvent evt) {  
    graphExample0 = new graphSamples.Graph0(title);  
    /** グラフの定義 */  
    panel.setGraph(graphExample0);  
    /** グラフをパネルに設定 */  
    panel.mkImage0();  
    /** 作図イメージ生成 */  
    setTitle(graphExample0.toString());  
    /** タイトル設定 */  
    repaint();  
    /** 全体再描画 */  
}
```





## Graph0.java

```
package graphSamples;

import graphLib.*;

/**
 *
 * @author tadaki
 */
public class Graph0 extends Graph {

    public Graph0(String name) {
        super(name);
        int n = 6;
        //頂点の生成
        graphLib.Vertex vList[] = new graphLib.Vertex[n];
        for (int i = 0; i < n; i++) {
            vList[i] = new graphLib.Vertex(String.valueOf(i));
            addVertex(vList[i]);
        }
        //頂点の表示座標を設定
        double d = 100.;
        vList[0].setPoint(d, d);
        vList[1].setPoint(d, 3 * d);
        vList[2].setPoint(2 * d, 2 * d);
        vList[3].setPoint(3 * d, d);
        vList[4].setPoint(3 * d, 3 * d);
        vList[5].setPoint(4 * d, 2 * d);
        //弧の定義
        int k = 0;
        addArc(vList[0], vList[1], String.valueOf(k));
        k++;
        addArc(vList[0], vList[1], String.valueOf(k));
        k++;
        addArc(vList[0], vList[2], String.valueOf(k));
        k++;
        addArc(vList[0], vList[3], String.valueOf(k));
        k++;
        addArc(vList[1], vList[2], String.valueOf(k));
        k++;
        addArc(vList[1], vList[4], String.valueOf(k));
        k++;
        addArc(vList[2], vList[3], String.valueOf(k));
        k++;
        addArc(vList[3], vList[4], String.valueOf(k));
    }
}
```

## Graph0.java

```
k++;
addArc(vList[4], vList[2], String.valueOf(k));
k++;
addArc(vList[4], vList[5], String.valueOf(k));
k++;
addArc(vList[5], vList[3], String.valueOf(k));
k++;
addArc(vList[5], vList[5], String.valueOf(k));
}
}
```

## Main.java

```
package example0;

/**
 * 例題
 * @author tadaki
 */
public class Main extends javax.swing.JFrame {

    private graphLib.Graph graphExample0 = null;
    private String title = "Example0";

    /** Creates new form Main */
    public Main(String title) {
        initComponents();
        this.title = title;
    }

    /** This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void initComponents() {

        buttons = new javax.swing.JPanel();
        close = new javax.swing.JButton();
        draw = new javax.swing.JButton();
        panel = new graphDraw.graphPanel();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

        buttons.setBackground(new java.awt.Color(204, 204, 255));

        close.setBackground(new java.awt.Color(0, 153, 204));
        close.setText("CLOSE");
        close.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                closeActionPerformed(evt);
            }
        });
        buttons.add(close);

        javax.swing.GroupLayout buttonsLayout = new javax.swing.GroupLayout(buttons);
        buttons.setLayout(buttonsLayout);
        buttonsLayout.setHorizontalGroup(
            buttonsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(buttonsLayout.createSequentialGroup()
                .addGap(0, 0, 0)
                .addComponent(panel))
        );
        buttonsLayout.setVerticalGroup(
            buttonsLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(buttonsLayout.createSequentialGroup()
                .addGap(0, 0, 0)
                .addComponent(panel))
        );

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(0, 0, 0)
                .addComponent(buttons))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(0, 0, 0)
                .addComponent(buttons))
        );
    }

    /**
     * This method is called from within the constructor to
     * initialize the form.
     * WARNING: Do NOT modify this code. The content of this method is
     * always regenerated by the Form Editor.
     */
    // <editor-fold defaultstate="collapsed" desc="Generated
Code">//GEN-BEGIN:initComponents
    private void closeActionPerformed(java.awt.event.ActionEvent evt) {
        System.exit(0);
    }
}
```

## Main.java

```
        draw.setBackground(new java.awt.Color(0, 153, 204));
        draw.setText("DRAW");
        draw.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                drawActionPerformed(evt);
            }
        });
        buttons.add(draw);

        getContentPane().add(buttons, java.awt.BorderLayout.NORTH);

        panel.setPreferredSize(new java.awt.Dimension(500, 400));
        getContentPane().add(panel, java.awt.BorderLayout.CENTER);

        pack();
}// </editor-fold>//GEN-END: initComponents

/**
 * 表示の開始
 * @param evt 表示開始のボタンイベント
 */
private void drawActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_drawActionPerformed
    graphExample0 = new graphSamples.Graph0(title);
    /** グラフの定義 */
    panel.setGraph(graphExample0);
    /** グラフをパネルに設定 */
    panel.mkImage();
    /** 作図イメージ生成 */
    setTitle(graphExample0.toString());
    /** タイトル設定 */
    repaint();
    /** 全体再描画 */
};//GEN-LAST:event_drawActionPerformed

/**
 * 終了
 * @param evt 終了のボタンイベント
 */
private void closeActionPerformed(java.awt.event.ActionEvent evt)
{//GEN-FIRST:event_closeActionPerformed
    dispose();
};//GEN-LAST:event_closeActionPerformed
```

## Main.java

```
/**  
 * 開始  
 * @param args the command line arguments  
 */  
public static void main(String args[]) {  
    java.awt.EventQueue.invokeLater(new Runnable() {  
  
        @Override  
        public void run() {  
            new Main("Example 0").setVisible(true);  
        }  
    });  
}  
// Variables declaration – do not modify//GEN-BEGIN:variables  
private javax.swing.JPanel buttons;  
private javax.swing.JButton close;  
private javax.swing.JButton draw;  
private graphDraw.graphPanel panel;  
// End of variables declaration//GEN-END:variables  
}
```

## Student.java

```
package StudentSample2;

/**
 * 生徒のクラス
 * Comparableインターフェイスの例
 * @author tadaki
 */
public class Student implements Comparable<Student> {

    private String name = null; //名前
    private int studentID = 0; //学生番号
    private int record = 0; //点数

    /**
     * コンストラクタ
     * @param name 名前
     * @param studentID 学生番号
     */
    public Student(String name, int studentID) {
        this.name = new String(name);
        this.studentID = studentID;
    }

    /**
     * 学生番号取得
     * @return 取得した学生番号
     */
    public int getStudentID() {
        return studentID;
    }

    /**
     * 名前取得
     * @return 取得した名前
     */
    public String getName() {
        return name;
    }

    /**
     * 得点取得
     * @return 取得した得点
     */
    public int getRecord() {
```

## Student.java

```
        return record;
    }

    /**
     * 得点設定
     * @param record 設定する得点
     */
    public void setRecord(int record) {
        this.record = record;
    }

    @Override
    /**
     * Student インスタンスの比較
     * インターフェイスComparableで必須
     */
    public int compareTo(Student o) {
        int k = 1;
        if (this.getRecord() < o.getRecord()) {
            k = -1;
        }
        return k;
    }
}
```

## StudentRecord.java

```
package StudentSample2;

import java.util.Vector;
/**
 *
 * @author tadaki
 */
public class StudentRecord {

    private Vector<Student> students = null;//生徒一覧
    /** 名前一覧 */
    private String names[] = {
        "Aoyama", "Asou", "Baba", "Chou", "Egashira",
        "Eto", "Funaki", "Goto", "Gunji", "Hara", "Hashimoto",
        "Ikeuchi", "Ito", "Jo", "Kayama", "Mori", "Naito", "Tada",
        "Yamada", "Yoshida"
    };

    /** コンストラクタ */
    public StudentRecord() {
        //生徒一覧を初期化
        students = new Vector<Student>();
        ;
        //登録
        for (int i = 0; i < names.length; i++) {
            Student s = new Student(names[i], 1000 + i);
            s.setRecord((int) (100 * Math.random()));
            students.add(s);
        }
    }

    /**
     * 学生一覧印刷
     */
    public void listStudents() {
        //拡張されたforループ
        for (Student s : getStudents()) {
            System.out.print(String.valueOf(s.getStudentID())
                + ":" + s.getName() + ":");

            System.out.println(String.valueOf(s.getRecord()));
        }
    }

    /**

```

## StudentRecord.java

```
* 学生一覧取得
* @return 学生一覧のVector
*/
public Vector<Student> getStudents() {
    return students;
}

/**
 * ソートの実行
 * @param <T> Comparableインターフェイスを実装したクラス
 * @param t Vector<T>
 */
public static <T extends Comparable<T>> void sort(Vector<T> t) {
    for (int i = t.size(); i > 0; i--) {
        for (int j = 0; j < i - 1; j++) {
            if (t.get(j).compareTo(t.get(j+1)) > 0) {
                T c = t.get(j);
                t.set(j, t.get(j + 1));
                t.set(j + 1, c);
            }
        }
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    StudentRecord studentRecord = new StudentRecord();
    StudentRecord.sort(studentRecord.getStudents());
    studentRecord.listStudents();
}
}
```