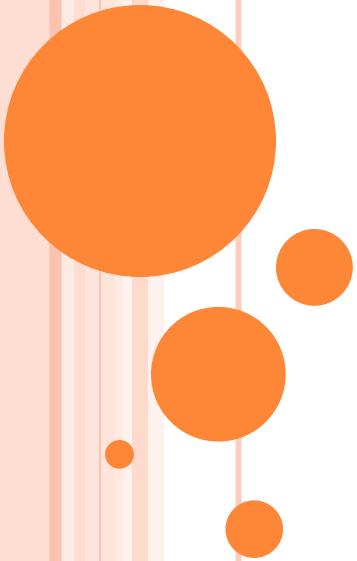
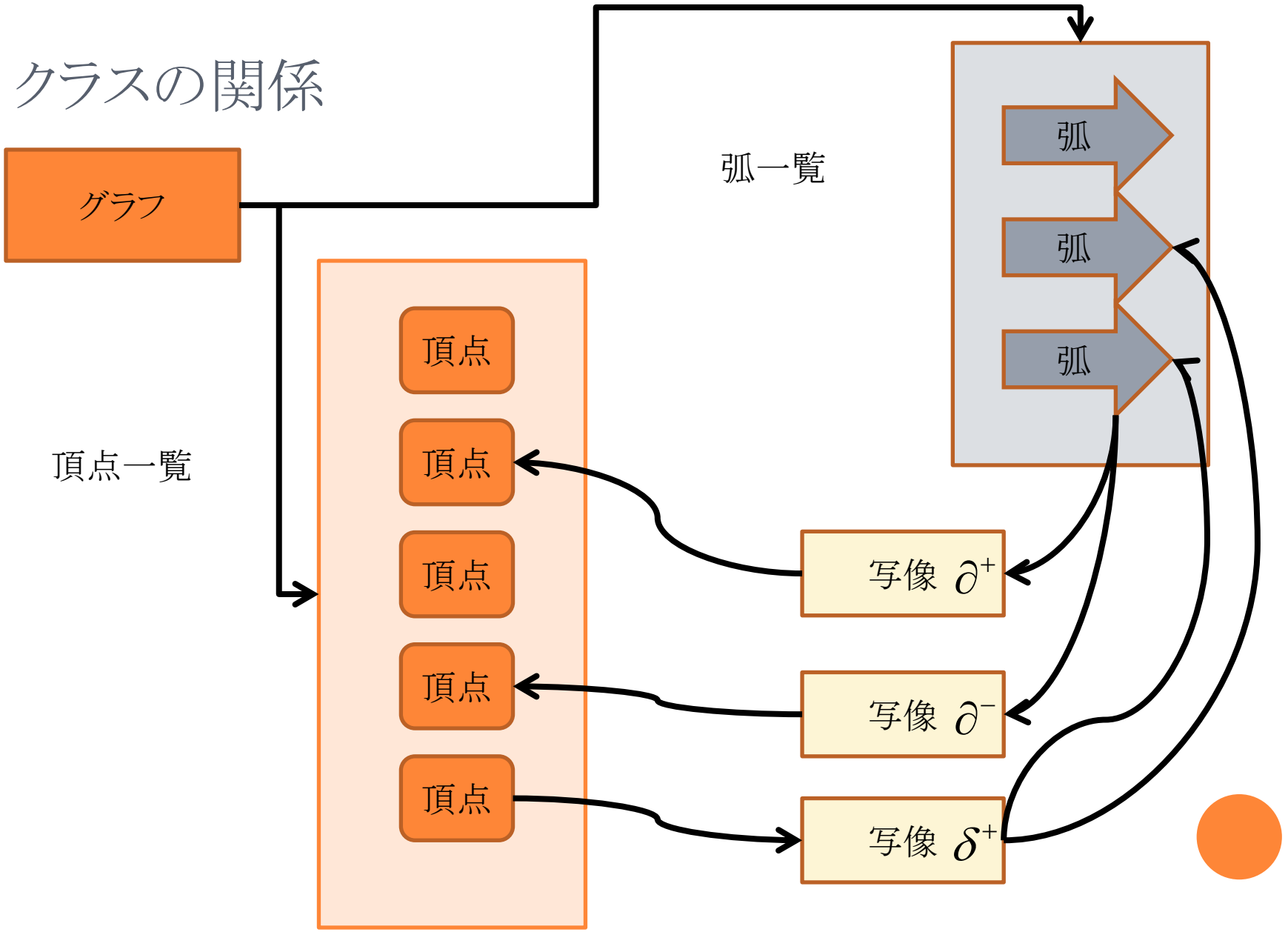


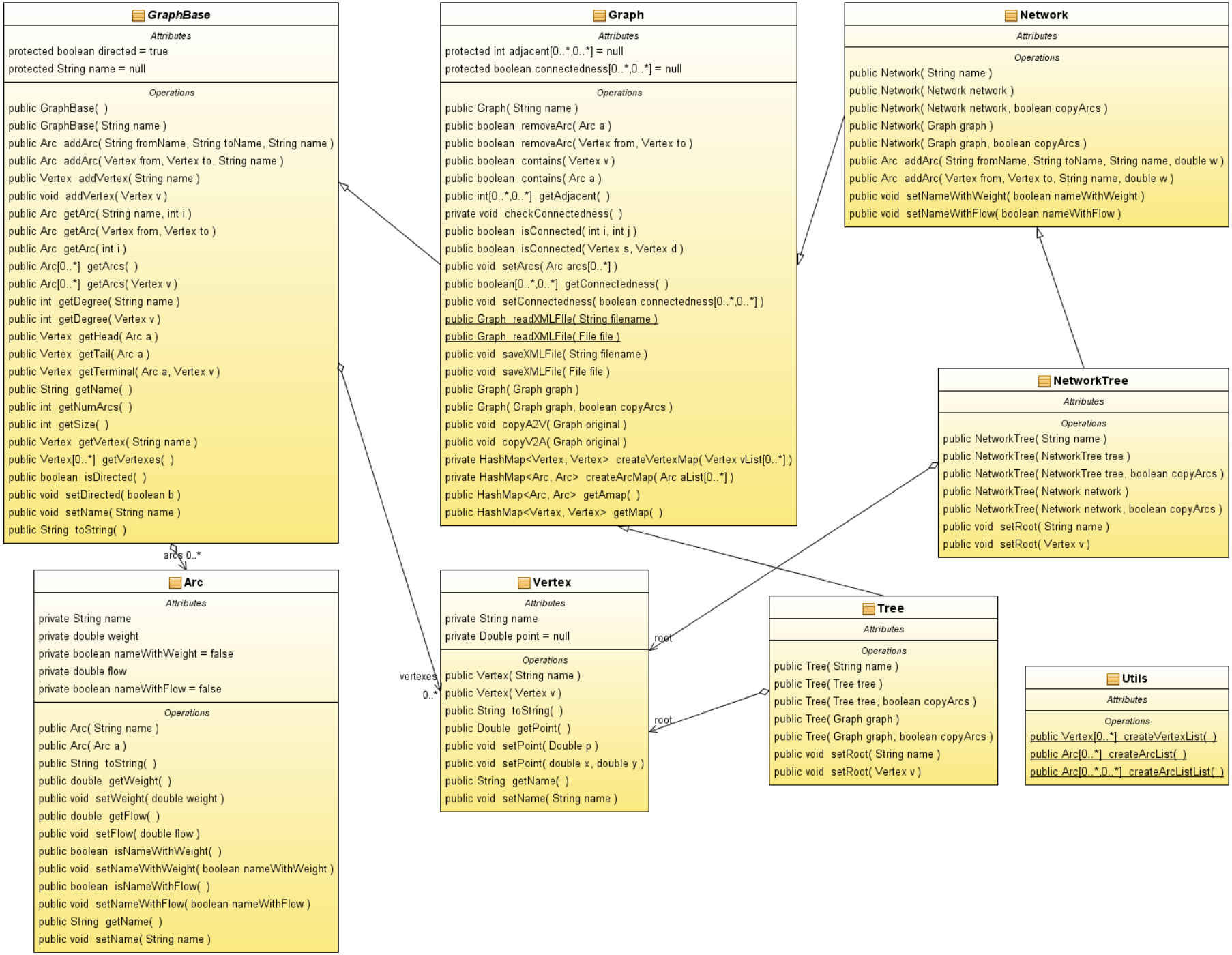
グラフを表すデータ構造
JAVAでの実装

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




GraphBase
Attributes
protected boolean directed = true
protected String name = null
Operations
public GraphBase( )
public GraphBase(String name)
public Arc addArc(String fromName, String toName, String name)
public Arc addArc(Vertex from, Vertex to, String name)
public Vertex addVertex(String name)
public void addVertex(Vertex v)
public Arc getArc(String name, int i)
public Arc getArc(Vertex from, Vertex to)
public Arc getArc(int i)
public Arc[0..*] getArcs( )
public Arc[0..*] getArcs(Vertex v)
public int getDegree(String name)
public int getDegree(Vertex v)
public Vertex getHead(Arc a)
public Vertex getTail(Arc a)
public Vertex getTerminal(Arc a, Vertex v)
public String getName( )
public int getNumArcs( )
public int getSize( )
public Vertex getVertex(String name)
public Vertex[0..*] getVertexes( )
public boolean isDirected( )
public void setDirected(boolean b)
public void setName(String name)
public String toString( )
Graph
Attributes
protected int adjacent[0..*,0..*] = null
protected boolean connectedness[0..*,0..*] = null
Operations
public Graph(String name)
public boolean removeArc(Arc a)
public boolean removeArc(Vertex from, Vertex to)
public boolean contains(Vertex v)
public boolean contains(Arc a)
public int[0..*,0..*] getAdjacent( )
private void checkConnectedness( )
public boolean isConnected(int i, int j)
public boolean isConnected(Vertex s, Vertex d)
public void setArcs(Arc arcs[0..*])
public boolean[0..*,0..*] getConnectedness( )
public void setConnectedness(boolean connectedness[0..*,0..*])
public Graph readXMLFile(String filename)
public Graph readXMLFile(File file)
public void saveXMLFile(String filename)
public void saveXMLFile(File file)
public Graph(Graph graph)
public Graph(Graph graph, boolean copyArcs)
public void copyA2V(Graph original)
public void copyV2A(Graph original)
private HashMap<Vertex, Vertex> createVertexMap(Vertex vList[0..*])
private HashMap<Arc, Arc> createArcMap(Arc aList[0..*])
public HashMap<Arc, Arc> getAmap( )
public HashMap<Vertex, Vertex> getMap( )
Network
Attributes
Operations
public Network(String name)
public Network(Network network)
public Network(Network network, boolean copyArcs)
public Network(Graph graph)
public Network(Graph graph, boolean copyArcs)
public Arc addArc(String fromName, String toName, String name, double w)
public Arc addArc(Vertex from, Vertex to, String name, double w)
public void setNameWithWeight(boolean nameWithWeight)
public void setNameWithFlow(boolean nameWithFlow)
NetworkTree
Attributes
Operations
public NetworkTree(String name)
public NetworkTree(NetworkTree tree)
public NetworkTree(NetworkTree tree, boolean copyArcs)
public NetworkTree(Network network)
public NetworkTree(Network network, boolean copyArcs)
public void setRoot(String name)
public void setRoot(Vertex v)
arcs 0..*
Arc
Attributes
private String name
private double weight
private boolean nameWithWeight = false
private double flow
private boolean nameWithFlow = false
Operations
public Arc(String name)
public Arc(Arc a)
public String toString( )
public double getWeight( )
public void setWeight(double weight)
public double getFlow( )
public void setFlow(double flow)
public boolean isNameWithWeight( )
public void setNameWithWeight(boolean nameWithWeight)
public boolean isNameWithFlow( )
public void setNameWithFlow(boolean nameWithFlow)
public String getName( )
public void setName(String name)
Vertex
Attributes
private String name
private Double point = null
Operations
public Vertex(String name)
public Vertex(Vertex v)
public String toString( )
public Double getPoint( )
public void setPoint(Double p)
public void setPoint(double x, double y)
public String getName( )
public void setName(String name)
vertexes
0..*
root
root
Tree
Attributes
Operations
public Tree(String name)
public Tree(Tree tree)
public Tree(Tree tree, boolean copyArcs)
public Tree(Graph graph)
public Tree(Graph graph, boolean copyArcs)
public void setRoot(String name)
public void setRoot(Vertex v)
Utils
Attributes
Operations
public Vertex[0..*] createVertexList( )
public Arc[0..*] createArcList( )
public Arc[0..*,0..*] createArcListList( )

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 List<Vertex> vertexes = null;
   /** 弧の一覧 */
   protected List<Arc> arcs = null;
   /** 有向グラフであるか */
   protected boolean directed = true;
   /** 弧からその始点への写像 */
   protected HashMap<Arc, Vertex> a2vHead = null;
   /** 弧からその終点への写像 */
   protected HashMap<Arc, Vertex> a2vTail = null;
   /** 頂点を始点とする弧のリスト */
   protected HashMap<Vertex, List<Arc>> v2a = null;
   /** 隣接行列 */
   protected int adjacent[][] = null;
   /** 二つの頂点の間に道があるかの有無 */
   protected boolean connectedness[][] = null;
   private HashMap<Vertex, Vertex> map = null;
   private HashMap<Arc, Arc> amap = null;
```

∂^+

∂^-

δ^+

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 List<Vertex> getVertexes()　頂点一覧を取得
public List<Arc> getArcs()　弧一覧を取得
public List<Arc> getArcs(Vertex v)　指定した頂点を視点とする
　弧の一覧
public Vertex getHead(Arc a)　指定した弧の始点
public Vertex getTail(Arc a)　指定した弧の終点

Vertexクラス：フィールド

```
/** 頂点のラベル */
private String name;
/** 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 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

```
/** 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;
```

グラフの定義

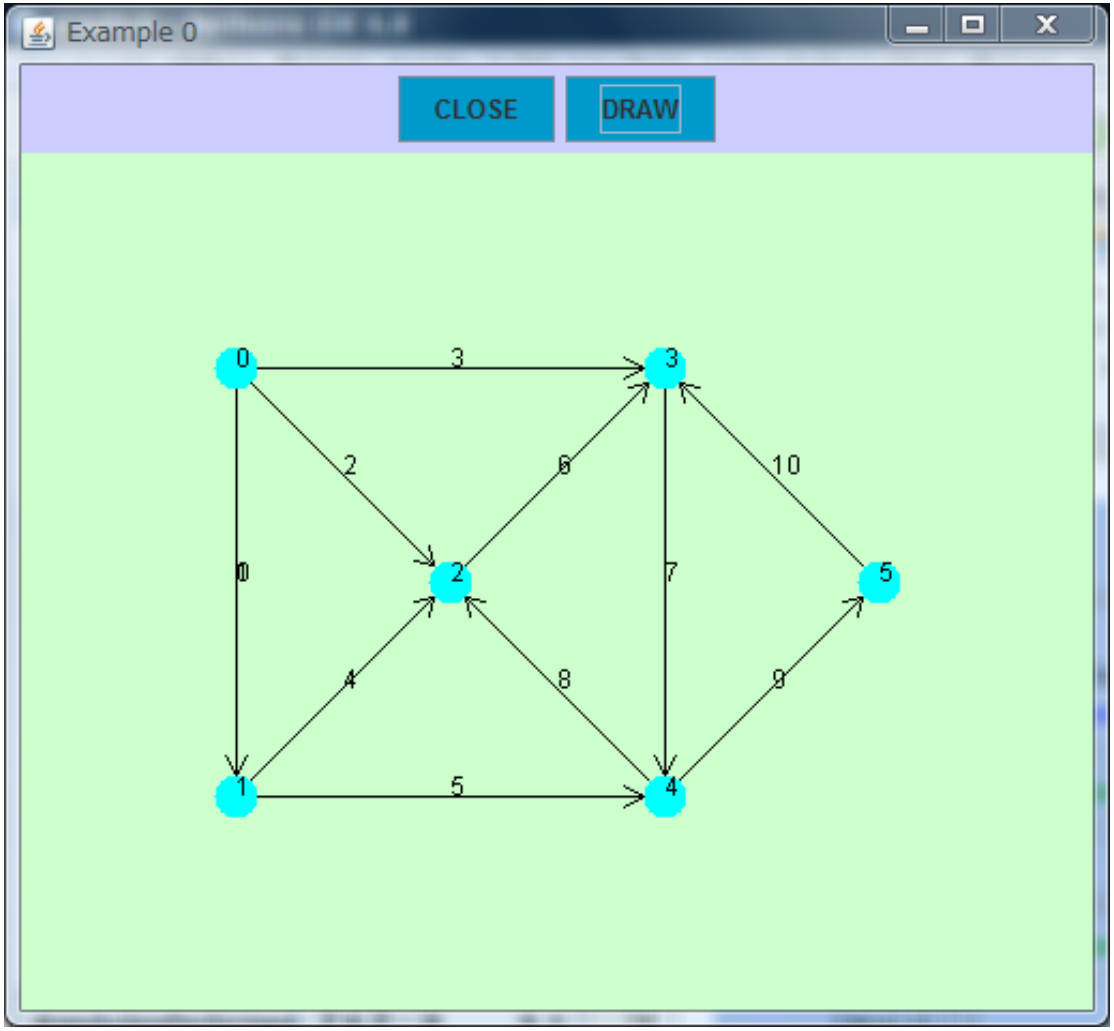
- graphLib.Graphを拡張

```
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 drawActionPerformed(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.mkImage();
        /** 作図イメージ生成 */
        setTitle(graphExample0.toString());
        /** タイトル設定 */
        repaint();
        /** 全体再描画 */
    }
```


Example 0
CLOSE
DRAW
0
1
2
3
4
5
3
2
6
10
0
7
4
8
9
5

Graph0.java

```
package example0;

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));        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);
```

```
        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 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
```

```
    /**
     * 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
}
```