Project Progression Report

242-402 COMPUTER ENG PROJECT II

Semester 2/2561

Project Name :

Executing Pictures as Programs

by

Wattanai Sirirak 5810110298

Submitted date 4/17/2019

Advisor

……………………………………………

(Dr. Andrew Davison)

Department of Computer Engineering

Faculty of Engineering

Prince of Songkla University

Abstract

Executing Pictures as Programs is a project for Java program to extract flowchart diagram or Nassi–Shneiderman diagrams via SVGsalamander engine to information. Both diagram can be drew as SVG (Scalable Vector Graphics) which is an XML-based vector image format for two-dimensional graphics with support for interactivity and animation. Use JGraphT to store and compute information into graph, generate this graph into picture with JGraphX then executable basic code.

Contents **Page**

Chapter 1. Introduction

Flowchart diagram and Nassi-Schneiderman diagram are nowadays used across many industries, serving different purposes and easy to understand. It’s harder to write a program. By this project, we don’t need to waste time to write codes. Just draw a picture as SVG format. This program will extract and convert SVG file to basic code then execute it.

1.1 Objectives

Write java code to extract SVG information from Flowchart diagram to get useful information such as id, x-y position, width, height, text, etc. Determines which kind of shape is it. JGraphT will compute the graph and JGraphX will generate graph picture. Then use JGraphT to convert that information to executable basic code.

1.2 Advantages

The program is good for users who are learning flowchart diagram and programming. It's easier to understand programming by pictures not just only codes.

1.3 Scope

The extractor program will be written by Java language and use SVGsalamander as a renderer. Use Inkscape and Shape Creator extension to create SVG picture as Flowchart diagram. The program will extract information, print JGraphT/JGraphX and use JGraph to convert information to basic code.

1.4 Procedure

1. Learn Java Language
2. Learn how to use Inkscape and Shape creator
3. Learn how to use SVGsalamander
4. Learn Flowchart diagram & Nassi–Shneiderman diagram
5. Write Extractor program to extract SVG file as information
6. Use JgraphT and JgraphX to generate graph and picture
7. Write Converter program to convert information to basic code

This figure shows how project works.

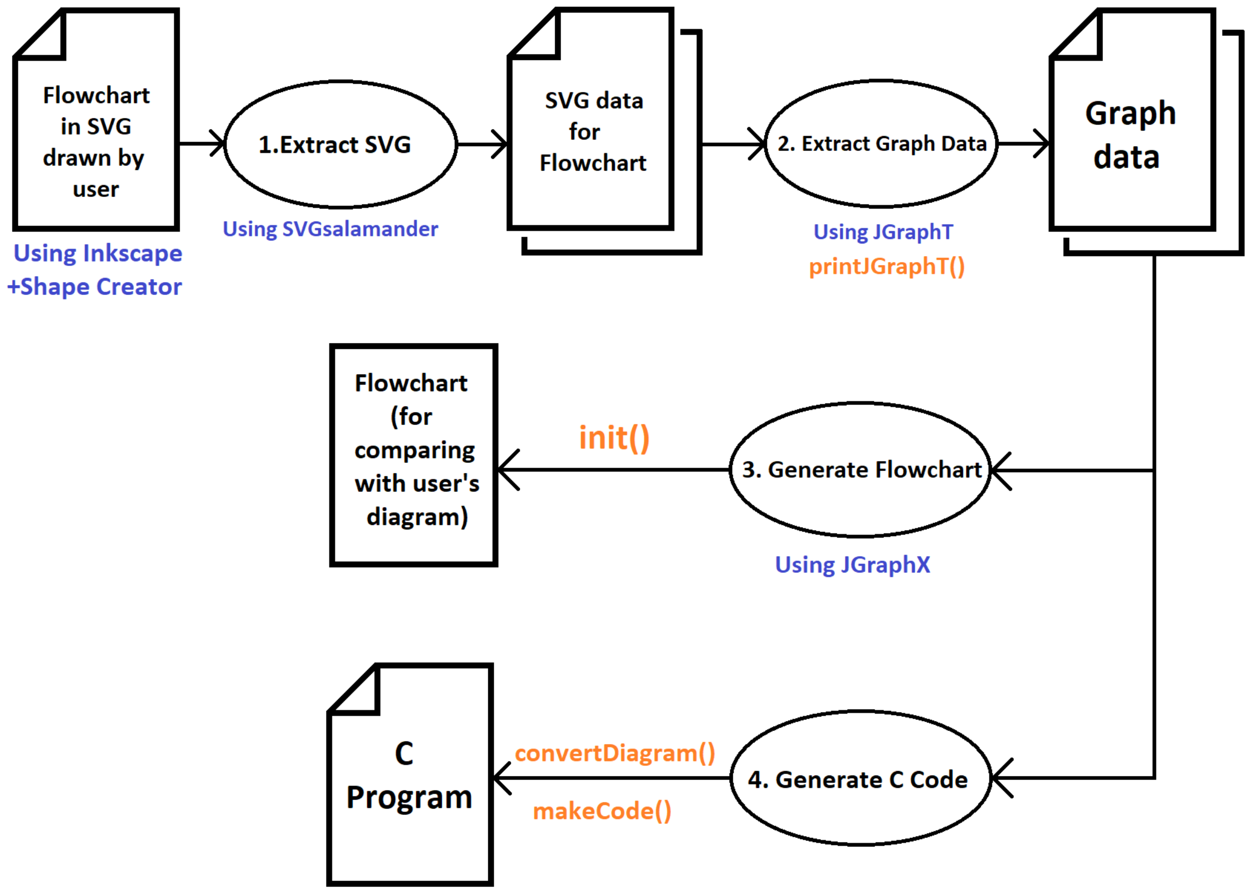


Figure 1. Project stages for Extractor.

Figure 1 shows how project works step by step.First step, User draw a flowchart diagram or a Nassi–Shneiderman diagram with Inkscape software then save it as SVG file. After this it will be the program's job. Extractor part will extract information from the SVG file then converter part will execute information that collected from extractor part. Make it as a graph in JGraphT and generate a picture from JGraphT by JGraphX then convert JGraphT into basic code.

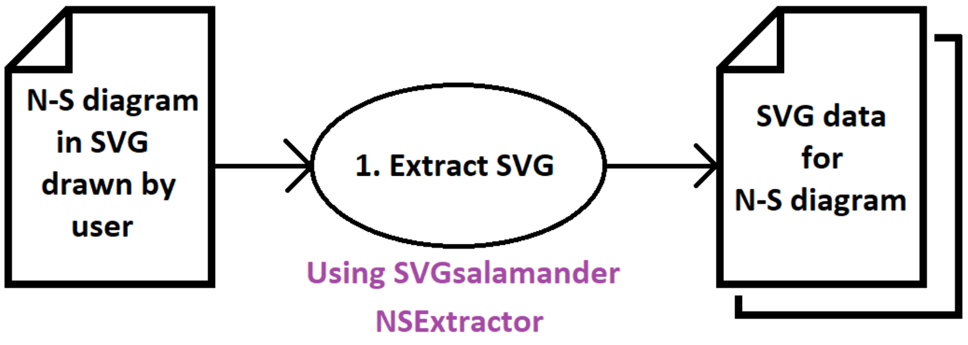
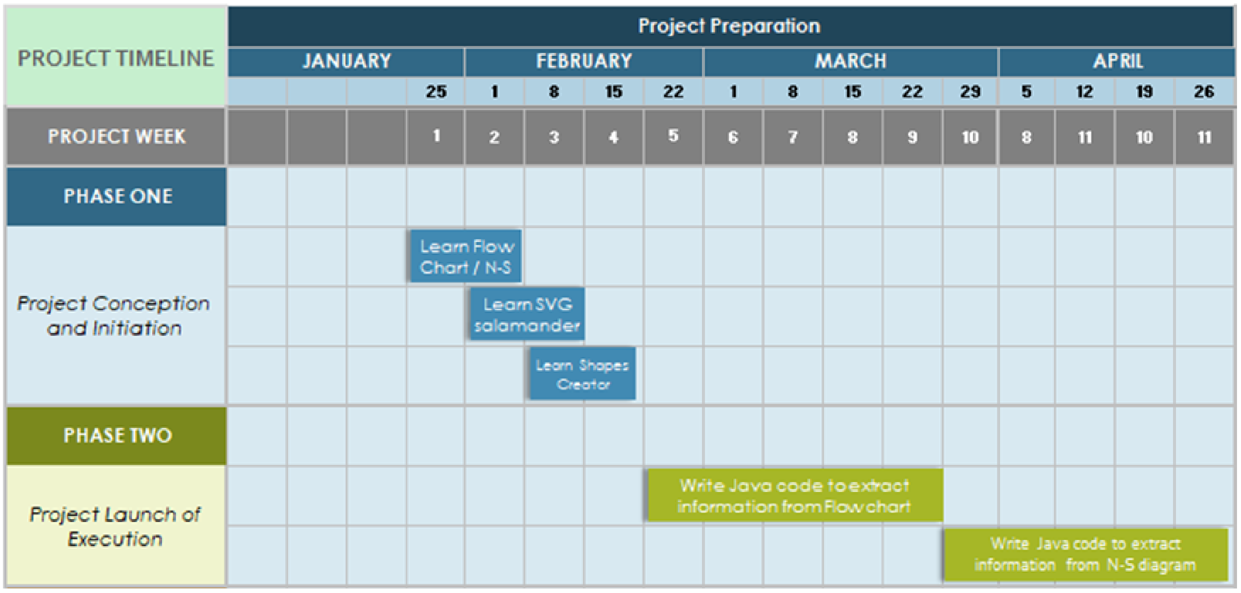


Figure 2. Project stages for NSExtractor.

1.5 Schedule

Project timeline for Project Preparation semester 2/2560

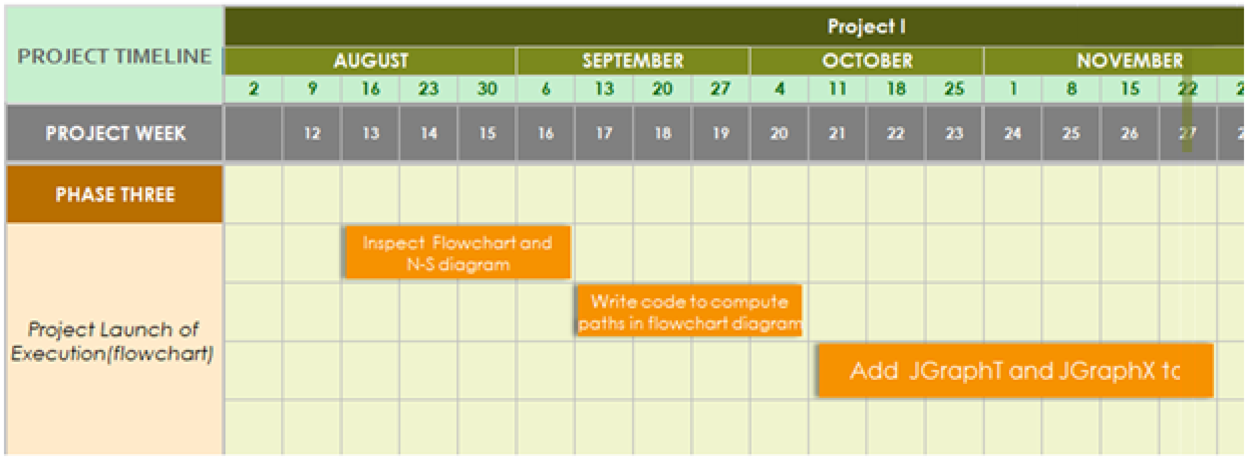


PROJECT TIMELINE	Project Preparation																
	JANUARY				FEBRUARY				MARCH					APRIL			
				25	1	8	15	22	1	8	15	22	29	5	12	19	26
PROJECT WEEK				1	2	3	4	5	6	7	8	9	10	8	11	10	11
PHASE ONE																	
Project Conception and Initiation				Learn Flow Chart / N-S													
					Learn SVG salamander												
						Learn Shapes Creator											
PHASE TWO																	
Project Launch of Execution								Write Java code to extract information from Flow chart									
													Write Java code to extract information from N-S diagram				

Figure 3. Project preparation timeline.

Figures 3 shows project schedule for Project Preparation at semester 2/2560. Start with learning Flowchart and N-S diagram, SVG salamander and Shape creator then start writing codes to extract Flowchart(Figure 1) and N-S diagram(Figure 2).

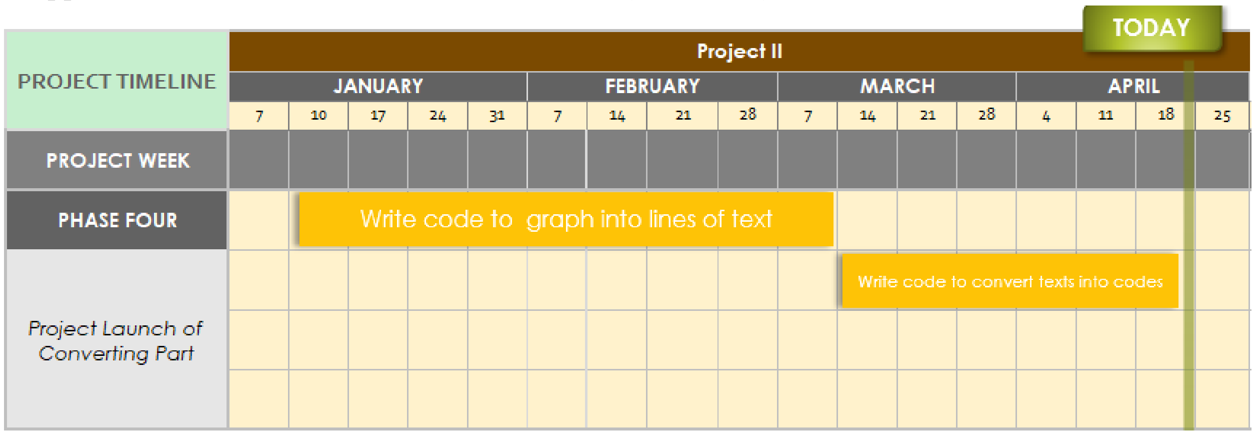
Project timeline for Computer Eng Project I semester 1/2561



PROJECT TIMELINE	Project I																	
	AUGUST					SEPTEMBER				OCTOBER				NOVEMBER				
	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15	22	2
PROJECT WEEK		12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	2
PHASE THREE																		
Project Launch of Execution(flowchart)			Inspect Flowchart and N-S diagram															
							Write code to compute paths in flowchart diagram											
											Add JGraphT and JGraphX tc							

Figure 4. Project I timeline.

Figures 4 shows project schedule for Project I at semester 1/2561. Start with inspecting Flowchart and N-S diagram then writing codes to compute paths in Flowchart diagram and merge program with JGraphT and JGraphX(Figure 1). NSExtractor(N-S diagram) has been dropped in this semester and focus on Extractor (Flowchart).



TODAY

PROJECT TIMELINE	Project II																
	JANUARY					FEBRUARY				MARCH				APRIL			
	7	10	17	24	31	7	14	21	28	7	14	21	28	4	11	18	25
PROJECT WEEK																	
PHASE FOUR		Write code to graph into lines of text															
Project Launch of Converting Part											Write code to convert texts into codes						

Figure 5. Project II timeline.

Figures 5 shows project schedule for Project II at semester 2/2561. All work is writing codes to convert graph from JGraphT to basic codes(Figure 1).

Chapter 2. Flowchart Diagrams

Flowchart diagram and Nassi-Shneiderman diagram are the most popular structures in nowadays. Therefore this project will use these two diagrams as pictures to execute as programs.

A flowchart is a type of diagram that represents an algorithm, workflow or process, showing the steps as boxes of various kinds, and their order by connecting them with arrows. This diagrammatic representation illustrates a solution model to a given problem. Flowcharts are used in analyzing, designing, documenting or managing a process or program in various fields[6][7]. I will use Edraw software to create examples.

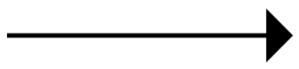
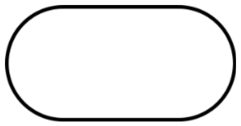
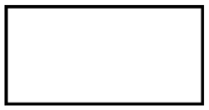
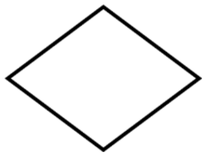
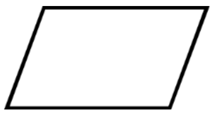
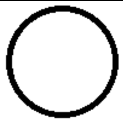
Shape	Name	description
	Flow line(Arrow head)	Shows the program's order of operation. A line coming from one symbol and ending at another.
	Terminal	Beginning or ending of a program or sub-process.
	Process	Set of operations that change value, form, or location of data.
	Conditional	Operation determining which of two paths the program will take.
	Input/Output	Input and output of data, as in entering data or displaying results.
	Circle	End point of if-else condition

Table 1. Flow chart common symbols.

Table 1 shows Flowchart symbols. Flowcharts use special shapes to represent different types of actions or steps in a process. Lines and arrows show the sequence of the steps, and the relationships among them. These are known as flowchart symbols[3].

Chapter 3. Nassi–Shneiderman Diagrams

A Nassi–Shneiderman diagram (NSD) in computer programming is a graphical design representation for structured programming. This type of diagram was developed in 1972 by Isaac Nassi and Ben Shneiderman who were both graduate students at SUNY-Stony Brook. These diagrams are also called structograms, as they show a program's structures[5][8]. I will use Edraw software to create examples.

There are 3 important Nassi-Shneiderman Shapes

3.1 Process blocks

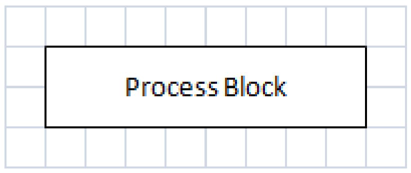


Figure 6. Process block.

Figure 6 shows process block. The process block represents the simplest of steps and requires no analyses. When a process block is encountered the action inside the block is performed and we move onto the next block.

3.2 Branching blocks

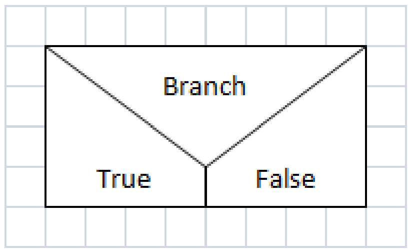


Figure 7. Branches block.

Figure 7 shows branches block. Branches block is the simple True/False or Yes/No branching block which offers the program two paths to take depending on whether or not a condition has been fulfilled. These blocks can be used as a looping procedure stopping the program from continuing until a condition has been fulfilled.

3.3 Testing Loops

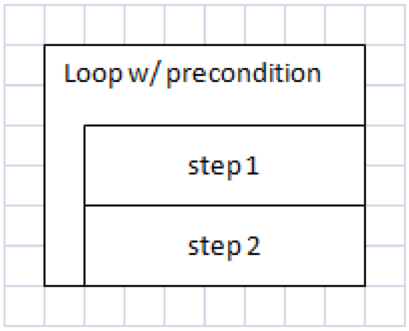


Figure 8. Test first loop block.

Figure 8 shows testing loop block. Testing Loops block allows the program to loop one or a set of processes until a particular condition is fulfilled. The process blocks covered by each loop are subset with a side-bar extending out from the condition.

There are two main types of testing loops, test first and test last blocks. The only difference between the two is the order in which the steps involved are completed. In the test first situation, when the program encounters the block it tests to see if the condition is fulfilled, then, if it is not completes the process blocks and then loops back. The test is performed again and, if the condition is still unfulfilled, it processes again. If at any stage the condition is fulfilled the program skips the process blocks and continues onto the next block.

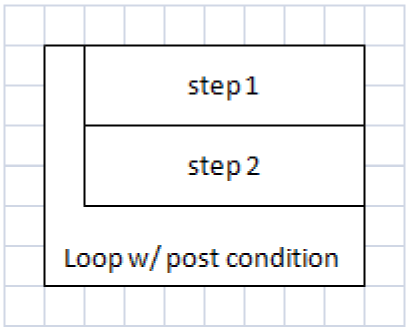


Figure 9. Test last loop block.

Figure 9 shows test last loop block. The test last block is simply reversed, the process blocks are completed before the test is performed. The test last loop allows for the process blocks to be performed at least once before the first test.

Chapter 4. SVGsalamander

SVG Salamander[2] is an SVG engine for Java that's designed to be small, fast, and allow programmers to use it with a minimum of fuss. It's in particular targeted for making it easy to integrate SVG into Java games and making it much easier for artists to design 2D game content - from rich interactive menus to charts and graphics to complex animations. Extractor and NSExtractor use this engine to extract file(Figure 1,2).

4.1 Features

Ant task to allow easy conversion from SVG to images from within Ant scripts. SVGIcon class greatly simplifies loading and drawing images to screen. A much smaller code foot print than Batik, and only one JAR file to include. Direct access to the scene graph tree. You can use Java commands to manipulate it directly. An index of all named shapes in the SVG graph is easily accessible. Picking shapes given (x, y) coordinates is possible, and can be used to implement graphical buttons selected by the mouse. Clip region sensitivity makes for fast rendering when only updating part of the image. This makes panning the camera quite efficient. Easy rendering to any Graphics2D or BufferedImage. Unlike Batik, the SVG Salamander engine does not own the graphics context, so you can pass it whatever graphics context you like. Internal and external links are implemented as URIs, which allows the engine to automatically import linked documents - even if they're stored on a remote server.SVG can be read from an InputStream, so we can create documents dynamicly from an in-program XSLT transformation.

4.2 Installing SVGsalamander

Download SVGsalamander from https://github.com/blackears/svgSalamander and extract zip file and find SVGsalamander which can be found in www/binaries then use extension class SVGsalamander via java Extension mechanism by using –cp or –classpath in command line while running or compiling the program.

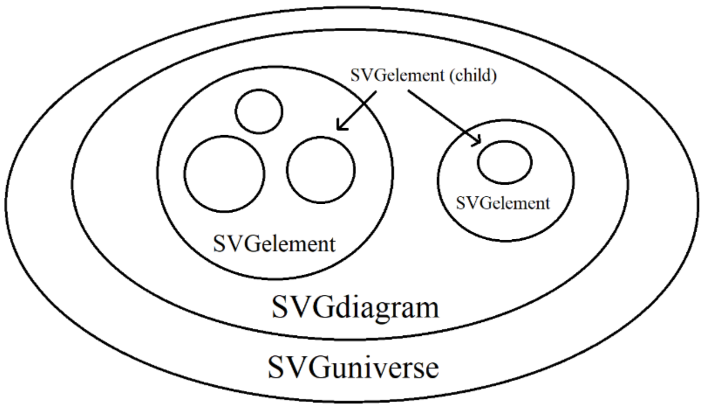


Figure 10. SVG data constructs.

Figure 10 shows SVG data construct inside SVG file. SVGdiagram could have many SVGelement inside and SVGelement could have many SVGelement inside too.

Example code to import SVG file (from SVGTest.java)

```
try {
  File f = new File(args[0]);
  uri = f.toURI();
} catch (Exception e) {
  System.out.println(e);
}
```

Example code to create new SVGuniverse (from Extractor.java)

```
SVGUniverse svgUniverse = new SVGUniverse();
```

Example code to get SVGdiagram (from Extractor.java)

```
SVGDiagram diagram = svgUniverse.getDiagram(uri);
```

Example code to get SVGelement (from Extractor.java)

```
SVGElement elem  = diagram.getRoot().getChild(i);
```

Chapter 5. SVGTest

SVGTest.java is a program to extract SVG file by using SVGsalamander as a renderer(Figure 1, stage 1). The output is including ID, position, width, height, text, transform, color and etc. This program has been written by Dr.Andrew Davision.

This is an example of SVGTest.java when execute this picture which drew by Inkscape.

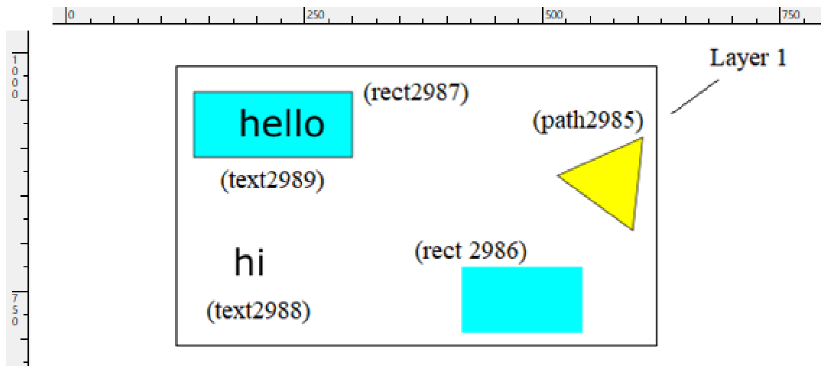
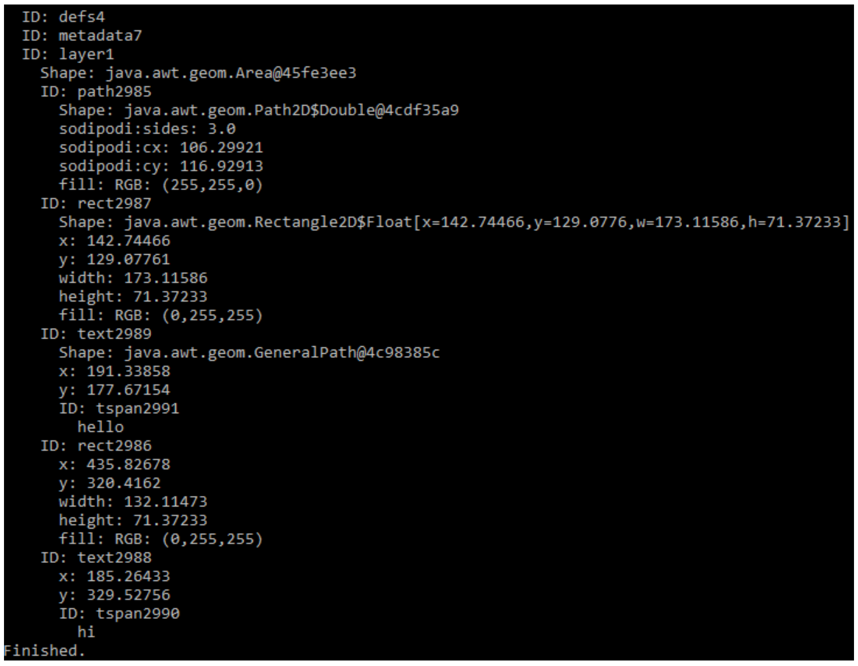


Figure 11. An example Inkscape picture.

Figure 11 shows the example included with text, rectangle and triangle.



```
  ID: defs4
  ID: metadata7
  ID: layer1
    Shape: java.awt.geom.Area@45fe3ee3
    ID: path2985
      Shape: java.awt.geom.Path2D$Double@4cdf35a9
      sodipodi:sides: 3.0
      sodipodi:cx: 106.29921
      sodipodi:cy: 116.92913
      fill: RGB: (255,255,0)
    ID: rect2987
      Shape: java.awt.geom.Rectangle2D$Float[x=142.74466,y=129.0776,w=173.11586,h=71.37233]
      x: 142.74466
      y: 129.07761
      width: 173.11586
      height: 71.37233
      fill: RGB: (0,255,255)
    ID: text2989
      Shape: java.awt.geom.GeneralPath@4c98385c
      x: 191.33858
      y: 177.67154
      ID: tspan2991
        hello
    ID: rect2986
      x: 435.82678
      y: 320.4162
      width: 132.11473
      height: 71.37233
      fill: RGB: (0,255,255)
    ID: text2988
      x: 185.26433
      y: 329.52756
      ID: tspan2990
        hi
Finished.
```

Figure 12. Output of SVGTest.java.

Figure 12 shows ID of each object, x-y position, transform, text and color as RGB.

5.1 How program extract the diagram

There are many important parts in SVGTest.java as following details.

Function main is a function to create new SVGuniverse, import a file to SVGdiagram then call function printAll with the main element of svg file.

```
SVGUniverse svgUniverse = new SVGUniverse();
SVGDiagram diagram = svgUniverse.getDiagram(file.toURI());
for(int i = 0; i < diagram.getRoot().getNumChildren(); i++) {
  SVGElement element  = diagram.getRoot().getChild(i);
  printAll(element, "  ");
}
```

Function printAll is a function to check if the element which received has child or not and print received element information by calling function printDetails. If the element has another child function printAll will call itself to do a loop again.

```
if (elem instanceof ShapeElement)
  printDetails(elem, tab+ "  ");
for(int i = 0; i < elem.getNumChildren(); i++) {
  SVGElement child  = elem.getChild(i);
  printAll(child, tab+ "  ");
}
```

After received an element from function printAll. Fucntion printDetails calls function getAtt and showAtt to print

```
ShapeElement se = (ShapeElement) elem;
if (se.getShape() != null)
  System.out.println(tab + "Shape: " + se.getShape());
if (se instanceof Tspan) {    // text span
  Tspan node = (Tspan)se;
  System.out.println(tab + node.getText());
}
else if (se instanceof Rect) {    // rectangle
  showAtt(elem, "x", tab);
  showAtt(elem, "y", tab);
  showAtt(elem, "width", tab);
  showAtt(elem, "height", tab);
  showStyle(elem, "fill", tab);
  //Rect node = (Rect)se;
}
```

Function showAtt is a function to print attribute of object.

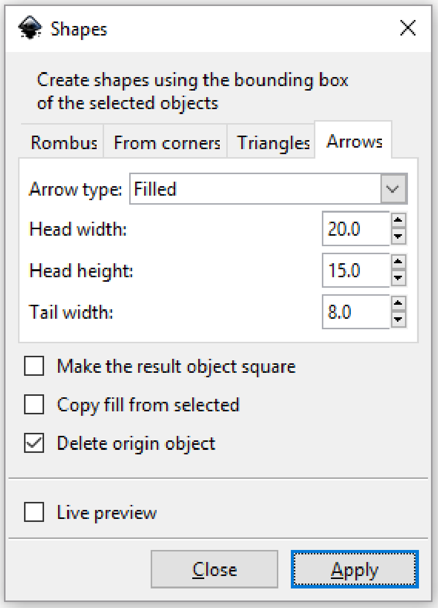
```
StyleAttribute attrib = new StyleAttribute(attNm);
elem.getStyle(attrib);
double value = attrib.getDoubleValue();
if (value != 0)
  System.out.println(tab + attNm + ": " + value);
```

Function getAtt is a function to return attribute of object.

```
StyleAttribute attrib = new StyleAttribute(attNm);
elem.getStyle(attrib);
return attrib.getStringValue();
```

Chapter 6. Shapes Creator

Shapes creator is an extension for Inkscape written in Python that creates shapes from the bounding boxes of selected objects. This extension is useful for drawing shapes (Figure 1, stage1) because with Inkscape it is hard to draw specific shapes so this extension will generate those shapes in fixed situation.



Shapes

Create shapes using the bounding box of the selected objects

Arrow type: Filled

Head width: 20.0

Head height: 15.0

Tail width: 8.0

☐ Make the result object square

☐ Copy fill from selected

☑ Delete origin object

☐ Live preview

Figure 13. Shapes Creator UI.

Figure 13 shows Shape creator UI. There are 4 shapes named as Rombus, From corners, Triangles, Arrow. Each shape has many types of shape which can select by drop down menu.

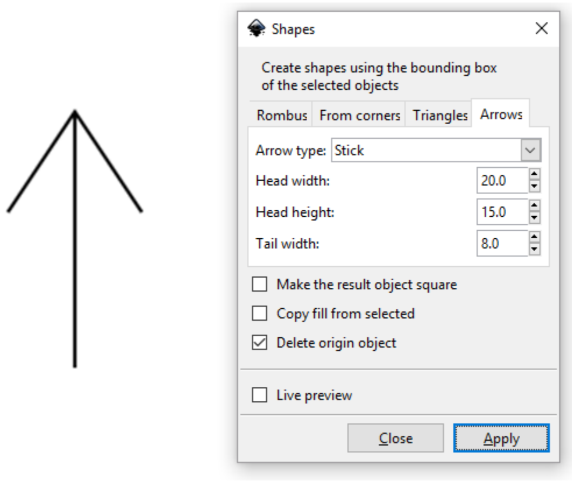
The available shapes are Rhombus, Shapes based from the bounding box corners Chamfer, Chamfer inverse, Rect inside, Round inside, Round inside inverse, Cross, Star from center, Star from corners. Triangles are Isosceles, Equilateral. Rectangle are bottom left, Rectangle bottom right, Rectangle top left, Rectangle top right. Arrows are Filled, Stick

Download Shapes creator from http://www.arakne.es/en/dessign/inkscape-python-extension-shapes/ and unpack the archive file. Copy the files into the directory listed at Edit > Preferences > System: User extensions. Restart Inkscape and the new extension will be available.

6.1 Shapes can be drawn with Shapes Creator

This section shows how to draw shapes by Shapes Creator and Inkscape directly for both Flowchart diagram and Nassi–Shneiderman diagram.

It's hard to use only Inkscape tools to draw Flowchart diagram. We can draw these objects easier by shapes creator and SVG file code will be fixed and easier to write execution code.


Shapes

Create shapes using the bounding box
of the selected objects

Rombus | From corners | Triangles | Arrows

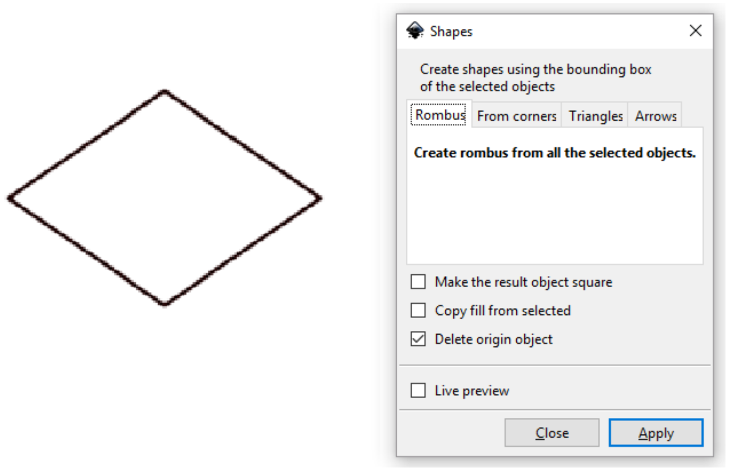
Arrow type: Stick
Head width: 20.0
Head height: 15.0
Tail width: 8.0

☐ Make the result object square
☐ Copy fill from selected
☑ Delete origin object

☐ Live preview

Figure 14. Stick arrow and Shapes creator setting.

Figure 14 shows stick arrow. Stick arrow can be used as Flowchart's Flow line symbol.


Shapes

Create shapes using the bounding box
of the selected objects

Rombus | From corners | Triangles | Arrows

Create rombus from all the selected objects.

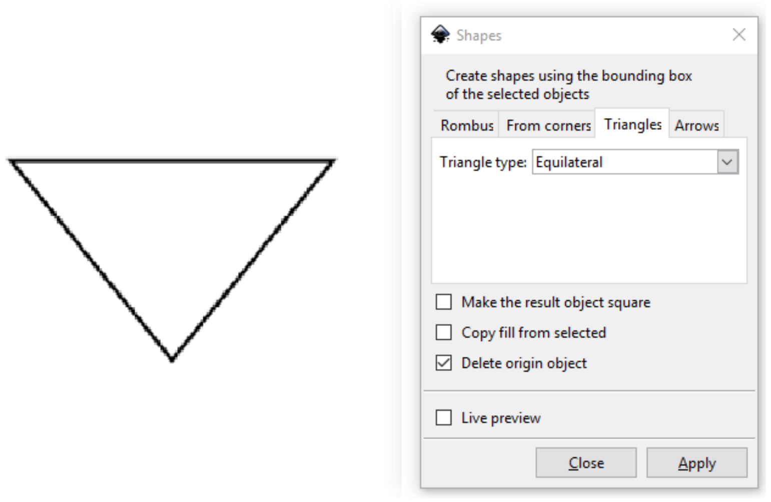
☐ Make the result object square
☐ Copy fill from selected
☑ Delete origin object

☐ Live preview

Figure 15. Rhombus and Shapes creator setting

Figure 15 shows stick Rhombus. Rhombus can be used as Flowchart's Conditional symbol.

It's hard to use only Inkscape tools to draw Nassi–Shneiderman diagram. We can draw these objects easier by shapes creator and SVG file code will be fixed and easier to write execution code.


Shapes

Create shapes using the bounding box of the selected objects

Rombus | From corners | Triangles | Arrows

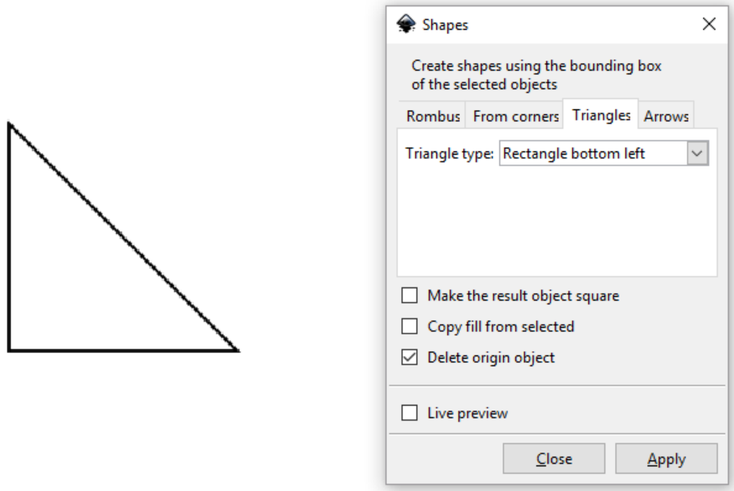
Triangle type: Equilateral

- [ ] Make the result object square
- [ ] Copy fill from selected
- [x] Delete origin object

- [ ] Live preview

Figure 16. Equilateral and Shapes creator setting

Figure 16 explains how to draw Equilateral. Equilateral can be used as a part of Nass-Shneiderman’s Branching blocks.


Shapes

Create shapes using the bounding box of the selected objects

Rombus | From corners | Triangles | Arrows

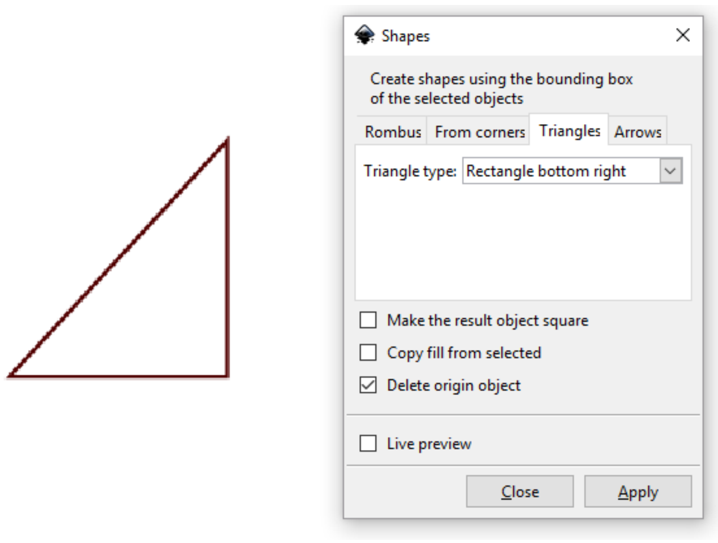
Triangle type: Rectangle bottom left

- [ ] Make the result object square
- [ ] Copy fill from selected
- [x] Delete origin object

- [ ] Live preview

Figure 17. Rectangle bottom left and Shapes creator setting

Figure 17 explains how to draw Rectangle bottom left. Rectangle bottom left can be used as a part of Nassi–Shneiderman’s Branching blocks**.**


Shapes

Create shapes using the bounding box of the selected objects

Rombus | From corners | Triangles | Arrows

Triangle type: Rectangle bottom right

- [ ] Make the result object square
- [ ] Copy fill from selected
- [x] Delete origin object

- [ ] Live preview

Figure 18. Rectangle bottom right and Shapes creator setting

Figure 18 explains how to draw Rectangle bottom right. Rectangle bottom right can be used as a part of Nassi–Shneiderman’s Branching blocks**.**

6.2 Shapes can be drawn with Inkscape directly

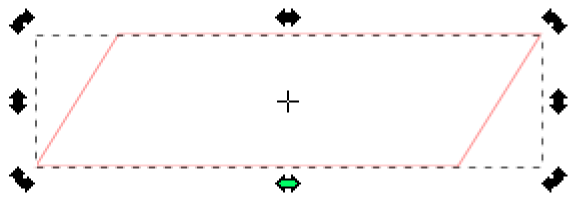


Figure 19. How to draw parallelogram.

Figure 19 explains how to draw parallelogram. Parallelogram shape can be draw with Inkscape directly by creating normal rectangle and click on the rectangle twice then hold click on bottom-middle arrow to draw parallelogram.

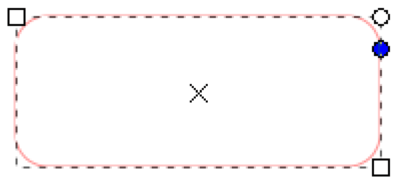


Figure 20. How to draw oval.

Figure 20 explains how to draw oval. Oval shape can be draw with Inkscape directly by creating normal rectangle and click on the rectangle twice then hold click on right corner bubble to draw Oval.

Chapter 7. NSExtractor

NSExtractor.java (Appendix D) is a program to extract Nassi–Shneiderman SVG file (Figure 2). Most of codes are similar to Extractor.java but there are only 3 subclasses Triangle, Rectangle and Text for now (Appendix E) .

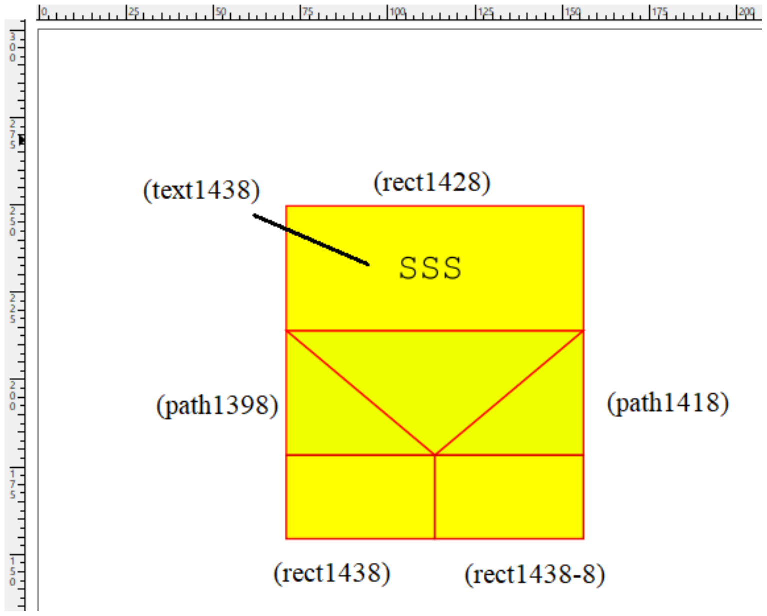
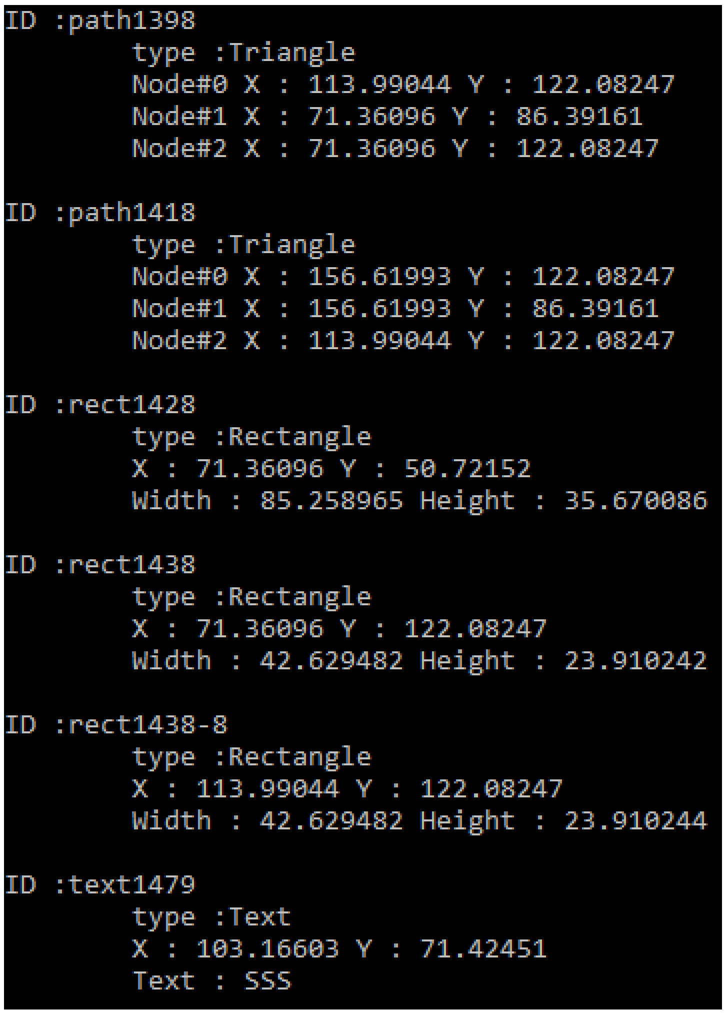


Figure 21. An example Inkscape N-S picture.

Figure 21 shows an example of some N-S shapes drew by Inkscape and Shapes creator extension.



```
ID :path1398
        type :Triangle
        Node#0 X : 113.99044 Y : 122.08247
        Node#1 X : 71.36096 Y : 86.39161
        Node#2 X : 71.36096 Y : 122.08247

ID :path1418
        type :Triangle
        Node#0 X : 156.61993 Y : 122.08247
        Node#1 X : 156.61993 Y : 86.39161
        Node#2 X : 113.99044 Y : 122.08247

ID :rect1428
        type :Rectangle
        X : 71.36096 Y : 50.72152
        Width : 85.258965 Height : 35.670086

ID :rect1438
        type :Rectangle
        X : 71.36096 Y : 122.08247
        Width : 42.629482 Height : 23.910242

ID :rect1438-8
        type :Rectangle
        X : 113.99044 Y : 122.08247
        Width : 42.629482 Height : 23.910244

ID :text1479
        type :Text
        X : 103.16603 Y : 71.42451
        Text : SSS
```

Figure 22. Output of NSExtractor.java.

Figure 22 shows the output of NSExtractor.java. Showed information which extracted and stored in subclasses Triangle, Rectangle and Text. There are 3 nodes in Triangle. X and Y positions, height and width in Rectangle. X and Y positions and text in Text.

7.1 How program extract the diagram

Most of NSExtractor codes are similar to Extractor.java but there are function and classes differnce because each diagram uses difference symbols.

Triangle class contains all triangles symbols.

```
public static class Triangle extends SVGNode
{
  float[] iAngleX = new float[4];
  float[] iAngleY = new float[4];;
  public Triangle(String iId,float iAngleX[],float iAngleY[])
  {
    super(iId);
    this.iAngleX = iAngleX.clone();
    this.iAngleY = iAngleY.clone();
  }

  public void printAll()
  {
    System.out.println("\nID :"+iId);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    for(int i=0;i<3;i++){
      System.out.println("\tNode#"+i+" X : "+iAngleX[i]+" Y : "+iAngleY[i]);
    }
  }
}
```

Function storeNode is difference because it needs to determine triangle.

```
public static void storeNode(SVGElement elem,int i,int objCount)
{
  float[] tmpx = new float[3];
  float[] tmpy = new float[3];
  String tText = "";
  String sType = "";
  String pres[]= {"x","y","width","height","cy","cx", "rx", "ry"};
  SVGElement element = elem.getChild(objCount);
  try{  if(element.hasAttribute("d",AnimationElement.AT_AUTO)){
   String[] info = element.getPresAbsolute("d").getStringValue().split(" ");
      for(int z=0; z<info.length;z++){
        if(info[z].contains("V")||info[z].contains("v")){
          tmpx[0] = tmpx[1];
          tmpy[0] = Float.parseFloat(info[z+1]);
          sType = "triangle";
          z++;
        }
      }
    }
  }
```

Chapter 8. JGraphT

JGraphT is a free Java graph library that provides mathematical graph-theory objects and algorithms. JGraphT supports various types of graphs including directed graph which is needed for flowchart diagram[1]. JgraphT is an important part to extract graph data(Figure 1, stage 2).

This is example directed graph and code.

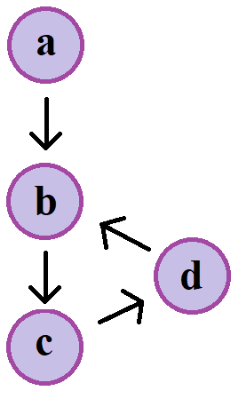
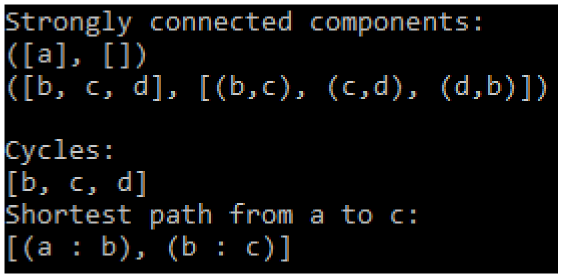


Figure 23. Directed graph example.

Figure 23 shows simple directed graph with a loop inside (b->c->d).

```
public class DirectedGraphDemo
{
  public static void main(String args[])
  {
    // constructs a directed graph with the specified vertices and edges
    DefaultDirectedGraph<String, DefaultEdge> directedGraph =
      new DefaultDirectedGraph<String, DefaultEdge>(DefaultEdge.class);
    directedGraph.addVertex("a");
    directedGraph.addVertex("b");
    directedGraph.addVertex("c");
    directedGraph.addVertex("d");
    directedGraph.addEdge("a", "b");
    directedGraph.addEdge("b", "c");
    directedGraph.addEdge("c", "d");
    directedGraph.addEdge("d", "b");

    // Computes all the strongly connected components of the directed graph
    StrongConnectivityAlgorithm<String, DefaultEdge> scAlg =
      new KosarajuStrongConnectivityInspector<>(directedGraph);
    List<Graph<String, DefaultEdge>> stronglyConnectedSubgraphs =
      scAlg.getStronglyConnectedComponents();

    // Prints the strongly connected components
    System.out.println("Strongly connected components:");
    for (int i = 0; i < stronglyConnectedSubgraphs.size(); i++) {
      System.out.println(stronglyConnectedSubgraphs.get(i));
    }

    // Gets all cycles from directGraph with TarjanSimpleCycles
    TarjanSimpleCycles tsCycles =
      new TarjanSimpleCycles<String, DefaultEdge> (directedGraph);
    // Prints all cycles
    for (int i = 0; i < tsCycles.findSimpleCycles().size(); i++) {
      System.out.println("\nCycles:");
      List<String> s = (List<String>)tsCycles.findSimpleCycles().get(i);
        System.out.println(s);
    }
    // Prints the shortest path from vertex a to vertex c
    System.out.println("Shortest path from a to c:");
    DijkstraShortestPath<String, DefaultEdge> dijkstraAlg =
      new DijkstraShortestPath<>(directedGraph);
    SingleSourcePaths<String,DefaultEdge> iPaths = dijkstraAlg.getPaths("a");
    System.out.println(iPaths.getPath("c") + "\n");
  }
}
```



```
Strongly connected components:
([a], [])
([b, c, d], [(b,c), (c,d), (d,b)])

Cycles:
[b, c, d]
Shortest path from a to c:
[(a : b), (b : c)]
```

Figure 24. JGraphT output of the example.

Figure 24 shows the output of JGraphT of directed graph example.

Chapter 9. JGraphX

JGraphX enables to produce Java Swing applications that feature interactive diagramming functionality. The core client functionality of JGraphX is a Java 5 compliable library that describes, displays and interacts with diagrams as part of larger Java Swing application. JGraphX is primarily designed for use in a desktop environment[4]. JgraphT is an important part to generate Flowchart picture(Figure 1, stage 3).

```
public class HelloWorld extends JFrame
{
  private static final long serialVersionUID = -2707712944901661771L;
  public HelloWorld()
  {
    super("Hello, World!");
    //Builds mxGraph class and gets the first child of the root in the model
    mxGraph graph = new mxGraph();
    Object parent = graph.getDefaultParent();
    graph.getModel().beginUpdate();
    try
    {
      //Adds vertexes to the graphs
      Object v1 = graph.insertVertex(parent, null, "Hello", 20, 20, 80,30);
      Object v2 = graph.insertVertex(parent, null, "World!",240, 150,80, 30);
      graph.insertEdge(parent, null, "Edge", v1, v2);
    }
    finally
    {
      graph.getModel().endUpdate();
    }
    //Gets components from mxGraph
    mxGraphComponent graphComponent = new mxGraphComponent(graph);
    getContentPane().add(graphComponent);
  }

  public static void main(String[] args)
  {
    //Generates JFrame
    HelloWorld frame = new HelloWorld();
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 320);
    frame.setVisible(true);
  }
}
```

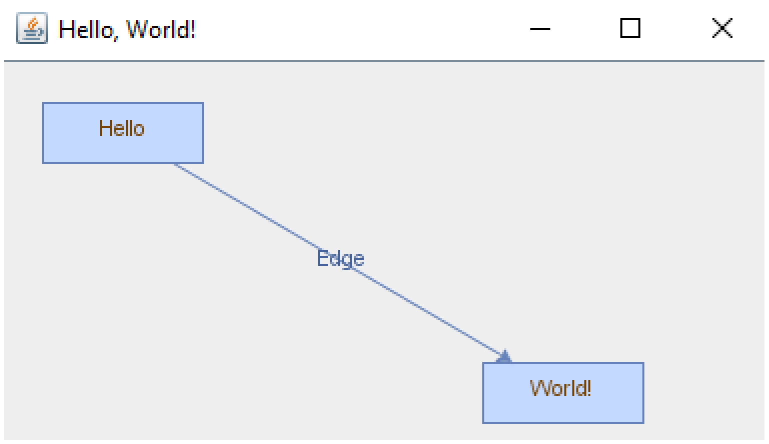


Figure 25. JGraphX output of the example.

Figure 25 shows JGraphX output of the example. Showing an edge connected from Hello node to World! Node.

Chapter 10. Program examples

This chapter shows examples with multiple tests by drawing svg file from original code then use Extractor program to extract svg file into code and generate picture with JGraphX.

10.1 printAllNumbers.c

Gets a number from user and print all number from 1 to selected number.
Test : one while loop.

10.2 printEvenNumbers.c

Gets a number from user and print all even number from selected number to 100.
Test : nested if in while loop.

10.3 ageCheck.c

Gets age from user and check if they are eligible to work or not.
Test : nested if.

10.4 printBackBelow100.c

Gets a number from user and print all number backward and it must below 100.
Test : nested while loop in if

10.5 printTriangleStar.c

Gets a number from user and print pattern triangle star(*).
Test : two while loop.

10.6 checkArmStrongNumber.c

Gets 3 digits number from user and check if it is an armstrong number or not.
Test : while and if.

10.7 findNPrimeNumbers.c

gets a number from user and print prime numbers for n numbers.
Test : if nested in nested while loop.

10.1 printAllNumbers.c

Gets a number from user and print all number from 1 to selected number.

Test : one while loop.

Draw a SVG file in Inkscape by user, Extractor will convert into SVG data(Figure 1, stage 1).

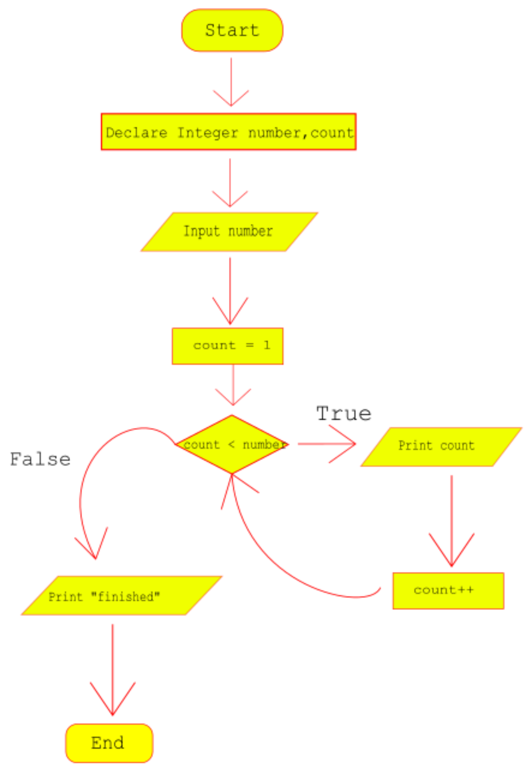


Figure 26. Example program drew in Inkscape

Example fragment code from SVG file.

```
<rect
   style="opacity:1;fill:#edff00;fill-opacity:1;stroke:#ff0000;stroke-
    width:0.34027454;stroke-miterlimit:4;
    stroke-dasharray:none;stroke-opacity:1"
   id="rect2111"
   width="44.918083"
   height="18.711079"
   x="56.376438"
   y="5.7896147"
   ry="8.0651178" />
```

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

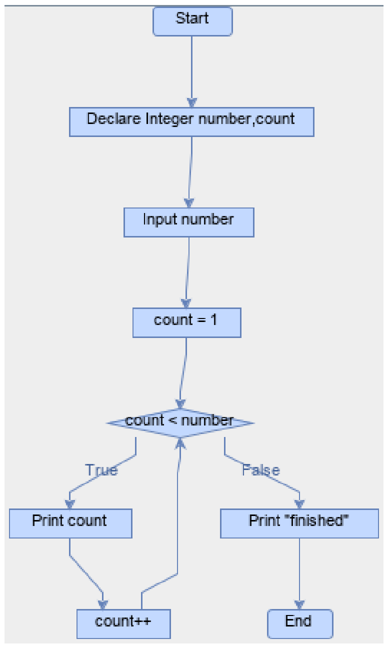


Figure 27. Picture generated from JGraphX

Extractor generates C code from graph data(Figure 1, stage 4).

Generated code from Extractor is fully correct.

```
main(){
int number,count;
scanf("%d",&number);
count = 1;
while(count < number){
  printf("%d",count);
  count++;
}
printf("finished");
}
```

10.2 printEvenNumbers.c

Gets a number from user and print all even number from selected number to 100.

Test : nested if in while loop.

Draw a SVG file in Inkscape by user, Extractor will convert into SVG data(Figure 1, stage 1).

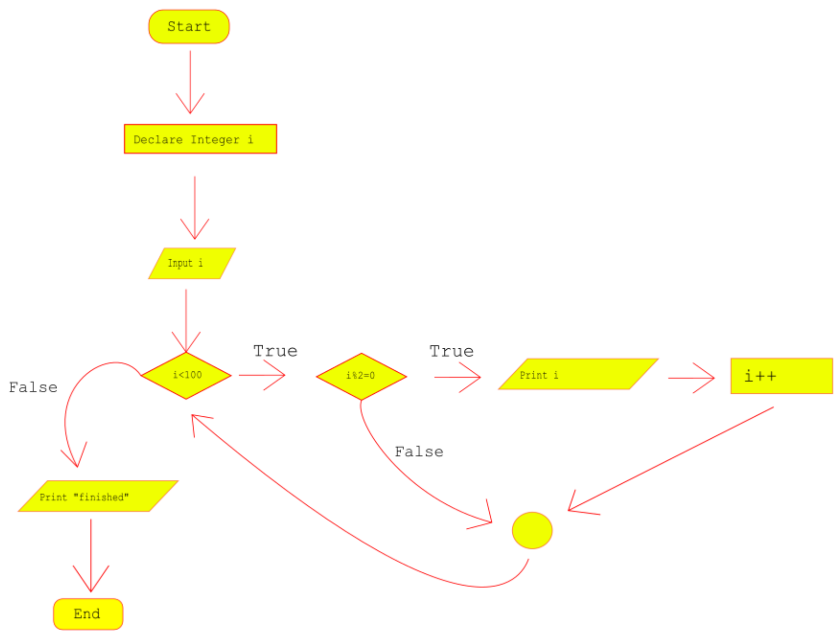


Figure 28. Example program drew in Inkscape

Example fragment code from SVG file.

```
<path
   inkscape:connector-curvature="0"
   d="M 80.986696,53.422772 V 18.349586 m -7.904128,23.802897
    7.904128,11.270289 7.90413,-11.270289"
   style="fill:none;stroke:#ff0000;stroke-width:0.38531762"
   id="path1141" />
```

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

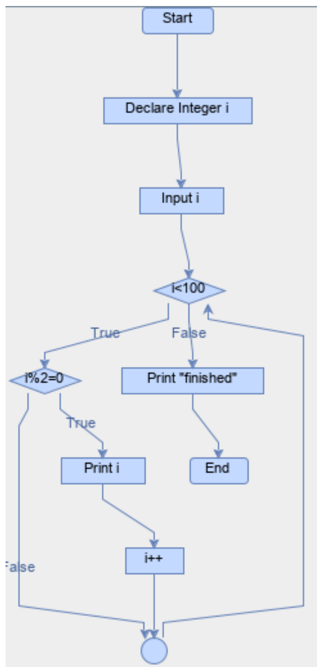


Figure 29. Picture generated from JGraphX

Extractor generates C code from graph data(Figure 1, stage 4).

Generated code from Extractor is fully correct.

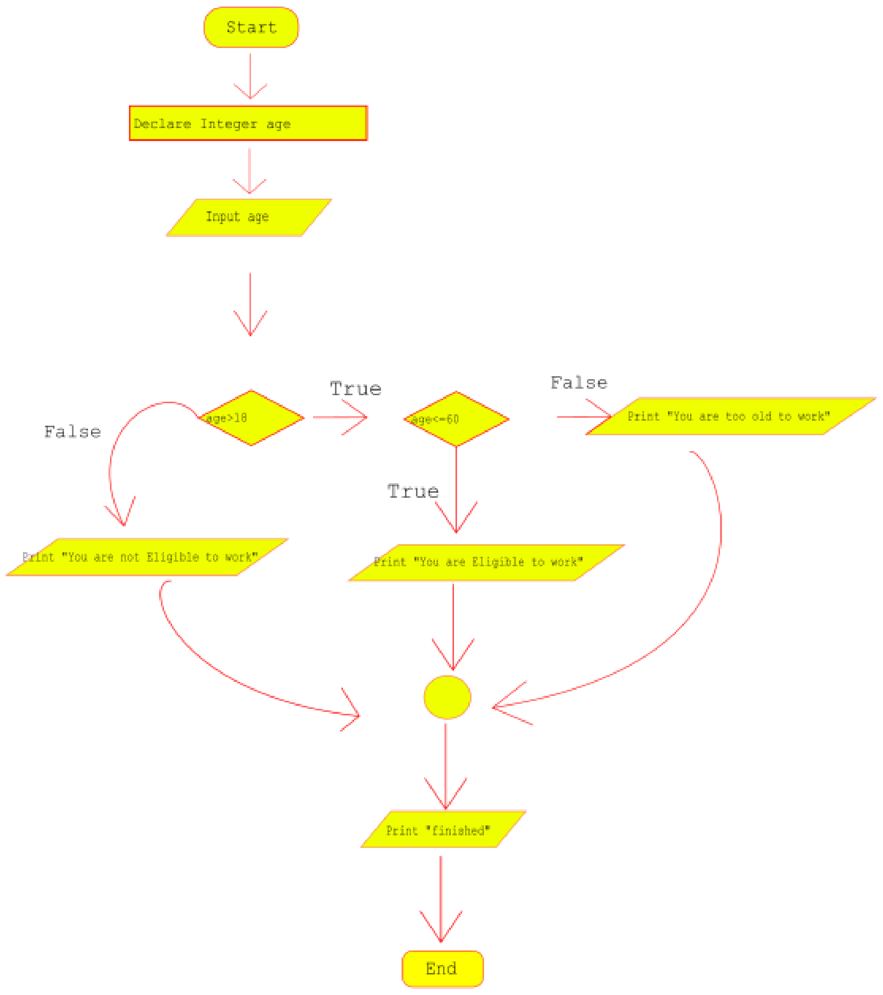
```
main(){
int i;
scanf("%d",&i);
while(i<100){
  if(i%2=0){
    printf("%d",i);
    i++;
  }
  else{

  }
}
printf("finished");
}
```

10.3 ageCheck.c

Gets age from user and check if they are eligible to work or not.

Test : nested if.

Draw a SVG file in Inkscape by user, Extractor will convert into SVG data(Figure 1, stage 1).



Start
Declare Integer age
Input age
age>18
True
age<=60
False
False
True
Print "You are too old to work"
Print "You are not Eligible to work"
Print "You are Eligible to work"
Print "finished"
End

Figure 30. Example program drew in Inkscape

Example fragment code from SVG file.

```
<path
   id="path4679-1"
   style="fill:none;stroke:#ff0000;stroke-width:0.47846198;stroke-
    miterlimit:4;stroke-dasharray:none"
   d="M 130.17735,340.50078 C 40.001098,330.6325 25.339006,275.96358
    40.531794,276.62403 m 80.616496,71.0405 9.02906,-7.16375 -8.46077,-13.43"
   inkscape:connector-curvature="0"
   sodipodi:nodetypes="ccccc" /
```

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

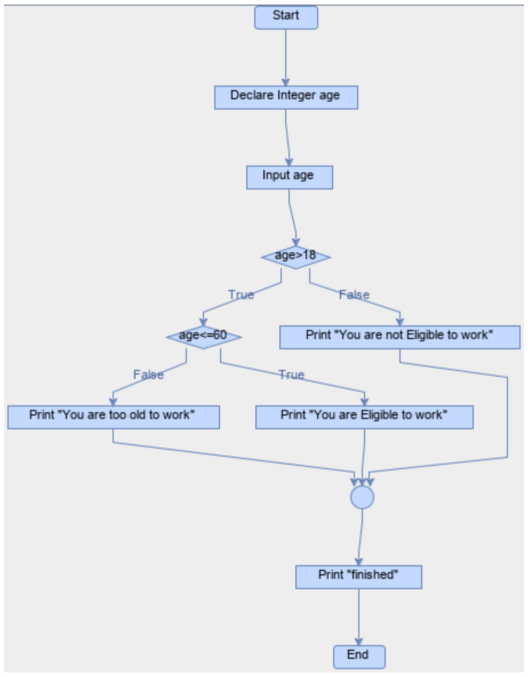

Start
Declare Integer age
Input age
age>18
True
False
age<=60
Print "You are not Eligible to work"
False
True
Print "You are too old to work"
Print "You are Eligible to work"
Print "finished"
End

Figure 31. Picture generated from JGraphX

Extractor generates C code from graph data(Figure 1, stage 4).

Generated code from Extractor is fully correct.

```
main(){
int age;
scanf("%d",&age);
if(age>18){
  if(age<=60){
    printf("You are Eligible to work");
  }
  else{
    printf("You are too old to work");
  }
}
else{
  printf("You are not Eligible to work");
}
printf("finished");
}
```

10.4 printBackBelow100.c

Gets a number from user and print all number backward and it must below 100.

Test : nested while loop in if

Draw a SVG file in Inkscape by user, Extractor will convert into SVG data(Figure 1, stage 1).

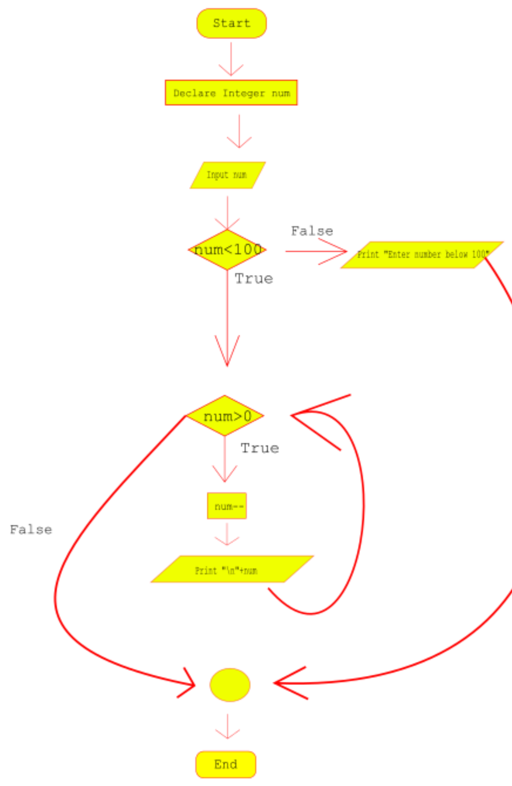


Figure 32. Example program drew in Inkscape

Example fragment code from SVG file.

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

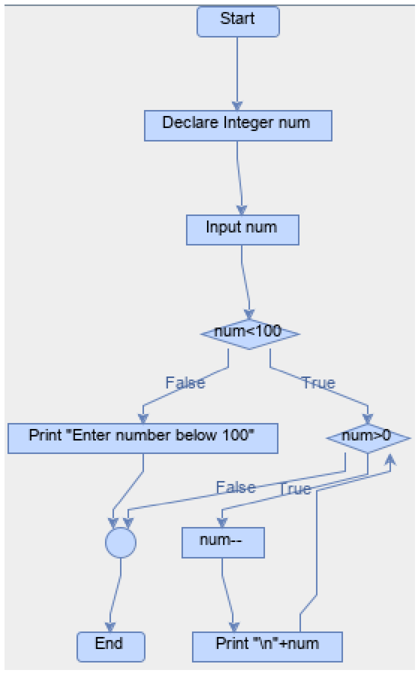


Figure 33. Picture generated from JGraphX

Extractor generates C code from graph data(Figure 1, stage 4).

Generated code from Extractor is fully correct.

```
main(){
int num;
scanf("%d",&num);
if(num<100){
  while(num>0){
    num--;
    printf("\n");
    printf("%d",num);
  }
}
else{
  printf("Enter number below 100");
}
}
```

10.5 printTriangleStar.c

Gets a number from user and print pattern triangle star(*).

Test : two while loop.

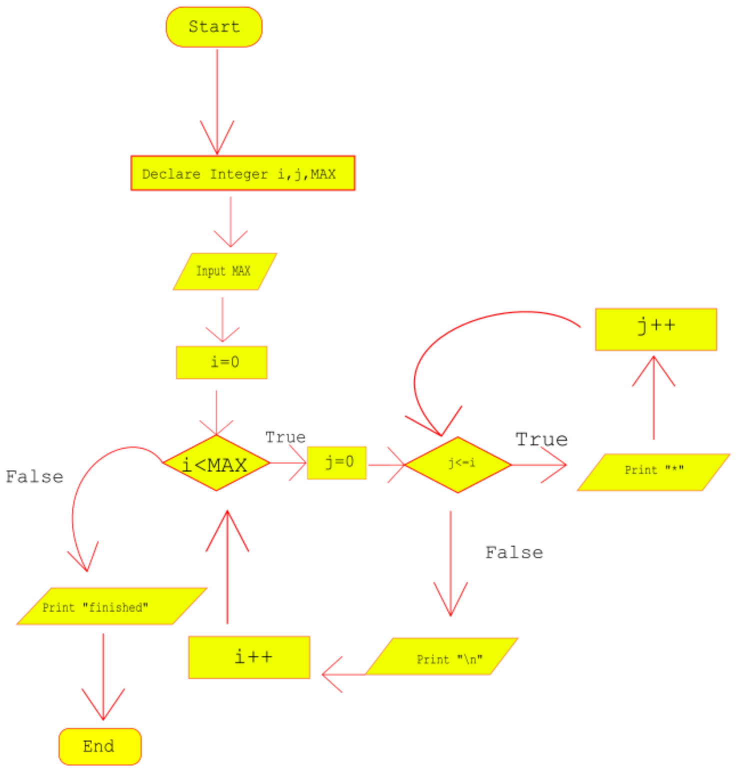


Figure 34. Example program drew in Inkscape

Example fragment code from SVG file.

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

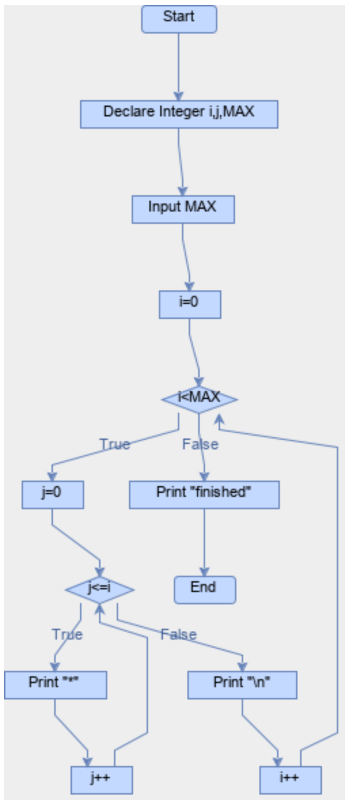


Figure 35. Picture generated from JGraphX

Extractor generates C code from graph data(Figure 1, stage 4).
Generated code from Extractor is fully correct.

```
main(){
int i,j,MAX;
scanf("%d",&MAX);
i=0;
while(i<MAX){
  j=0;
  while(j<=i){
    printf("*");
    j++;
  }
  printf("\n");
  i++;
}
printf("finished");
}
```

10.6 checkArmStrongNumber.c

Gets 3 digits number from user and check if it is an armstrong number or not.

Test : while and if.

Draw a SVG file in Inkscape by user, Extractor will convert into SVG data(Figure 1, stage 1).

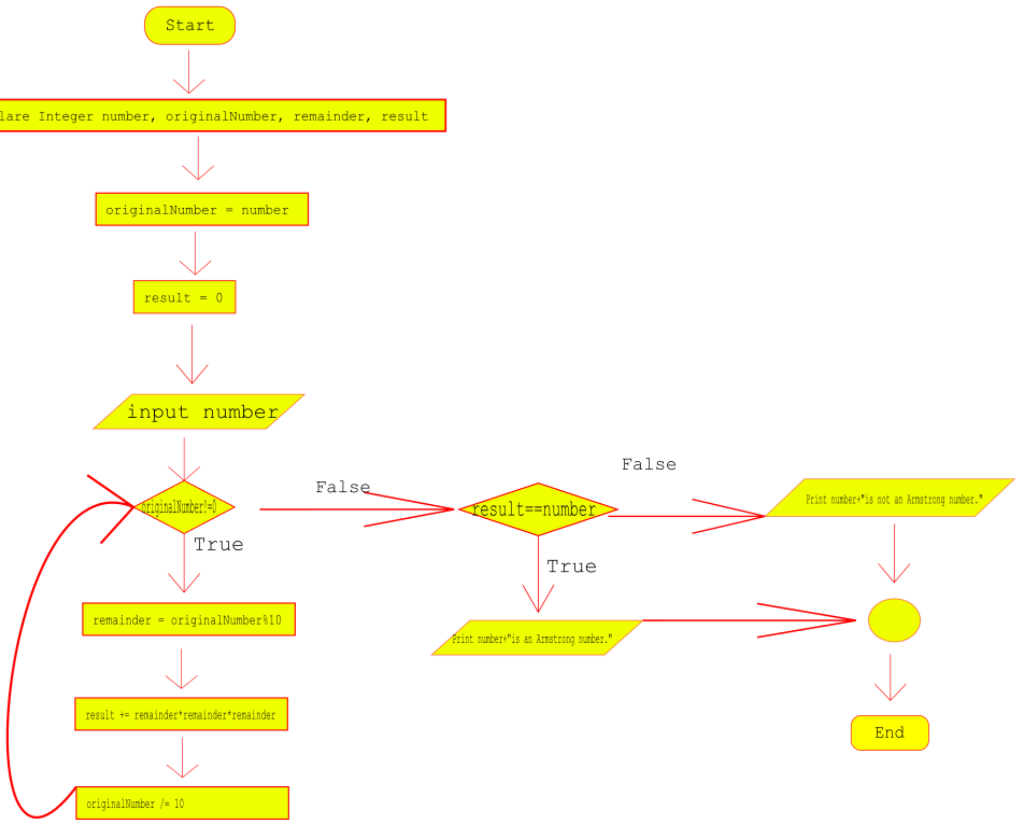

Start
Declare Integer number, originalNumber, remainder, result
originalNumber = number
result = 0
input number
originalNumber!=0
True
remainder = originalNumber%10
result += remainder*remainder*remainder
originalNumber /= 10
False
result==number
True
Print number+"is an Armstrong number."
False
Print number+"is not an Armstrong number."
End

Figure 36. Example program drew in Inkscape

Example fragment code from SVG file.

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

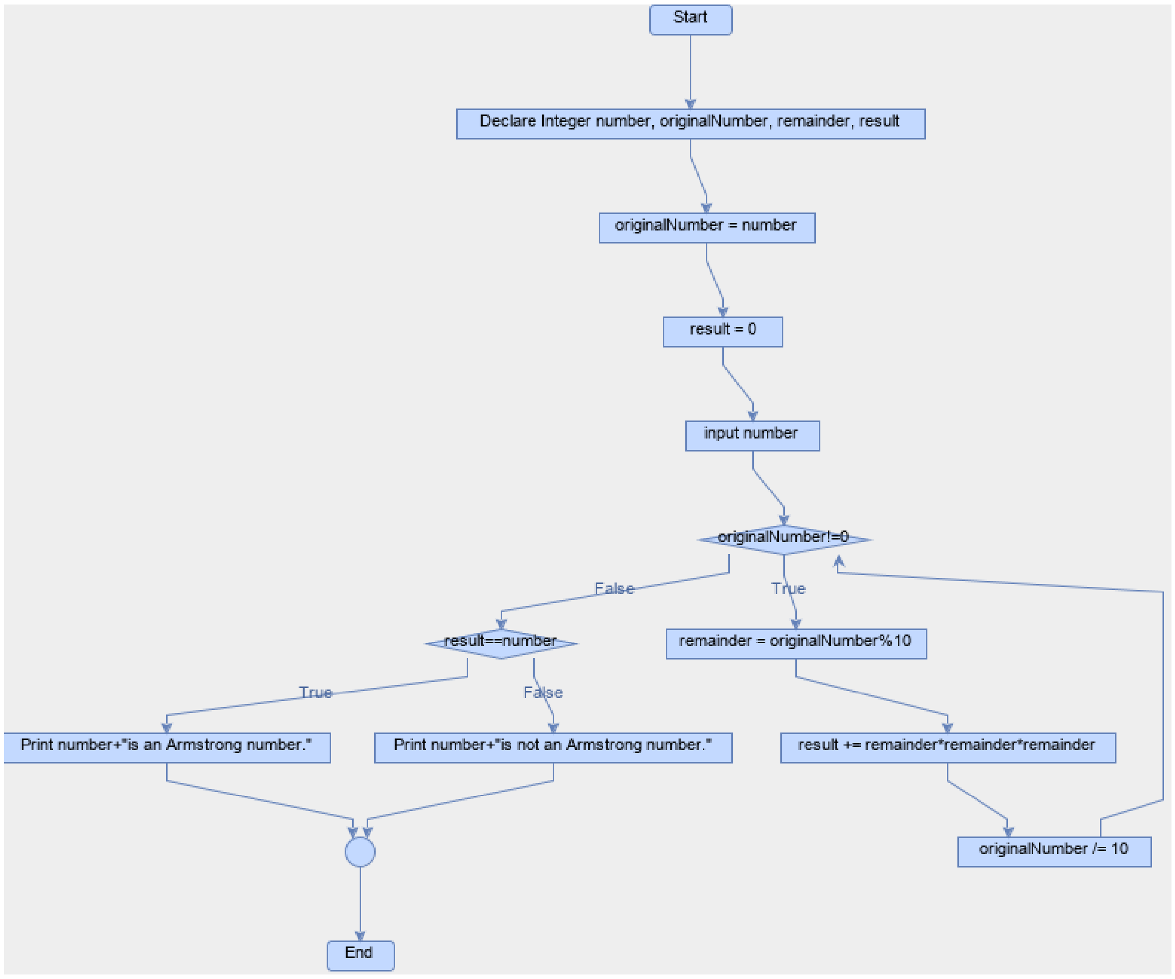

Start
Declare Integer number, originalNumber, remainder, result
originalNumber = number
result = 0
input number
originalNumber!=0
False
True
result==number
remainder = originalNumber%10
True
False
Print number+"is an Armstrong number."
Print number+"is not an Armstrong number."
result += remainder*remainder*remainder
originalNumber /= 10
End

Figure 37. Picture generated from JGraphX

Generated code from Extractor is fully correct.

Generated code from the program

```
main(){
int number, originalNumber, remainder, result;
originalNumber = number;
result = 0;
input number;
while(originalNumber!=0){
  remainder = originalNumber%10;
  result += remainder*remainder*remainder;
  originalNumber /= 10;
}
if(result==number){
  printf("%d",number);
  printf("is an Armstrong number.");
}
else{
  printf("%d",number);
  printf("is not an Armstrong number.");
}
}
```

10.7 findNPrimeNumbers.c

gets a number from user and print prime numbers for n numbers.

Test : if nested in nested while loop.

Draw a SVG file in Inkscape by user, Extractor will convert into SVG data(Figure 1, stage 1).

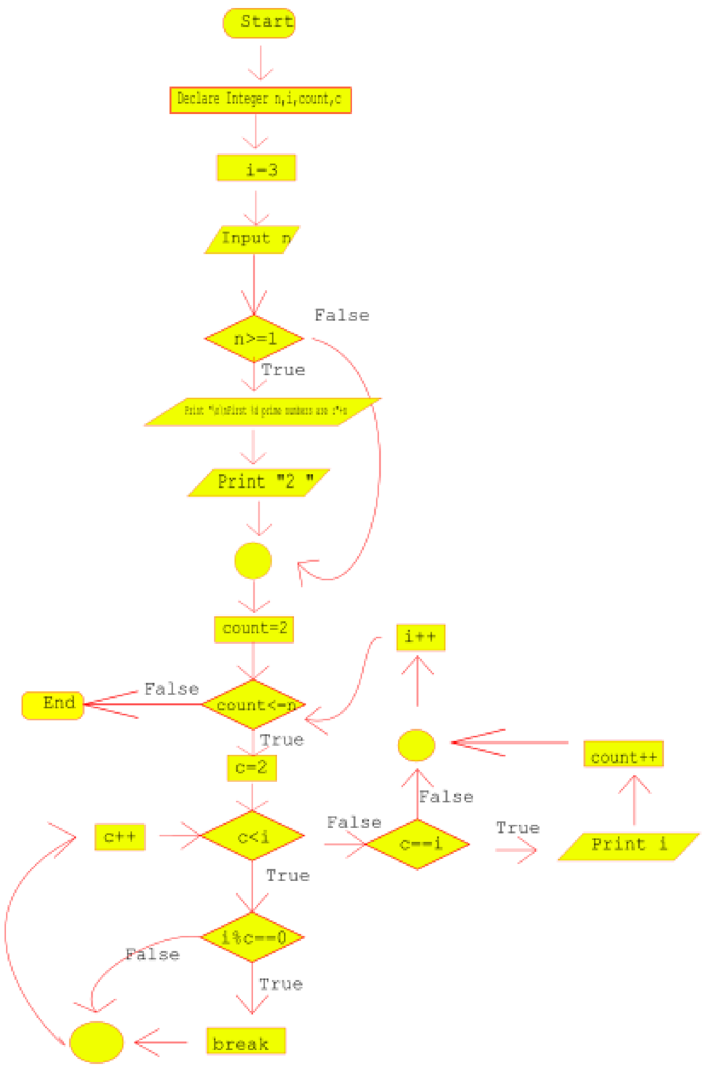


Figure 38. Example program drew in Inkscape

Example fragment code from SVG file.

Extractor extracts SVG data to get graph data by using JGraphT(Figure 1, stage 2), then generate flowchart using JGraphX(Figure 1, stage 3).

Generated picture from Extractor is not fully correct because input/output shapes cound not be generated in rhombus shapes.

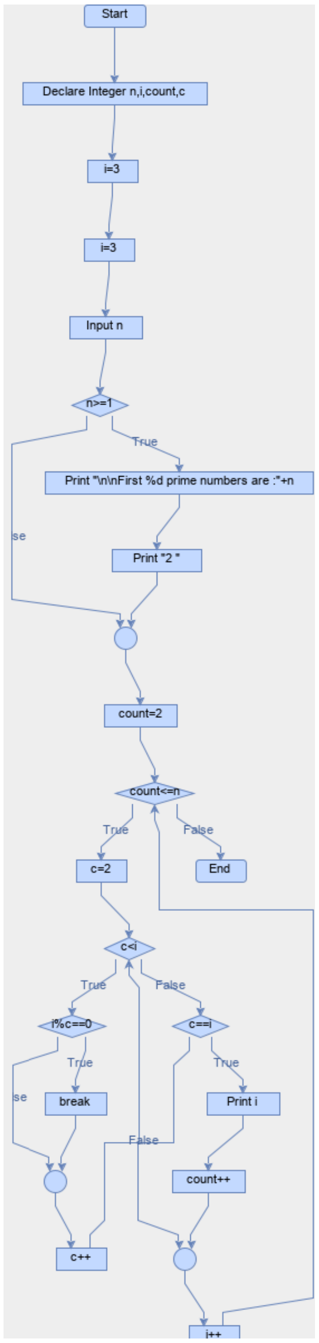


Figure 39. Picture generated from JGraphX

Extractor generates C code from graph data(Figure 1, stage 4).

Generated code from the program

```
main(){
int n,i,count,c;
i=3;
scanf("%d",&n);
if(n>=1){
  printf("\n\nFirst %d prime numbers are :");
  printf("%d",n);
  printf("2 ");
}
else{

}
count=2;
while(count<=n){
  c=2;
  while(c<i){
    if(i%c==0){
      break;
    }
    else{

    }
    c++;
  }
  if(c==i){
    printf("%d",i);
    count++;
  }
  else{

  }
  i++;
}
}
```

Chapter 11. Extraction of SVG Data

Extractor.java (Appendix B) is a program to extract data from Flowchart SVG file(Figure 1, stage 1), generate Flowchart picture(Figure 1, stage 2) and convert to C code(Figure 1, stage 3). This chapter will explain how Extractor.java extract SVG file.

The program currectly can be drew with many shapes and symbols included arrow(flow line), rectangle(process), diamond(conditional), rhombus(input/output), Oval(terminal) and text. (see more in Chapter 2)

Function main is a function to create new SVGuniverse and import a SVG file to diagram then call function looper with the main element of SVG file. Return the summation number of all object(Figure 1, stage 1).

```
SVGUniverse svgUniverse = new SVGUniverse();
SVGDiagram diagram = svgUniverse.getDiagram(uri);
int objCount = 0;
for(int i = 0; i < diagram.getRoot().getNumChildren(); i++) {
  SVGElement elem  = diagram.getRoot().getChild(i);
  objCount = findElem(elem,objCount);
}
```

Function findElem is a function to check if the element which received has child or not. Number of objCount increases according to number of node and return objCount.If the element has no more child the function will call storeNode to store this element.

```
public static int findElem(SVGElement elem,int objCount)
{
  objCount++;
  for(int i = 0; i < elem.getNumChildren(); i++) {
    storeNode(elem,objCount,i);
    SVGElement element  = elem.getChild(i);
    objCount = findElem(element, objCount);
  }
  return objCount;
}
```

Function storeNode is a function to store all nodes to arraylist SVGNode and determine type of received shape.

```
public static void storeNode(SVGElement elem,int i,int objCount)
{
  float[] tmpx = new float[4];
  float[] tmpy = new float[4];
  String tText = "";
  String type = "";
  String att[] =
{"x","y","width","height","rx","ry","transform","xml:space","sodipodi:role"};
  AffineTransform matrix = new AffineTransform(0,0,0,0,0,0);
  SVGElement element = elem.getChild(objCount);
  try{
    for(int j=0;j<att.length;j++){
      if(element.hasAttribute(att[j],AnimationElement.AT_AUTO)){
        if(att[j]=="x"){
          tmpx[0] = getPresValue(element,att[j]);
          type = "terminal";
        }
      }
      Rectangle2D.Float(tmpx[0],tmpy[0],tmpx[1],tmpy[1]);
      SVGNode temp = new Rect(element.getId(),"terminal",tempRect);
      nodes.add(temp);
    }
```

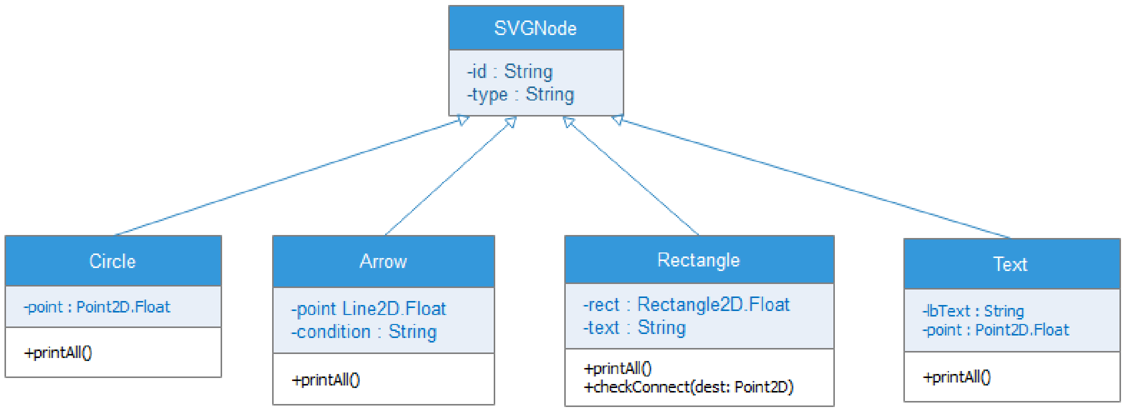


Figure 40. UML class diagram of SVGNode.

Figure 40 shows UML class diagram of SVGNode. Superclass SVGNode and 4 Subclasses Circle, Rectangle, Arrow and Text use to store node which extracted from function storeNode (Appendix C).

```
public static class SVGNode
{
  String id;
  String type;
  public SVGNode(String id,String type)
  {
    this.id = id;
    this.type = type;
  }
  public boolean isShape(String shape)
  {
    if(type.contains(shape))
      return true;
    return false;
  }
}
public static class Circle extends SVGNode
{
  Point2D.Float point;
  public Circle(String id,String type,float iX,float iY)
  {
    super(id,type);
    this.point = new Point2D.Float(iX,iY);
  }
  public void printAll()
  {
    System.out.println("\nID :"+id);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    System.out.println("\tPosition(X,Y) : "+point.toString());
  }
}
```

Function compareNode and printConnect work together to add vertexes and edges to JGraphT.

```
public static void compareNode(int objCount)
{
  for(SVGNode proc : nodes){
    if(proc instanceof Arrow){
      Arrow arrow = (Arrow)proc;
      SVGNode connection1 = printConnect(arrow.point.getP1(),"Arrow");
      SVGNode connection2 = printConnect(arrow.point.getP2(),"Arrow");
      if(connection1!=null&&connection2!=null){
        diGraph.addVertex(connection1);
        diGraph.addVertex(connection2);
        diGraph.addEdge(connection1,connection2,arrow);
      }
    }
  }
```

Function printConnect compares starting point and destination point of each arrow with all other symbols then return the closest symbol.

```
public static SVGNode printConnect(Point2D point,String checkType)
{
  String txt = "";
  double distanceRect=1000,distanceArrow=1000;
  Rect nearestRect = null;
  Arrow nearestArrow = null;
  for(SVGNode proc : nodes){
    if(proc instanceof Rect){
      Point2D.Float centerPoint =
       new Point2D.Float((float)((Rect)proc).rect.getCenterX(),(float)
       ((Rect)proc).rect.getCenterY());
      if(point.distance(centerPoint)<distanceRect){
        nearestRect = ((Rect)proc);
        distanceRect = point.distance(centerPoint);
      }
    }
    if(proc instanceof Arrow){
      if(((Arrow)proc).point.ptSegDist(point)<distanceArrow){
        nearestArrow = ((Arrow)proc);
        distanceArrow = ((Arrow)proc).point.ptSegDist(point);
      }
    }
  }
  return nearestRect;
}
```

Chapter 12. Using JGraphT

This chapter shows example codes from JgraphT inside Extractor program(Figure 1, stage 2).

12.1 Prints all cycles

TarjanSimpleCycles is a class that can find cycles by fucntion findSimpleCycles. This program uses it to find all loops in the diagram.

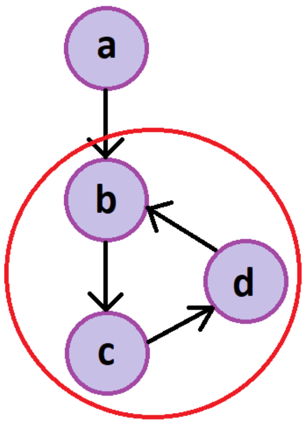


Figure 41. cycle(loop) example

Figure 41 shows red circle around loop inside the example from b to d.

```
TarjanSimpleCycles tsCycles = new TarjanSimpleCycles<SVGNode, Arrow>
(diGraph);
for (int i = 0; i < tsCycles.findSimpleCycles().size(); i++) {
  System.out.println("\nCycles:");
  List<Rect> s = (List<Rect>)tsCycles.findSimpleCycles().get(i);
  for(int j = 0; j < s.size(); j++){
    Rect z = (Rect)s.get(j);
    System.out.println(z.text);
  }
}
```

12.2 Prints all paths

AllDirectedPaths is a class to find all directed paths by fucntion getAllPaths. A directed graph (or digraph) is a graph that is a set of vertices connected by edges, where the edges have a direction associated with them.

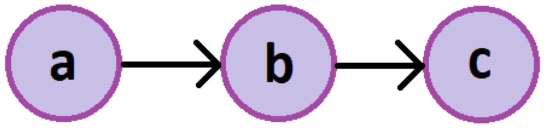


Figure 42. directed path example

Figure 42 shows directed path example from a to c.

```
// find start point and target point of the graph
SVGNode start = null;
SVGNode dest = null;
for(SVGNode proc : nodes){
  if(proc instanceof Rect){
    if(((Rect)proc).text.equalsIgnoreCase("start"))
      start = (Rect)proc;
    if(((Rect)proc).text.equalsIgnoreCase("end"))
      dest = (Rect)proc;
  }
}
AllDirectedPaths<SVGNode, Arrow> allPaths = new AllDirectedPaths<> (diGraph);
List<GraphPath<SVGNode,Arrow>>allPathsList=
 allPaths.getAllPaths(start,dest,true,20);
System.out.println("\nAll paths from Start to End:");
for(GraphPath p : allPathsList) {
  System.out.println("\nPath###");
  SVGNode source,target = null;
  List<Arrow> allPathsEdgeList = p.getEdgeList();
  for(Arrow pl : allPathsEdgeList){
    source = (SVGNode)p.getGraph().getEdgeSource(pl);
    target = (SVGNode)p.getGraph().getEdgeTarget(pl);
    if(source instanceof Rect)
      System.out.println("\t->"+((Rect)source).text+" ("+source.type+")
        "+pl.condition);
  }
  if(target instanceof Rect)
    System.out.println("\t->"+((Rect)target).text+" ("+target.type+")");
}
```

12.3 Prints the shortest path

DijkstraShortestPath is an algorithm which computes shortest paths between vertices by function getPath. Shortest path is the shortest path between a source vertex to a target vertex.

Example is the shortest path between a and c in the example graph is a > b > c.

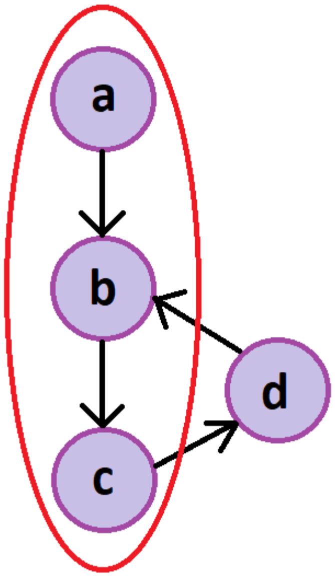


Figure 43. shortest path example

Figure 43 shows red cricle around shortest path from a to c.

```
//takes directed graph into DijkstraShortestPath class to get the path
DijkstraShortestPath<SVGNode, Arrow> dsPath = new
DijkstraShortestPath<>(diGraph);
  GraphPath<SVGNode, Arrow> shortestPath = dsPath.getPath(start,dest);
  System.out.println("\nShortest path from Start to End:");
  List<SVGNode> shortestVertexPath = shortestPath.getVertexList();
  for(SVGNode p : shortestVertexPath)
    if(source instanceof Rect)
      System.out.println("\t->"+((Rect)source).text+"  ("+source.type+")
       "+pl.condition);
```

Chapter 13. Using JGraphX

This chapter shows example codes and outputs from JgraphX inside Extractor program (Figure 1, stage 3).

13.1 JGraphX codes example

JGraphX codes for Extractor.java using JGraphXadapter to convert JGraphT to JGraphX and use JApplet to generate picture.

```
Extractor applet = new Extractor();
  jgxAdapter = new JGraphXAdapter<>(diGraph);
  applet.init();
public void init()
{
  //JGrapX initialize
  jgxAdapter = new JGraphXAdapter<>(diGraph);
  setPreferredSize(DEFAULT_SIZE);
  mxGraphComponent component = new mxGraphComponent(jgxAdapter);
  component.setConnectable(false);
  component.getGraph().setAllowDanglingEdges(false);
  getContentPane().add(component);
  resize(DEFAULT_SIZE);
}
// changes value of shape from name into text label
mxCell cell = (mxCell)jgxAdapter.getDefaultParent();
for(int i=0;i<cell.getChildCount();i++){
  mxICell icell = cell.getChildAt(i);
  if(icell.getValue() instanceof Rect){
    String strTemp = ((Rect)icell.getValue()).text;
    String type = ((Rect)icell.getValue()).type;
    icell.setValue(((Rect)icell.getValue()).text);
    icell.setGeometry(new mxGeometry(0,0,strTemp.length()*5+30,20));
    Map<String, Object> styles =
     new HashMap<String,
Object>(jgxAdapter.getView().getState(icell).getStyle());
     // configs styles of JGraphX with mxConstants class which contains
styles of mxGraph
    styles.put(mxConstants.STYLE_FONTCOLOR, "black");
    if(type.equals("rectangle"))
      styles.put(mxConstants.STYLE_SHAPE, mxConstants.SHAPE_RECTANGLE);
    else if(type.equals("diamond"))
      styles.put(mxConstants.STYLE_SHAPE, mxConstants.SHAPE_RHOMBUS);
    else if(type.equals("oval"))
      styles.put(mxConstants.STYLE_ROUNDED, "1");
    jgxAdapter.getView().getState(icell).setStyle(styles);
    icell.setVisible(true);
  }
  if(icell.getValue() instanceof Circle){
    icell.setValue(null);
    icell.setGeometry(new mxGeometry(0,0,20,20));
    Map<String, Object> styles =
     new HashMap<String,
Object>(jgxAdapter.getView().getState(icell).getStyle());
    styles.put(mxConstants.STYLE_SHAPE, mxConstants.SHAPE_ELLIPSE);
    jgxAdapter.getView().getState(icell).setStyle(styles);
    icell.setVisible(true);
  }

  // changes edge value as true or false if previous is diamond shape
  if(icell.getValue() instanceof Arrow)
    icell.setValue(((Arrow)icell.getValue()).condition);
}

// transforms graph layout as Hierarchical style
mxHierarchicalLayout layout = new mxHierarchicalLayout(jgxAdapter);;
layout.execute(jgxAdapter.getDefaultParent());

// genarates JApplet
JFrame frame = new JFrame();
frame.getContentPane().add(applet);
frame.setTitle("JGraphT Adapter to JGraphX");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.pack();
frame.setVisible(true);
```

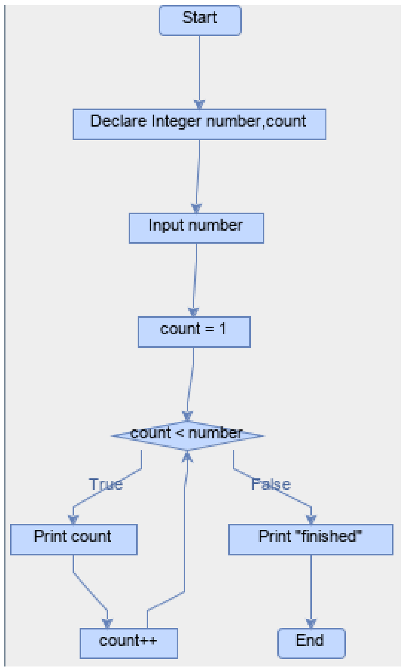


Figure 44. JGraphX output of printAllNumbers.c

Figure 44 shows JGraphX output of an example printAllNumbers.c in chapter 10.1.

Chapter 14. Generating C code

Converter part start working after extractor part extracted diagram. Converter will look into extracted information and convert to codes(Figure 1, stage 4). I'm still at work with it and it is complicated such as when the converter found a diamond shape. It might be while loop, if-else or do while loop then before the program can convert it the program need to process it first. Here is some example codes for this.

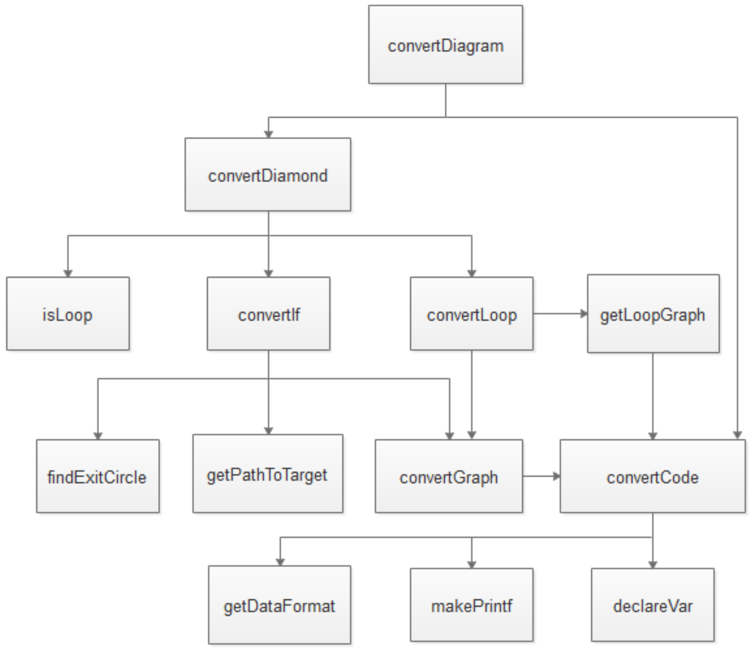


Figure 45. Convert part code overview.

Figure 45 shows code overview of generating C code included with all implicated functions.

When the program found diamond it will call convertDiamond to check for a loop, then this function will call convertLoop or convertIf.

```
public static ConvertInfo convertDiamond(SVGNode start,String sps)
//converts loop or if-else graph then return text and next node
{
  ConvertInfo info = null;
  if(isLoop(start))
    info = convertLoop(start,sps);
  else
    info = convertIf(start,sps);
  return info;
}
```

Function isLoop checks loop of current node from number of incoming edges.

```
public static boolean isLoop(SVGNode node)
{
  /*This can prevert errors in case if-else inside while
    While loop always has 2+ incoming edges
  */
  Set<Arrow> edgesSet = diGraph.incomingEdgesOf(node);
  List<Arrow> edgesList = new ArrayList<>(edgesSet);
  if(edgesSet.size()>1)
    return true;
  else
    return false;
}
```

Function convertIf converts if and else path into string and finds the next node to process.

```
public static ConvertInfo convertIf(SVGNode start,String sps)
{
  //gets first node of both side of current node
  SVGNode trueBranch = getBranch(start,true);
  SVGNode falseBranch = getBranch(start,false);
  /*gets true and false paths to the end of diagram
    if there are more paths then calling another convertDiamond
    will deal with more paths
  */
  Graph truePaths = getPathToTarget(trueBranch,null);
  Graph falsePaths = getPathToTarget(falseBranch,null);

  //finds both paths and returns a node where both paths meet each other
  SVGNode nextNode = findExitCircle(truePaths,falsePaths,start);

  //gets paths to that node
  truePaths = getPathToTarget(trueBranch,nextNode);
  falsePaths = getPathToTarget(falseBranch,nextNode);
  String ifString = sps+"if("+((Rect)start).text+"){\n"+
   convertGraph(truePaths,trueBranch,sps+"  ",false)+"\n"+sps+"}\n"+
   sps+"else{\n"+convertGraph(falsePaths,falseBranch,sps+"
   ",false)+"\n"+sps+"}";
  return new ConvertInfo(ifString,nextNode);
}
```

Function convertLoop converts loop into string and finds the next node to process.

```
public static ConvertInfo convertLoop(SVGNode start,String sps)
{
  //gets first node of both side of current node
  SVGNode trueBranch = getBranch(start,true);
  SVGNode falseBranch = getBranch(start,false);
  SVGNode nextNode = null;
  String loopString="";

  //CycleDetector checks loops in the graph
  CycleDetector cDetector = new CycleDetector(diGraph);
  //checks if there is a loop in any side of the condition
  if(cDetector.detectCyclesContainingVertex(trueBranch)){
    Graph trueLoopBranch = getLoopGraph(start);

    //converts graph to code by calling convertGraph method
    loopString = sps+"while("+((Rect)start).text+"){\n"+
     convertGraph(trueLoopBranch,start,sps+"  ",true)+"\n"+sps+"}";
    nextNode = falseBranch;
  }
  else if(cDetector.detectCyclesContainingVertex(falseBranch)){
    Graph falseLoopBranch = getLoopGraph(start);
    loopString = sps+"while(!"+((Rect)start).text+"){\n"+
     convertGraph(falseLoopBranch,start,sps+"  ",true)+"\n"+sps+"}";
    nextNode = trueBranch;
  }
  return new ConvertInfo(loopString,nextNode);
}
```

Chapter 15: Conclusions and Future Work

15.1 Conclusions

At this time users can draw Flowchart with Inkscape and Shape Creator extension with different shapes included Flow line, Terminal, Process, Condional, Input/Output and Circle.

Extractor can deal with many cases which included in following shapes and using SVGsalamander to extract and convert to SVG data then use JGraphT to get graph data.

Extractor will generate Flowchart picture by using JGraphX for user to comparing with user's diagram to check correction.

Extractor generates C code from graph data which user can use and learn from this code later.

15.2 Future Work

For future work, the program can improve by adding more shapes to extract and fix converting of the converting to C code part and add more type of loop which is for loop.

References

[1] Barak Naveh, JGraphT Available from:
https://github.com/jgrapht/jgrapht

[2] Blackears**,** SVGsalamander [Online] Available
from: https://github.com/blackears/svgSalamander/wik [Accessed: Feb 8th 2018]

[3] CS Odessa, *Basic Flowchart Symbols and Meaning* [Online] Available
from:http://www.conceptdraw.com/How-To-Guide/flowchart-symbols [Accessed: Feb 1st 2018]

[4] JGraph Ltd, JGraphX [Online] Available from :
https://jgraph.github.io/mxgraph/docs/manual_javavis.html [Accessed: October 11th 2018]

[5] Nassi, I.; Shneiderman, B. *Flowchart techniques for structured programming*, SIGPLAN Notices XII, August 1973.

[6] Tony Gaddis. *Starting Out with Programming Logic & Design (4th Edition)*. Haywood Community College, Pearson, February 2015.

[7] Wikipedia, *Flow Chart*. [Online] Available from:
https://en.wikipedia.org/wiki/Flowchart [Accessed: Jan 25th 2018]

[8] Wikipedia, *Nassi–Shneiderman* [Online] Available from:
https://en.wikipedia.org/wiki/Nassi–Shneiderman_diagram [Accessed: Feb 1st 2018]

Appendix A
SVGTest.java

```
// Andrew Davison, ad@fivedots.coe.psu.ac.th, Jan. 2018
/* Parsing an SVG file using SVG Salamander
   (https://github.com/blackears/svgSalamander)

*/
import java.io.*;
import java.util.*;
import java.net.*;
import com.kitfox.svg.*;
import com.kitfox.svg.xml.*;
public class SVGTest
{
   public static void main(String[] args)
  {
    if (args.length != 1) {
       System.out.println("Usage: SVGTest <SVG
       fnm>");
      return;
    }
    try {
      SVGUniverse svgUniverse = new SVGUniverse();
      File file = new File(args[0]);
      SVGDiagram diagram = svgUniverse.ge
      tDiagram(fil
      e.toURI());
    /* load generates a warning and a stack trace -- ignored */
    for(int i = 0; i < diagram.getRoot().getNu
              mChildren(); i++)
    {
      SVGElement element  = diagram.getRoot
                   ().getChild(i);
      printAll(element, "  ");
    }
  }
  catch(Exception e)
  {  System.out.println(e);  }
}  // end of main()
private static void printAll(SVGElement elem, String tab)
{
  System.out.println(tab + "ID: " + elem.getId());
  if (elem instanceof ShapeElement)
    printDetails(elem, tab+ "  ");
     for(int i = 0; i < elem.getNumChildren(); i++) {
    SVGElement child  = elem.getChild(i);
    printAll(child, tab+ "  ");
  }
}  // end of printAll()

private static void printDetails(SVGElement elem, String tab)
{
  ShapeElement se = (ShapeElement) elem;
  if (se.getShape() != null)
          System.out.println(tab + "Shape: " +
            se.getShape());

  if (se instanceof Tspan) {    // text span
    Tspan node = (Tspan)se;
    System.out.println(tab + node.getText());
  }
  else if (se instanceof Rect) {    // rectangle
    showAtt(elem, "x", tab);
    showAtt(elem, "y", tab);
    showAtt(elem, "width", tab);
    showAtt(elem, "height", tab);
    showStyle(elem, "fill", tab);
    //Rect node = (Rect)se;
  }
  else if (se instanceof Text) {    // text box
    showAtt(elem, "x", tab);
    showAtt(elem, "y", tab);
    // Text node = (Text)se;
  }
  else if (se instanceof Path) {    // polygon
    showAtt(elem, "sodipodi:sides", tab);  //
             inkscape specific
    showAtt(elem, "sodipodi:cx", tab);
    showAtt(elem, "sodipodi:cy", tab);

    String trStr = getAtt(elem, "transform");
    System.out.println(tab + "transform: " +
              trStr);
    Transform tr = Transform.parse(trStr);
    tr.print();
    showStyle(elem, "fill", tab);
    // Path node = (Path)se;
  }
}  // end of printDetails()

private static void showAtt(SVGElement elem, String attNm, String tab)
{
  try {
    StyleAttribute attrib = new StyleAttrib
              ute(attNm);
    elem.getStyle(attrib);
    double value = attrib.getDoubleValue();
    if (value != 0)
      System.out.println(tab + attNm + ": " + value);
  }
    catch(SVGException e)
  { System.out.println(attNm + " attribute error: " + e);  }
}  // end of showAtt()
private static String getAtt(SVGElement elem, String attNm)
{
  try {
    StyleAttribute attrib = new StyleAttribute(attNm);
    elem.getStyle(attrib);
    return attrib.getStringValue();
  }
  catch(SVGException e)
  { System.out.println(attNm + " attribute error: " + e);
    return null;
  }
}  // end of getAtt()
private static void showStyle(SVGElement elem, String styleNm, String
tab)
/* format of "style" string is:
  style="fill:#ffff00;fill-rule:evenodd
      ;stroke:#000000;
  stroke-width:1px;stroke-linecap:butt;
    stroke-linejoin:miter;stroke-opacity:1"
  */
  {
    try {
      StyleAttribute attrib = new StyleAttrib
                    ute("style");
      elem.getStyle(attrib);
      String styleStr = attrib.getStringValue();
      String[] kvPairs = styleStr.split(";");   //
                    split string   into pairs
      Map<String,String> map = new HashMap<>();
      for(String pair : kvPairs) {
        String[] entry = pair.split(":");    //
                          split each pair
        map.put(entry[0].trim(), entry[1].trim());
      }
      String styleVal = map.get(styleNm);
      if (styleVal != null) {
        if (styleNm.equals("fill"))
   System.out.println(tab + styleNm + ": " +
          toRGB(styleVal));
        else
```

```
        System.out.println(tab + styleNm + ": "
                               + styleVal);
    }
  }
  catch(SVGException e)
  { System.out.println(styleNm + " style error:
           " + e);  }
}  // end of showStyle()

private static String toRGB(String colorStr)
// format is "#ffffff"
{
  int r = Integer.valueOf( color
          Str.substring(1,3), 16);
  int g = Integer.valueOf( color
          Str.substring(3,5), 16);
         int g = Integer.valueOf( color
         Str.substring(3,5), 16);
  int b = Integer.valueOf( c olorStr.su
          bstring(5,7), 16);
    retu  rn   "RGB: (" + r + "," + g + "," + b +
           ")";
}  // end of toRGB()
 }  // end of SVGTest class
```

Appendix B
Extractor.java

```
public class Extractor extends JApplet
{
  private static final Dimension DEFAULT_SIZE =
   new Dimension(400, 600);
  static ListenableGraph<SVGNode, Arrow> diGraph =
   new DefaultListenableGraph<>(
   new DefaultDirectedGraph<SVGNode, Arrow>(Arrow.class));
  static private JGraphXAdapter<SVGNode, Arrow> jgxAdapter;
  static Stack stack = new Stack();
  static ArrayList<SVGNode> nodes = new ArrayList<SVGNode>();
  static Map<String,String> varMap = new HashMap<String,String>();

  public static void main(String[] args)
  {
    Extractor applet = new Extractor();
    jgxAdapter = new JGraphXAdapter<>(diGraph);
    applet.init();

    char cChecker = '1';
    URI uri = null;
    try {
      File f = new File(args[0]);
      uri = f.toURI();
    }
    catch (Exception e) {
      System.out.println(e);
    }
    SVGUniverse svgUniverse = new SVGUniverse();
    SVGDiagram diagram = svgUniverse.getDiagram(uri);
    int objCount = 0;
    for(int i=0;i<diagram.getRoot().getNumChildren();i++){
      SVGElement elem =diagram.getRoot().getChild(i);
      objCount = findElem(elem,objCount);
    }
    printNode();
    compareNode(objCount);

    // changes value of shape from name into text label
    mxCell cell = (mxCell)jgxAdapter.getDefaultParent();
    for(int i=0;i<cell.getChildCount();i++){
      mxICell icell = cell.getChildAt(i);
      if(icell.getValue() instanceof Rect){
        String strTemp = ((Rect)icell.getValue()).text;
        String type = ((Rect)icell.getValue()).type;
        icell.setValue(((Rect)icell.getValue()).text);
        icell.setGeometry(
         new mxGeometry(0,0,strTemp.length()*5+30,20));
        Map<String, Object> styles =
         new HashMap<String, Object>(jgxAdapter.getView().
         getState(icell).getStyle());
         // configs styles of JGraphX with
        //mxConstants class which contains styles of mxGraph
        styles.put(mxConstants.STYLE_FONTCOLOR, "black");
        if(type.equals("process"))
          styles.put(mxConstants.STYLE_SHAPE,
           mxConstants.SHAPE_RECTANGLE);
        else if(type.equals("condition"))
          styles.put(mxConstants.STYLE_SHAPE,
           mxConstants.SHAPE_RHOMBUS);
        else if(type.equals("terminal"))
          styles.put(mxConstants.STYLE_ROUNDED, "1");
        jgxAdapter.getView().getState(icell).setStyle(styles);
        icell.setVisible(true);
      }
      if(icell.getValue() instanceof Circle){
        icell.setValue(null);
        icell.setGeometry(new mxGeometry(0,0,20,20));
        Map<String, Object> styles =
         new HashMap<String, Object>(jgxAdapter.getView().
          getState(icell).getStyle());
        styles.put(mxConstants.STYLE_SHAPE, mxConstants.SHAPE_ELLIPSE);
        jgxAdapter.getView().getState(icell).setStyle(styles);
        icell.setVisible(true);
      }

      //changes edge value as true or false
      //if previous is diamond shape
      if(icell.getValue() instanceof Arrow)
        icell.setValue(((Arrow)icell.getValue()).condition);
    }

    // transforms graph layout as Hierarchical style
    mxHierarchicalLayout layout =
     new mxHierarchicalLayout(jgxAdapter);;
    layout.execute(jgxAdapter.getDefaultParent());

    // genarates JApplet
    JFrame frame = new JFrame();
    frame.getContentPane().add(applet);
    frame.setTitle("JGraphT Adapter to JGraphX");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.pack();
    frame.setVisible(true);

    printJGraphT();

    while(!stack.empty())
      System.out.println(stack.pop());
  }
  public void init()
  {
    //JGrapX initialize
    jgxAdapter = new JGraphXAdapter<>(diGraph);
    setPreferredSize(DEFAULT_SIZE);
    mxGraphComponent component = new mxGraphComponent(jgxAdapter);
    component.setConnectable(false);
    component.getGraph().setAllowDanglingEdges(false);
  getContentPane().add(component);
  resize(DEFAULT_SIZE);
}
static public void printJGraphT()
{
  // prints all cycles
  TarjanSimpleCycles tsCycles =
   new TarjanSimpleCycles<SVGNode, Arrow> (diGraph);
  for (int i = 0; i < tsCycles.findSimpleCycles().size(); i++) {
    System.out.println("\nCycles:");
    List<Rect> s = (List<Rect>)tsCycles.findSimpleCycles().get(i);
    for(int j = 0; j < s.size(); j++){
      if(s instanceof Rect){
        Rect z = (Rect)s.get(j);
      }
    }
  }

  // find start point and target point of the graph
  SVGNode start = null;
  SVGNode dest = null;
  for(SVGNode proc : nodes){
    if(proc instanceof Rect){
      if(((Rect)proc).text.equalsIgnoreCase("start"))
        start = (Rect)proc;
      if(((Rect)proc).text.equalsIgnoreCase("end"))
        dest = (Rect)proc;
    }
  }

  // prints all paths
  AllDirectedPaths<SVGNode, Arrow> allPaths =
   new AllDirectedPaths<> (diGraph);
  List<GraphPath<SVGNode, Arrow>> allPathsList =
   allPaths.getAllPaths(start,dest,true,20);
  System.out.println("\nAll paths from Start to End:");
  for(GraphPath p : allPathsList) {
    System.out.println("\nPath###");
    SVGNode source,target = null;
    List<Arrow> allPathsEdgeList = p.getEdgeList();
    for(Arrow pl : allPathsEdgeList){
      source = (SVGNode)p.getGraph().getEdgeSource(pl);
      target = (SVGNode)p.getGraph().getEdgeTarget(pl);
      if(source instanceof Rect)
        System.out.println("\t->"+((Rect)source).text+"
         ("+source.type+") "+pl.condition);
    }
    if(target instanceof Rect)
      System.out.println("\t->"+((Rect)target).text+"
       ("+target.type+")");
  }

  // prints the shortest path
  DijkstraShortestPath<SVGNode, Arrow> dsPath =
   new DijkstraShortestPath<>(diGraph);
  GraphPath<SVGNode, Arrow> shortestPath =
   dsPath.getPath(start,dest);
  System.out.println("\nShortest path from Start to End:");
    SVGNode source,target = null;
    List<Arrow> allPathsEdgeList = shortestPath.getEdgeList();
    for(Arrow pl : allPathsEdgeList){
      source = (SVGNode)shortestPath.getGraph().getEdgeSource(pl);
      target = (SVGNode)shortestPath.getGraph().getEdgeTarget(pl);
      if(source instanceof Rect)
        System.out.println("\t->"+((Rect)source).text+"
         ("+source.type+") "+pl.condition);
    }
    if(target instanceof Rect)
      System.out.println("\t->"+((Rect)target).text+"
       ("+target.type+")");

  String result = convertDiagram(shortestPath.getStartVertex());
  System.out.println(result);
}

public static void declareVar(String body,String var)
//add new variable by put variables into varMap
{
  //deals when user declare multiple variables at once
  if(body.contains(",")){
    String[] strArray = body.split(",");
    for(String str : strArray){
      //deal with variable which initialized with value
      if(str.contains("="))
        str = str.split("=")[0];
      //put variable and its type in varMap
      varMap.put(str,var);
    }
  }
  else {
    if(body.contains("="))
       body = body.split("=")[0];
    varMap.put(body,var);
  }
}

public static String makeCode(String body,String sps)
//converts text to code
{
  if(body.contains("Start"))
    return "main(){";
  else if (body.contains("End"))
    return "}";
  else if (body.contains("Set")){
    body = body.replaceAll("Set ","");
    return body;
  }
  else if (body.contains("Declare")){
    if(body.contains("Integer")){
      body = body.replaceAll("Declare Integer ","");

      //calls declareVar to put variable into variable map
    declareVar(body,"integer");
    return "int "+body+";";
  }
  if(body.contains("Unsigned Integer")){
    body = body.replaceAll("Declare Unsigned Integer ","");
    declareVar(body,"unsigned integer");
    return "unsigned int "+body+";";
  }
  else if(body.contains("String")){
    body = body.replaceAll("Declare String ","");
    declareVar(body,"string");
    return "string "+body+";";
  }
  else if(body.contains("Float")){
    body = body.replaceAll("Declare Float ","");
    declareVar(body,"float");
    return "float "+body+";";
  }
  else if(body.contains("Char")){
    body = body.replaceAll("Declare Char ","");
    declareVar(body,"char");
    return "char "+body+";";
  }
  else if(body.contains("Unsigned Char")){
    body = body.replaceAll("Declare Unsigned Char ","");
    declareVar(body,"unsigned char");
    return "unsigned char "+body+";";
  }
  else if(body.contains("Short")){
    body = body.replaceAll("Declare Short ","");
    declareVar(body,"short");
    return "short"+body+";";
  }
  else if(body.contains("Long")){
    body = body.replaceAll("Declare Long ","");
    declareVar(body,"long");
    return "long"+body+";";
  }
  else if(body.contains("Unsigned Long")){
    body = body.replaceAll("Declare Long ","");
    declareVar(body,"unsigned long");
    return "unsigned long"+body+";";
  }
  else if(body.contains("Double")){
    body = body.replaceAll("Declare Double ","");
    declareVar(body,"double");
    return "double"+body+";";
  }
  else if(body.contains("Long Double")){
    body = body.replaceAll("Declare Long Double ","");
    declareVar(body,"long double");
    return "long double"+body+";";
  }
}
else if (body.contains("Input")){
  body = body.replaceAll("Input ","");
```

```
    //if user didn't declare variable
    //the program will use scanf without data format
    if(varMap.get(body)==null)
      return "scanf("+body+");";
    String dataFormat = getDataFormat(varMap.get(body));
    return "scanf(\""+dataFormat+"\",&"+body+");";
  }
  else if (body.contains("Print")){
    body = body.replaceAll("Print ","");
    StringBuilder strCode = new StringBuilder();

    //in case the text has "+" it will
    //split the text to multiple lines of code
    if(body.contains("+")){
      String[] strArray = body.split("\\+");
      for(String str : strArray){
        if(strCode.length()!=0)
          strCode.append("\n"+sps+makePrintf(str));
        else
          strCode.append(makePrintf(str));
      }
    }
    else
      strCode.append(makePrintf(body));
    return strCode.toString();
  }
  return body+";";
}

public static String makePrintf(String body)
//seperated print part from makeCode function
{
  if(body.contains("\"")){
    body = body.replaceAll("\"","");
    return "printf(\""+body+"\");";
  }
  else {
    //if user didn't declare variable
    //the program will use printf without data format
    if(varMap.get(body)==null)
      return "printf("+body+");";
    String dataFormat = getDataFormat(varMap.get(body));
    return "printf(\""+dataFormat+"\","+body+");";
  }
}

public static String getDataFormat(String data)
//gets data format compare with input string
{
  if(data.equals("string"))
    return "%s";
  else if(data.equals("integer"))
    return "%d";
  else if(data.equals("unsigned integer"))
    return "%u";
  else if(data.equals("float"))
    return "%f";
  else if(data.equals("char"))
    return "%c";
  else if(data.equals("unsigned char"))
    return "%c";
  else if(data.equals("short"))
    return "%d";
  else if(data.equals("long"))
    return "%ld";
  else if(data.equals("unsigned long"))
    return "%lu";
  else if(data.equals("double"))
    return "%lf";
  else if(data.equals("unsigned double"))
    return "%Lf";
  else
    return "";
}

public static Graph getPathToTarget(SVGNode source,SVGNode target)
//gets and returns path between starting
//node to the target as a graph
{
  //if the target node is null it will gets the path to "end" node
  HashSet pathSet = new HashSet();
  if(target==null){
    for(SVGNode proc : nodes){
      if(proc instanceof Rect){
        if(((Rect)proc).text.equalsIgnoreCase("end"))
          target = (Rect)proc;
      }
    }
  }

  //gets shortest path between source node and target node
  DijkstraShortestPath<SVGNode, Arrow> dsPath =
   new DijkstraShortestPath<>(diGraph);
  GraphPath<SVGNode, Arrow> shortestPath =
   dsPath.getPath(source,target);
  for(SVGNode proc : shortestPath.getVertexList())
    pathSet.add(proc);
  return new AsSubgraph<SVGNode, Arrow>(diGraph, pathSet);
}

public static Graph getLoopGraph(SVGNode start)
{
  HashSet whileSet = new HashSet();
  List<SVGNode> loopList = null;
  //finds all loops from TarjanSimpleCycles
  //then check with starting vertex
  TarjanSimpleCycles<SVGNode, Arrow> tjCycles =
   new TarjanSimpleCycles<SVGNode, Arrow>(diGraph);
  List<List<SVGNode>> allCycles = tjCycles.findSimpleCycles();
  //find a cycle which match with start node
  for(List<SVGNode> list : allCycles){
    if(!(list.get(0).isShape("condition")))
      Collections.reverse(list);
    if(start==list.get(0)){
      loopList = list;
      break;
    }
  }
  //creates new subgraph which contain this cycle
  if(loopList!=null){
    for(SVGNode proc : loopList){
      if(proc instanceof Rect)
      whileSet.add(proc);
    }
    return new AsSubgraph<SVGNode,Arrow>(diGraph,
     whileSet,diGraph.edgeSet());
  }
  return null;
}

public static class ConvertInfo
//this class made to allow any class to return 2 values at once
{
  //text is needed for convertDiamond to return loop/if-else string
  String text;

  //nextNode is needed for convertDiamond
  //to return next node to process
  SVGNode nextNode;
  public ConvertInfo(String text,SVGNode nextNode)
  {
    this.text = text;
    this.nextNode = nextNode;
  }
}

public static SVGNode findExitCircle(
 Graph tPaths,Graph fPaths,SVGNode start)
//finds both paths and returns a node
//where both paths meet each other
{
  Set<SVGNode> trueNodeSet = tPaths.vertexSet();
  List<SVGNode> tList = new ArrayList(trueNodeSet);
  Collections.reverse(tList);
  Set<SVGNode> falseNodeSet = fPaths.vertexSet();
  for(SVGNode node1 : tList){
    for(SVGNode node2 : falseNodeSet){
      if((node1==node2)&&(node1.type.equals("circle"))){
        return node1;
      }
    }
  }
  return null;
}

public static boolean isLoop(SVGNode node)
{
  /*This can prevert errors in case if-else inside while
    While loop always has 2+ incoming edges
  */
  Set<Arrow> edgesSet = diGraph.incomingEdgesOf(node);
  List<Arrow> edgesList = new ArrayList<>(edgesSet);
  if(edgesSet.size()>1)
    return true;
  else
    return false;
}

public static ConvertInfo convertIf(SVGNode start,String sps)
{
  //gets first node of both side of current node
  SVGNode trueBranch = getBranch(start,true);
  SVGNode falseBranch = getBranch(start,false);
  /*gets true and false paths to the end of diagram
    if there are more paths then calling another convertDiamond
    will deal with more paths
  */
  Graph truePaths = getPathToTarget(trueBranch,null);
  Graph falsePaths = getPathToTarget(falseBranch,null);

  //finds both paths and returns a node
  //where both paths meet each other
  SVGNode nextNode = findExitCircle(truePaths,falsePaths,start);

  //gets paths to that node
  truePaths = getPathToTarget(trueBranch,nextNode);
  falsePaths = getPathToTarget(falseBranch,nextNode);
  String ifString = sps+"if("+((Rect)start).text+"){\n"+
   convertGraph(truePaths,trueBranch,sps+"  ",false)+"\n"+
   sps+"}\n"+sps+"else{\n"+
   convertGraph(falsePaths,falseBranch,sps+"  ",false)+"\n"+sps+"}";
  return new ConvertInfo(ifString,nextNode);
}

public static ConvertInfo convertLoop(SVGNode start,String sps)
{
  //gets first node of both side of current node
  SVGNode trueBranch = getBranch(start,true);
  SVGNode falseBranch = getBranch(start,false);
  SVGNode nextNode = null;
  String loopString="";

  //CycleDetector checks loops in the graph
  CycleDetector cDetector = new CycleDetector(diGraph);

  //checks if there is a loop in any side of the condition
  if(cDetector.detectCyclesContainingVertex(trueBranch)){
    Graph trueLoopBranch = getLoopGraph(start);

    //converts graph to code by calling convertGraph method
    loopString = sps+"while("+((Rect)start).text+"){\n"+
     convertGraph(trueLoopBranch,start,sps+"  ",true)+"\n"+sps+"}";
    nextNode = falseBranch;
  }
  else if(cDetector.detectCyclesContainingVertex(falseBranch)){
    Graph falseLoopBranch = getLoopGraph(start);
    loopString = sps+"while(!"+((Rect)start).text+"){\n"+
     convertGraph(falseLoopBranch,start,sps+"  ",true)+"\n"+sps+"}";
    nextNode = trueBranch;
  }
  return new ConvertInfo(loopString,nextNode);
}

public static ConvertInfo convertDiamond(SVGNode start,String sps)
//converts loop or if-else graph then return text and next node
{
  ConvertInfo info = null;
  if(isLoop(start))
    info = convertLoop(start,sps);
  else
    info = convertIf(start,sps);
  return info;
}

public static String convertGraph(
 Graph graph,SVGNode start,String sps,boolean isLoop)
{
  StringBuilder codeStr = new StringBuilder();

  //returns nothing if graph is null
  if(graph==null)
    return "";
  Set<SVGNode> vertexSet = graph.vertexSet();
  List<SVGNode> vertexList = new ArrayList(vertexSet);

  //finds start node from new graph not global graph
  SVGNode node=null;
  for(SVGNode proc : vertexList){
    if(proc==start)
      node=proc;
  }

  /*getting node from by vertexSet will not sorted correctly
    so using nextVertex from current graph is better
  */
  if(isLoop)
    node=nextVertex(node,graph);
  while(node!=null){
    /*if current vertex is the start vertex it will break or if
      current vertex is another condition it will call convertDiamond
    */
    if(node.isShape("condition")){
      if(node==start&&isLoop)
        break;
      ConvertInfo info = convertDiamond(node,sps);
      if(codeStr.length()!=0)
        codeStr.append("\n"+info.text);
      else
        codeStr.append(info.text);
      node=info.nextNode;
    }
    //if current vertex is one of Rect class it will append text
    else if(node instanceof Rect){
      if(codeStr.length()!=0)
        codeStr.append("\n"+sps+makeCode(((Rect)node).text,sps));
      else
        codeStr.append(sps+makeCode(((Rect)node).text,sps));
      node=nextVertex(node,graph);
    }
    //deal with other classes which have no text inside
    else {
      if(!isLoop){

        break;
      }
      else{

        node=nextVertex(node,diGraph);
        if(node!=null)
        System.out.println(((Rect)node).text);
      }
    }

  }
  return codeStr.toString();
}

public static String convertDiagram(SVGNode vertex)
//main function to convert each node of the diagram
{
  StringBuilder codeStr = new StringBuilder();
  while(vertex!=null){
    if(vertex.isShape("condition")){
      ConvertInfo info = convertDiamond(vertex,"");
      codeStr.append("\n"+info.text);
      vertex = info.nextNode;
    }

    /*if it is not condition it will add code
      Rect included with condition,terminal,process,data
    */
    else if(vertex instanceof Rect){
      codeStr.append("\n"+makeCode(((Rect)vertex).text,""));
      vertex = nextVertex(vertex,diGraph);
    }
    else
      vertex = nextVertex(vertex,diGraph);
  }
  return codeStr.toString();
}

public static SVGNode getBranch(SVGNode vertex,boolean edge)
{
  //gets branch of any true or false of vertex
  Set<Arrow> edgesSet = diGraph.outgoingEdgesOf(vertex);
  List<Arrow> edgesList = new ArrayList<>(edgesSet);
  for(Arrow arrow : edgesList){
    if(arrow.condition.equalsIgnoreCase("true")&&(edge))
      return diGraph.getEdgeTarget(arrow);
    else if(arrow.condition.equalsIgnoreCase("false")&&(!edge))
      return diGraph.getEdgeTarget(arrow);
  }
  return null;
}

public static SVGNode nextVertex(SVGNode vertex,Graph graph)
{
  //gets next vertex of current vertex
  if(vertex==null)
    return null;
  Set<Arrow> edgesSet = null;
  try{
    edgesSet = graph.outgoingEdgesOf(vertex);
  } catch(Exception e){
    return null;
  }
  List<Arrow> edgesList = new ArrayList<>(edgesSet);
  if(edgesList.size()!=0)
    return (SVGNode)graph.getEdgeTarget(edgesList.get(0));
  return null;
}

public static float getPresValue(SVGElement element,String att)
{
  return Float.parseFloat(
   element.getPresAbsolute(att).getStringValue());
}

public static boolean containsIgnoreCase(String str1,String str2)
{
  if(str1.toLowerCase().contains(str2.toLowerCase()))
    return true;
  else
    return false;
}

public static void compareNode(int objCount)
{
  for(SVGNode proc : nodes){
    if(proc instanceof Arrow){
      Arrow arrow = (Arrow)proc;
      SVGNode connection1 =
       printConnect(arrow.point.getP1(),"Arrow");
      SVGNode connection2 =
       printConnect(arrow.point.getP2(),"Arrow");
      if(connection1!=null&&connection2!=null){
        diGraph.addVertex(connection1);
        diGraph.addVertex(connection2);
        diGraph.addEdge(connection1,connection2,arrow);
      }
    }
  }

  for(SVGNode proc : nodes){
    if(proc instanceof Text){
      Text text = (Text)proc;
      if(containsIgnoreCase(text.lbText,"true")||
       containsIgnoreCase(text.lbText,"false")){
        SVGNode textTemp = printConnect(text.point,text.lbText);
        ((Arrow)textTemp).condition = text.lbText;
      }
    }
  }
}
public static SVGNode printConnect(Point2D point,String checkType){
  String txt = "";
  double distanceRect=1000,distanceArrow=1000;
  SVGNode nearestRect = null;
  Arrow nearestArrow = null;
  for(SVGNode proc : nodes){
    if(proc instanceof Rect){
      Point2D.Float centerPoint =
       new Point2D.Float((float)((Rect)proc).rect.
       getCenterX(),(float)((Rect)proc).rect.getCenterY());
      if(point.distance(centerPoint)<distanceRect){
        nearestRect = proc;
        distanceRect = point.distance(centerPoint);
      }
    }
    if(proc instanceof Circle){
      if(point.distance(((Circle)proc).point)<distanceRect){
        nearestRect = proc;
        distanceRect = point.distance(((Circle)proc).point);
      }
    }
    if(proc instanceof Arrow){
      if(((Arrow)proc).point.ptSegDist(point)<distanceArrow){
        nearestArrow = ((Arrow)proc);
        distanceArrow = ((Arrow)proc).point.ptSegDist(point);
      }
    }
  }
  if(checkType.equalsIgnoreCase("true")||
   checkType.equalsIgnoreCase("false"))
    return nearestArrow;
  for(SVGNode temp : nodes){
    if(temp instanceof Text){
      Text text = (Text)temp;
      if(nearestRect instanceof Rect){
        if(((Rect)nearestRect).checkConnect(text.point)){
          ((Rect)nearestRect).text = text.lbText;
          txt = text.lbText;
        }
      }
    }
  }
  return nearestRect;
}
public static void printNode(){
  for(SVGNode proc : nodes){
    if(proc instanceof Rect)
      ((Rect)proc).printAll();
    else if(proc instanceof Text)
      ((Text)proc).printAll();
    else if(proc instanceof Arrow)
      ((Arrow)proc).printAll();
    else if(proc instanceof Circle){
      ((Circle)proc).printAll();
    }
  }
}
public static int findElem(SVGElement elem,int objCount){
  objCount++;
  for(int i = 0; i < elem.getNumChildren(); i++) {
    storeNode(elem,objCount,i);
    SVGElement element  = elem.getChild(i);
    objCount = findElem(element, objCount);
  }
  return objCount;
}
public static void storeNode(SVGElement elem,int i,int objCount){
  float[] tmpx = new float[4];
  float[] tmpy = new float[4];
  String tText = "";
  String type = "";
  String att[] = {"x","y","width","height","rx","ry","cx","cy",
   "transform","xml:space","sodipodi:role"};
  AffineTransform matrix = new AffineTransform(0,0,0,0,0,0);
  SVGElement element = elem.getChild(objCount);
  try{
    for(int j=0;j<att.length;j++){
      if(element.hasAttribute(att[j],AnimationElement.AT_AUTO)){
        if(att[j]=="x"){
          tmpx[0] = getPresValue(element,att[j]);
          type = "process";
        }
        if(att[j]=="y"){
          tmpy[0] = getPresValue(element,att[j]);
          type = "process";
        }
        if(att[j]=="width")
          tmpx[1] = getPresValue(element,att[j]);
        if(att[j]=="height")
          tmpy[1] = getPresValue(element,att[j]);
        if(att[j]=="rx"){
          tmpx[2] = getPresValue(element,att[j]);
          type = "terminal";
        }
        if(att[j]=="ry"){
          tmpy[2] = getPresValue(element,att[j]);
          type = "terminal";
        }
        if(att[j]=="cx"){
            tmpx[0] = getPresValue(element,att[j]);
            type= "circle";
          }
          if(att[j]=="cy"){
            tmpy[0] = getPresValue(element,att[j]);
            type = "circle";
          }
          if(att[j]=="transform"){
            String trans = element.getPresAbsolute(
             att[j]).getStringValue();
            matrix = element.parseSingleTransform(trans);
            type = "data";
          }
          if(att[j]=="xml:space"){
            for(int tcount=0;tcount<element.getNumChildren();tcount++)
{
              Tspan text = (Tspan) (element.getChild(tcount));
              tText = tText + text.getText();
            }
            type = "text";
          }
          if(att[j]=="sodipodi:role")
            type = "tspan";
        }
      }
    }
    catch(SVGException e) { System.out.println(e);  }
    try{
      if(element.hasAttribute("d",AnimationElement.AT_AUTO)){
       String[] info=element.getPresAbsolute("d").
       getStringValue().split(" ");
        for(int z=0; z<info.length;z++){
          if(!(info[0].contains("M")||info[z].contains("m")))
            type = "circle";
          if(info[z].contains("M")||info[z].contains("m")){
            if(z!=0){
              String s[] = info[z-1].split(",");
              if(info[0].contains("m")){
                tmpx[0] = Float.parseFloat(s[0])+tmpx[1];
                tmpy[0] = Float.parseFloat(s[1])+tmpy[1];
              }
              else {
                tmpx[0] = Float.parseFloat(s[0]);
                tmpy[0] = Float.parseFloat(s[1]);
              }
              type = "arrow";
              break;
            }
            String s[] = info[z+1].split(",");
            tmpx[1] = Float.parseFloat(s[0]);
            tmpy[1] = Float.parseFloat(s[1]);
            z++;
          }
          else if(info[z].contains("V")||info[z].contains("v")){
            tmpx[0] = tmpx[1];
            if(info[z].contains("V"))
              tmpy[0] = Float.parseFloat(info[z+1]);
            if(info[z].contains("v"))
              tmpy[0] = tmpy[1]+Float.parseFloat(info[z+1]);
            type = "arrow";
            break;
          }
          else if(info[z].contains("C")||info[z].contains("c"))
            z+=2;
          else if(info[z].contains("H"))
            System.out.println("\tX-Destination point : -"+info[z+1]);
          else if(info[z].contains("h"))
            System.out.println("\tX-Destination point : "+info[z+1]);
          else if(info[z].contains("l")||info[z].contains("L")){
            System.out.println("\tCondition point : "+info[z+1]);
            type = "condition";
          }
          else if(info[z].contains("Z")||info[z].contains("z")){
            for(int j=1;j<5;j++){
                String s[] = info[j].split(",");
                tmpx[j-1] = Float.parseFloat(s[0]);
                tmpy[j-1] = Float.parseFloat(s[1]);
            }
            type = "condition";
            break;
          }
        }
      }
      else if(!element.hasAttribute("d",AnimationElement.AT_AUTO))
      {
        if(element.getId().contains("path"))
          type="circle";
      }
    }
    catch(SVGException e) { System.out.println(e);  }
    tmpx[0] = (float)((int)( tmpx[0] *100f))/100f;
    tmpy[0] = (float)((int)( tmpy[0] *100f))/100f;
    tmpx[1] = (float)((int)( tmpx[1] *100f))/100f;
    tmpy[1] = (float)((int)( tmpy[1] *100f))/100f;
    tmpx[2] = (float)((int)( tmpx[2] *100f))/100f;
    tmpy[2] = (float)((int)( tmpy[2] *100f))/100f;
    tmpx[3] = (float)((int)( tmpx[3] *100f))/100f;
    tmpy[3] = (float)((int)( tmpy[3] *100f))/100f;
    if(type == "circle"){
      Point2D.Float tempPoint = new Point2D.Float(tmpx[0],tmpy[0]);
      SVGNode temp =
       new Circle(element.getId(),"circle",tmpx[0],tmpy[0]);
      nodes.add(temp);
    }
    else if(type == "process"){
      Rectangle2D.Float tempRect =
       new Rectangle2D.Float(tmpx[0],tmpy[0],tmpx[1],tmpy[1]);
      SVGNode temp = new Rect(element.getId(),"process",tempRect);
      nodes.add(temp);
    }
    else if (type == "terminal"){
      Rectangle2D.Float tempRect =
     new Rectangle2D.Float(tmpx[0],tmpy[0],tmpx[1],tmpy[1]);
    SVGNode temp = new Rect(element.getId(),"terminal",tempRect);
    nodes.add(temp);
  }
  else if (type == "data"){
    Rectangle2D.Float tempRect =
     new Rectangle2D.Float(tmpx[0],tmpy[0],tmpx[1],tmpy[1]);
    Rectangle2D temp2D =
     matrix.createTransformedShape(tempRect).getBounds2D();
    tempRect = new Rectangle2D.Float((float)temp2D.getX(),
     (float)temp2D.getY(),(float)temp2D.
     getWidth(),(float)temp2D.getHeight());
    SVGNode temp = new Rect(element.getId(),"data",tempRect);
    nodes.add(temp);
  }
  else if (type == "condition"){
    Rectangle2D.Float tempRect = null;
    if(tmpy[1]==tmpy[2])
      tempRect = new Rectangle2D.Float(tmpx[0]-tmpx[1],
       tmpy[0],tmpx[1]*2,tmpy[1]*2);
    else {
      tempRect = new Rectangle2D.Float(tmpx[0],tmpy[0],-1,-1);
      for (int k = 0; k < 4; k++) {
        tempRect.add(tmpx[k],tmpy[k]);
      }
    }
    SVGNode temp = new Rect(element.getId(),"condition",tempRect);
    nodes.add(temp);
  }
  else if (type == "arrow"){
    SVGNode temp = new Arrow(element.getId(),"arrow",tmpx[0],
     tmpy[0],tmpx[1],tmpy[1]);
    nodes.add(temp);
  }
  else if (type == "text"){
    Point2D.Float tempPoint = new Point2D.Float(tmpx[0],tmpy[0]);
    SVGNode temp = null;
    if((float)matrix.getScaleY()!=0)
      temp = new Text(element.getId(),"text",tText,tmpx[0],
       tmpy[0]*(float)matrix.getScaleY());
    else
      temp = new Text(element.getId(),"text",tText,tmpx[0],tmpy[0]);
    nodes.add(temp);
  }

}
```

Appendix C
SVGNode class in Extractor.java

```
public static class SVGNode
{
  String id;
  String type;
  public SVGNode(String id,String type)
  {
    this.id = id;
    this.type = type;
  }
  public boolean isShape(String shape)
  {
    if(type.contains(shape))
      return true;
    return false;
  }
}
public static class Circle extends SVGNode
{
  Point2D.Float point;
  public Circle(String id,String type,float iX,float iY)
  {
    super(id,type);
    this.point = new Point2D.Float(iX,iY);
  }
  public void printAll()
  {
    System.out.println("\nID :"+id);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    System.out.println("\tPosition(X,Y) : "+point.toString());
  }
}
public static class Rect extends SVGNode
{
  Rectangle2D.Float rect;
  String text="";
  public Rect(String id,String type,Rectangle2D.Float rect)
  {
    super(id,type);
    this.rect=rect;
  }
  public void printAll()
  {
    System.out.println("\nID :"+id);
    System.out.println("\ttype :"+type);
    System.out.println("\tPosition(X,Y) : "+rect.toString());
  }
  public boolean checkConnect(Point2D dest)
  {
    return rect.contains(dest);
  }
}
public static class Arrow extends SVGNode implements Serializable,
Cloneable
 {
  Line2D.Float point;
  String condition="";
  public Arrow(String id,String type,float startX,float startY,float
   destX,float destY)
  {
    super(id,type);
    startX = (float)((int)( startX *100f))/100f;
    startY = (float)((int)( startY *100f))/100f;
    destX = (float)((int)( destX *100f))/100f;
    destY = (float)((int)( destY *100f))/100f;
    this.point = new Line2D.Float(startX,startY,destX,destY);
  }
  public void printAll()
  {
    System.out.println("\nID :"+id);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    System.out.println("\tStarting point : "+point.getP1());
    System.out.println("\tDestination point : "+point.getP2());
  }
}
public static class Text extends SVGNode
{
  String lbText;
  Point2D.Float point;
  public Text(String id,String type,String lbText,float iX,float iY)
  {
    super(id,type);
    this.lbText = lbText;
    this.point = new Point2D.Float(iX,iY);
  }
  public void printAll()
  {
    System.out.println("\nID :"+id);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    System.out.println("\tPosition(X,Y) : "+point.toString());
    System.out.println("\tText : "+lbText);
  }
}
}
```

Appendix D

My program NSExtractor.java

```
import com.kitfox.svg.***;
import java.util.Scanner;
import java.io.*;
import java.net.URI;
import com.kitfox.svg.animation.AnimationElement;
import java.util.ArrayList;

//NSExtractor.java
public class NSExtractor
{
  static ArrayList<SVGNode> nodes = new ArrayList<SVGNode>();
  public static void main(String[] args){

    char cChecker = '1';
    URI uri = null;
    try {
      File f = new File(args[0]);
      uri = f.toURI();
    }
    catch (Exception e)
      System.out.println(e);

    SVGUniverse svgUniverse = new SVGUniverse();
    SVGDiagram diagram = svgUniverse.getDiagram(uri);


    int objCount = 0;
    for(int i = 0; i < diagram.getRoot().getNumChildren(); i++) {

      SVGElement elem  = diagram.getRoot().getChild(i);
      objCount = findElem(elem,objCount);
    }
    printNode();
  }
  public static void printNode()
  {
    for(SVGNode proc : nodes){
      if(proc instanceof Rectangle){
        Rectangle aaa = (Rectangle)proc;
        aaa.printAll();
      }
      else if(proc instanceof Triangle){
        Triangle aaa = (Triangle)proc;
        aaa.printAll();
      }
      else if(proc instanceof Text){
        Text aaa = (Text)proc;
        aaa.printAll();
      }
    }
  }
public static int findElem(SVGElement elem,int objCount)
{
  objCount++;
  for(int i = 0; i < elem.getNumChildren(); i++) {
    storeNode(elem,objCount,i);
    SVGElement element  = elem.getChild(i);
    objCount = findElem(element, objCount);
  }
  return objCount;
}
public static void storeNode(SVGElement elem,int i,int objCount)
{
  float[] tmpx = new float[3];
  float[] tmpy = new float[3];
  String tText = "";
  String sType = "";
  String pres[]= {"x","y","width","height","cy","cx", "rx", "ry"};
  SVGElement element = elem.getChild(objCount);

  try{
    if(element.hasAttribute("x",AnimationElement.AT_AUTO)){
      tmpx[0] = Float.parseFloat(
       element.getPresAbsolute("x").getStringValue());
      sType = "rectangle";
    }
    if(element.hasAttribute("y",AnimationElement.AT_AUTO)){
      tmpy[0] = Float.parseFloat(
       element.getPresAbsolute("y").getStringValue());
      sType = "rectangle";
    }
    if(element.hasAttribute("width",AnimationElement.AT_AUTO))
      tmpx[1] = Float.parseFloat(
       element.getPresAbsolute("width").getStringValue());
    if(element.hasAttribute("height",AnimationElement.AT_AUTO))
      tmpy[1] = Float.parseFloat(
       element.getPresAbsolute("height").getStringValue());
    if(element.hasAttribute("xml:space",AnimationElement.AT_AUTO)){
      Tspan text = (Tspan) (element.getChild(0));
      tText = text.getText();
      sType = "text";
    }
if(element.hasAttribute("sodipodi:role",AnimationElement.AT_AUTO))
        sType = "tspan";
    }
    catch(SVGException e)
      System.out.println(e);

    try{
      if(element.hasAttribute("d",AnimationElement.AT_AUTO)){
        String[] info = element.getPresAbsolute(
         "d").getStringValue().split(" ");
        for(int z=0; z<info.length;z++){
          if(info[z].contains("M")||info[z].contains("m")){
            if(z!=0){
              String s[] = info[z-1].split(",");
              if(info[0].contains("m")){
                tmpx[0] = Float.parseFloat(s[0])+tmpx[1];
                tmpy[0] = Float.parseFloat(s[1])+tmpy[1];
              } else {
                tmpx[0] = Float.parseFloat(s[0]);
                tmpy[0] = Float.parseFloat(s[1]);
              }
              sType = "arrow";
              break;
            }
            String s[] = info[z+1].split(",");
            tmpx[1] = Float.parseFloat(s[0]);
            tmpy[1] = Float.parseFloat(s[1]);
            z++;
          }
          else if(info[z].contains("V")||info[z].contains("v")){
            tmpx[0] = tmpx[1];
            tmpy[0] = Float.parseFloat(info[z+1]);
            sType = "triangle";
            z++;
          }
          else if(info[z].contains("C")||info[z].contains("c"))
            z+=2;
          else if(info[z].contains("H")){
            String s[] = info[z+1].split(",");
            tmpx[2] = Float.parseFloat(s[0]);
            tmpy[2] = tmpy[0];
            break;
          }
          else if(info[z].contains("h"))
            System.out.println("\tX-Destination point : "+info[z+1]);
          else if(info[z].contains("l")||info[z].contains("L")){
            System.out.println("\tCondition point : "+info[z+1]);
            sType = "diamond";
          }
          else {
            String s[] = info[z].split(",");
            tmpx[0] = Float.parseFloat(s[0]);
            tmpy[0] = Float.parseFloat(s[1]);
            sType = "triangle";
          }
        }
      }
    }
```

Appendix E
SVGNode class in NSExtractor.java

```
public static class SVGNode
{
  String iId;
  public SVGNode(String iId)
  {
    this.iId = iId;
  }
}
public static class Rectangle extends SVGNode
{
  float iX;
  float iY;
  float iWidth;
  float iHeight;
  public Rectangle(String iId,float iX,float iY,float
   iWidth,float iHeight)
  {
    super(iId);
    this.iY = iY;
    this.iX = iX;
    this.iWidth = iWidth;
    this.iHeight = iHeight;
  }
  public void printAll()
  {
    System.out.println("\nID :"+iId);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    System.out.println("\tX : "+iX+" Y : "+iY);
    System.out.println("\tWidth : "+iWidth+" Height : "+iHeight);
  }
}
public static class Triangle extends SVGNode
{
  float[] iAngleX = new float[4];
  float[] iAngleY = new float[4];;
  public Triangle(String iId,float iAngleX[],float iAngleY[])
  {
    super(iId);
    this.iAngleX = iAngleX.clone();
    this.iAngleY = iAngleY.clone();
  }
  public void printAll()
  {
    System.out.println("\nID :"+iId);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    for(int i=0;i<3;i++){
      System.out.println("\tNode#"+i+" X : "+iAngleX[i]+" Y :
       "+iAngleY[i]);
    }
  }
}
public static class Text extends SVGNode
{
  String sText;
  float iX;
  float iY;
  public Text(String iId,String sText,float iX,float iY)
  {
    super(iId);
    this.sText = sText;
    this.iX = iX;
    this.iY = iY;
  }
  public void printAll()
  {
    System.out.println("\nID :"+iId);
    System.out.println("\ttype :"+this.getClass().getSimpleName());
    System.out.println("\tX : "+iX+" Y : "+iY);
    System.out.println("\tText : "+sText);
  }
}
```