Java implementation for Mason's Method to compute overall gain of a control system

Programming Assignment 1

Dahlia Chehata Mahmoud   ID: 28

I. **Problem Statement:**

It is required to implement Mason's Method to compute the overall gain of a given control system by means of Signal Flow Graph SFG.

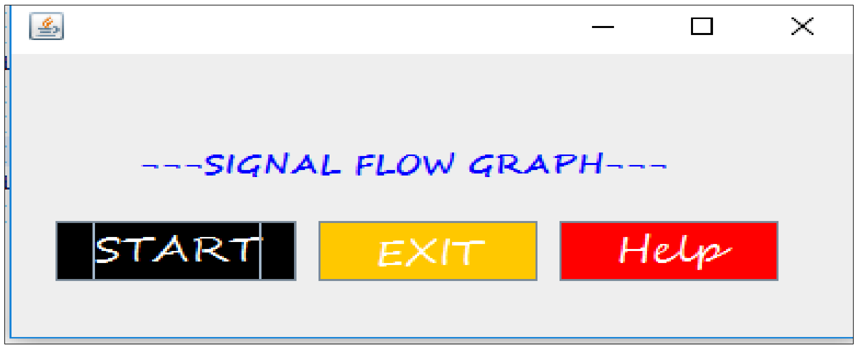
Requirements:

1- Graphical interface.
2- Draw the signal flow graph showing nodes, branches, gains, …
3- Listing all forward paths, individual loops, all combination of *n* non-touching loops.
4- The values of Δ1, Δ2, ...Δ*m* where *m* is number of forward paths.
5- Overall system transfer function.

II. **Main Features of the program:**

The program supports GUI with a group of buttons to add nodes, edges, compute the overall gain, several tables demonstrating forward paths, individual loops, all possible combinations of non-touching loops as well as their correspondent gains

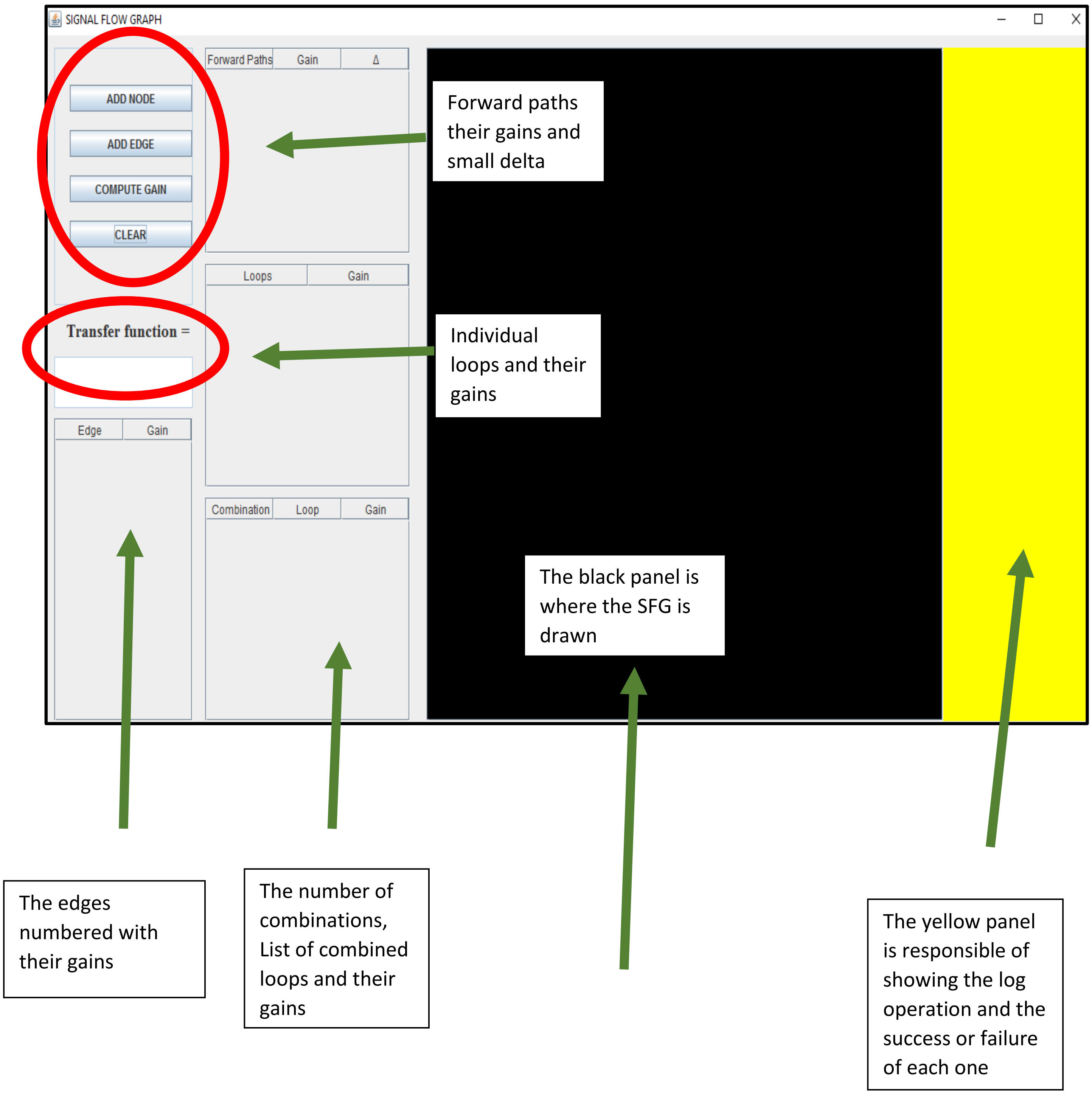
By clicking the exe:



There are 3 options:

- to start a new SFG,
- to exit the window
- to open the user's guide

By clicking the start Button:
This window appears:



Forward paths their gains and small delta

Individual loops and their gains

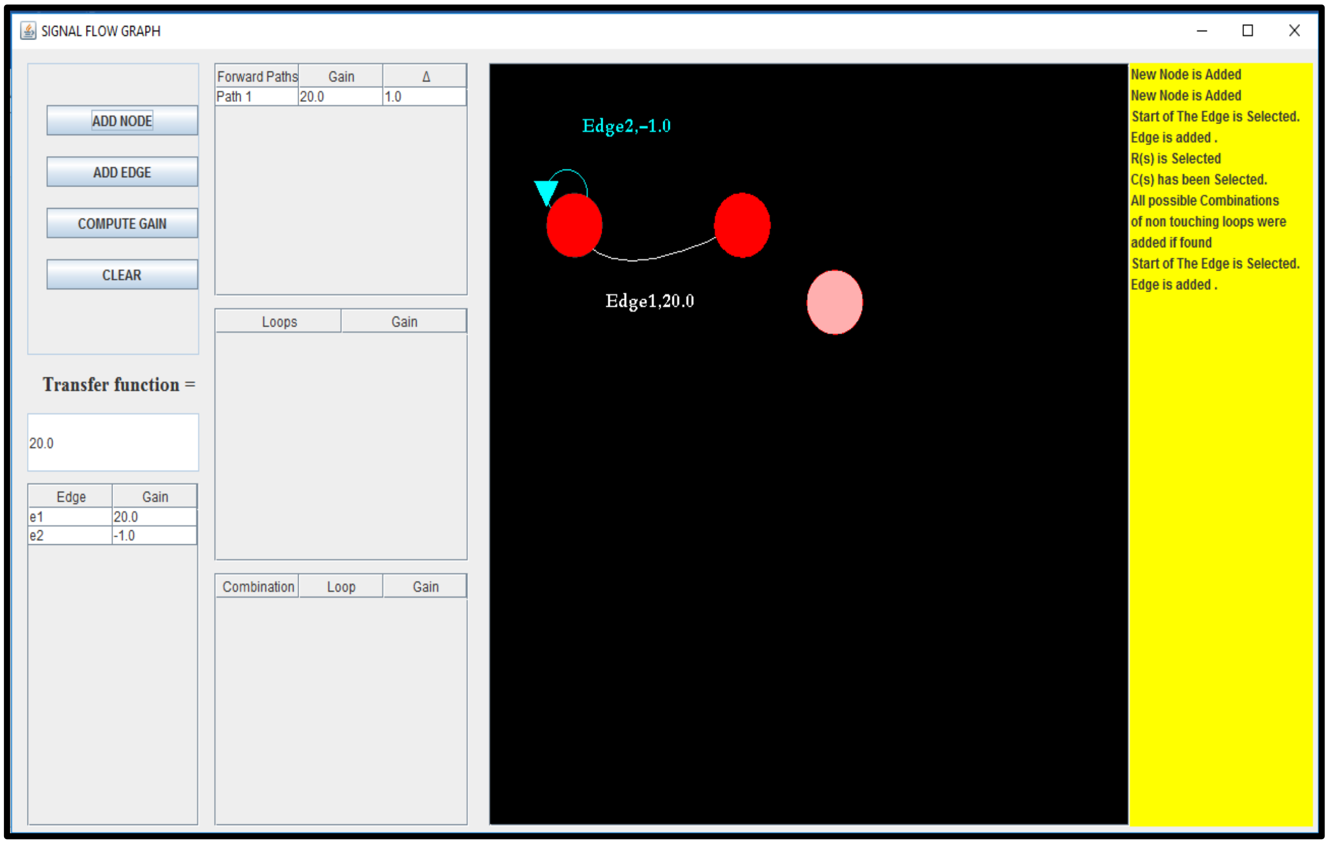
The black panel is where the SFG is drawn

The edges numbered with their gains

The number of combinations, List of combined loops and their gains

The yellow panel is responsible of showing the log operation and the success or failure of each one

III. Additional options:



Forward Paths	Gain	Δ
Path 1	20.0	1.0

Loops	Gain

Combination	Loop	Gain

Transfer function =

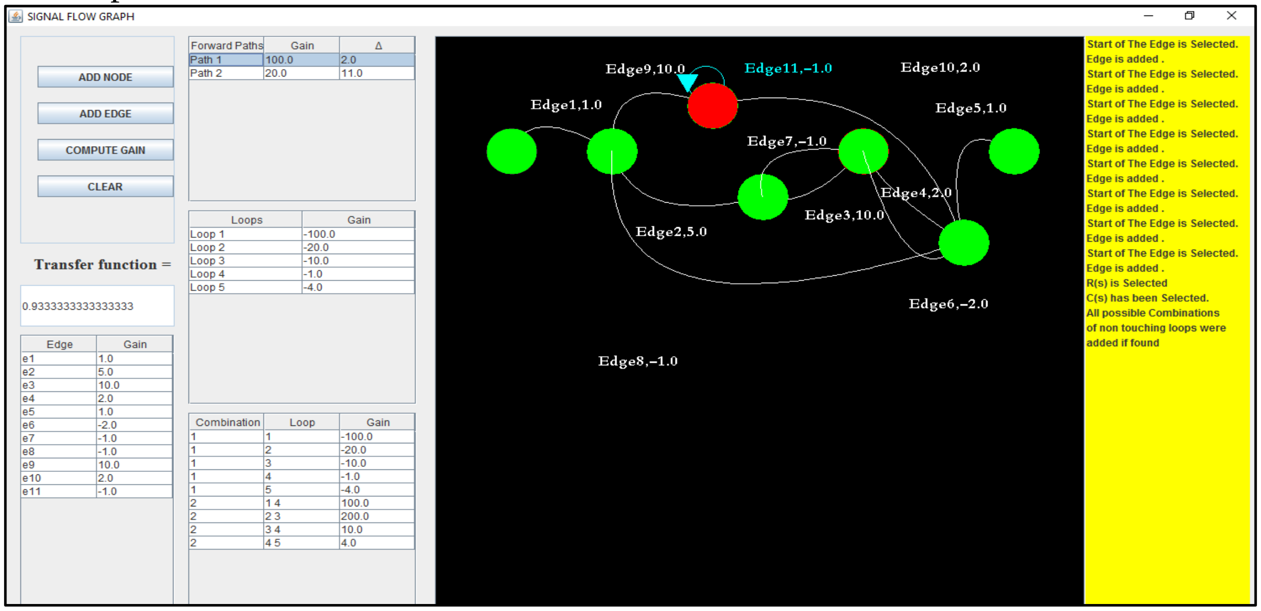
20.0

Edge	Gain
e1	20.0
e2	-1.0

```
New Node is Added
New Node is Added
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
Start of The Edge is Selected.
Edge is added .
```

- The selection of a node sets the color node to pink, once the node is added and is not selected, its color is set to red
- Self-loops are marked with cyan color with arrows
- The yellow panel is for log operations: it states the result of its operation: success or failure so the user is supplied with messages constantly
- The clear button CLEAR the drawing black panel, the log panel, tables and the transfer function result at any time, to start a new SFG
- The number of nodes is not necessarily given at the beginning of the program as it is required. This additional feature makes it easier to draw any number of nodes at any time and the mouse click event in java will simulate its motion on the panel to data
- The edge index is written on each edge as well as it gain
- The down corner left panel contains the index of each edge and the correspondent gain
- The user can modify the value on the edge by clicking on it in the table, add the new value in the box and press enter
- The EXIT option in the first panel closes the window
- The HELP button contains a manual to help the user to deal with the program

- By clicking any row in the tables (forward paths, loops or combinations) the color of the nodes in the chosen path are set to green as showing in the sample runs section below



Transfer function =

0.9333333333333333

Edge	Gain
e1	1.0
e2	5.0
e3	10.0
e4	2.0
e5	1.0
e6	-2.0
e7	-1.0
e8	-1.0
e9	10.0
e10	2.0
e11	-1.0

Forward Paths	Gain	Δ
Path 1	100.0	2.0
Path 2	20.0	11.0

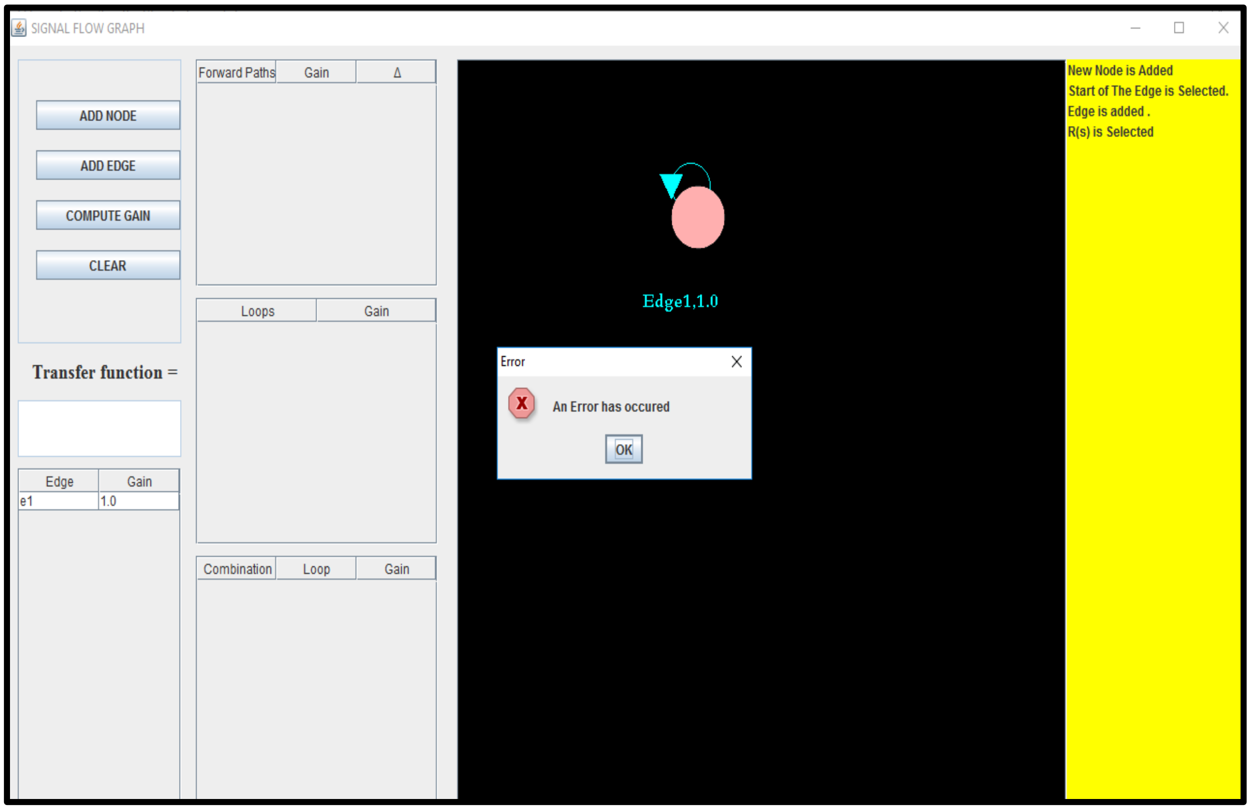
Loops	Gain
Loop 1	-100.0
Loop 2	-20.0
Loop 3	-10.0
Loop 4	-1.0
Loop 5	-4.0

Combination	Loop	Gain
1	1	-100.0
1	2	-20.0
1	3	-10.0
1	4	-1.0
1	5	-4.0
2	1 4	100.0
2	2 3	200.0
2	3 4	10.0
2	4 5	4.0

```
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
```

- The program can handle several errors and shows error messages.

example: compute gain on single node



Transfer function =

Edge	Gain
e1	1.0

Forward Paths	Gain	Δ

Loops	Gain

Combination	Loop	Gain

```
An Error has occured
```

```
New Node is Added
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
```

IV. Data Structure:

- The forward paths and loops were extracted using a **modified Depth First search DFS method**. this method explores each path from a starting node to the end, then mark the nodes as visited for backtracking. In case of loops the source and sink node are one, so the method takes the same parameter twice.
- The combinations of non-touching loops were added in
- `private ArrayList<ArrayList<int[]>> NonTouchingLoopsCombinations;`

the Backend:

- The class Node implements the interface INode , it contains the node data from its name ,number ,the list of its edges
- The class Edge implements IEdge interface,it contains the edge info,its starting and end node ,the gain value
- The class Path implementing IPath interface, contains a linked list of nodes

  Its main methods are the path gain which compute the gain of thenodes of the list and the bitmasking method used in extracting the different combinations f loops
- The class MasonMethod implementing IMason,is the main class responsible for the signal flow graph calculations. It contains the DFS Method used in GetForwardPaths() and GetIndividualLoops() methods.

  Also contains these methods : calculateBigDelta(),allLoopsCombinations(), nonTouchingLoopsCombinations();calculateForwardPathDelta();

example:

for computing the transfer function:

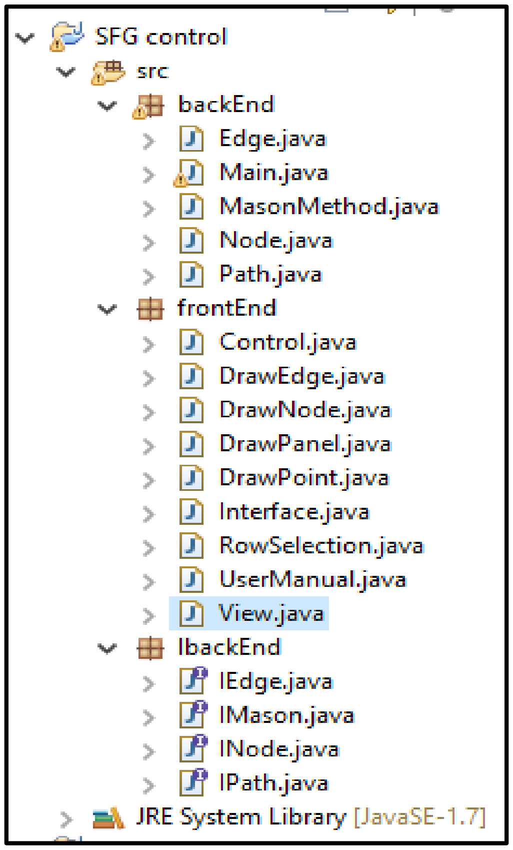
```
@ Override
  public double OverallTransferFunction(INode source,INode sink){
        getForwardPaths(source,sink);
        getIndividualLoops();
        NonTouchingLoopsCombinations = new ArrayList<>();
            forwardPathsDeltas = new ArrayList<>();
            double denominator=calculateBigDelta();
          double numenator=0;
          for (int i=0;i<forwardPaths.size();i++)
            numenator+=forwardPaths.get(i).getPathGain()
            *calculateForwardPathDelta(forwardPaths.get(i));
         return numenator/denominator;

   }
```

The Frontend:

- Is mainly responsible for the graphical user interface.
- It follows The MVC design pattern
- The control class links between the view class(GUI) and the backend(Model)
- The GUI is done using javax.swing
- DrawEdge class and DrawNode class are subclasses of Node and Edge classes in the backend.these classes contains the x and y coordinates data of nodes and edges as well as the height ,width. They contain also the main method draw() from `java.awt.Graphics`
- The View class follows the groupLayout in Swing that gives a sequential order of panels and tables as shown in the screenshot

V. Main modules:

1.Hierarchy:



```
SFG control
  src
    backEnd
      Edge.java
      Main.java
      MasonMethod.java
      Node.java
      Path.java
    frontEnd
      Control.java
      DrawEdge.java
      DrawNode.java
      DrawPanel.java
      DrawPoint.java
      Interface.java
      RowSelection.java
      UserManual.java
      View.java
    IbackEnd
      IEdge.java
      IMason.java
      INode.java
      IPath.java
  JRE System Library [JavaSE-1.7]
```

2.Packages:

The code is written in java in 2 main packages:
frontEnd : responsible for the graphical user interface and graphics to draw the nodes ,edges, self-loops, it places the forward paths and their gains as well as the loops gains in table and supports control buttons for addition and modification of the SFG
backEnd: gathers data from nodes, branches and different paths, identifies forward paths, individual loops, all possible combinations of non-touching loops and compute their gains using Mason's Formula

The package **IbackEnd** contains the interfaces of the backend package's classes To allow contact with the frontEnd's classes.

2.Classes: *as seen in the hierarchy*

Backend:

- Node class implements INode interface
- DrawNode class extends Node
- Egde Class implements IEdge interface
- DrawEdge extends Edge
- MasonMethod Class implements IMason interface
- Path Class implements IPath

For further details please refer to the data structure section

Frontend:

- The control class links between the view class(GUI) and the backend(Model)
- DrawEdge class and DrawNode class are subclasses of Node and Edge classes in the backend.these classes contains the x and y coordinates data of nodes and edges as well as the height ,width. They contain also the main method draw() from `java.awt.Graphics`
- The View class follows the groupLayout in Swing that gives a sequential order of panels and tables as shown in the screenshot
- Interface class is responsible for the façade window with options: start ,exit and help
- User Manual class is responsible for the panel that shows when HELP is pressed

VI. Algorithms used:

- **Refer to the data structure and Main Modules sections for additional info**
- The modified DFS method used to get forward paths and loops ,is the same as the original but when the path's end node is sink node, the algorithm begins the backtracking procedure

Example for the modified DFS method:

```
private void DFS(INode node, INode srcNode, INode endNode, LinkedList<IPath>
paths,
                  INode[] nodes, double gain) {
            visited[node.getnum()] = true;
            for (int i = 0; i < nodes[node.getnum()].getEdges().size(); i++)
{
                  if
(nodes[node.getnum()].getEdges().get(i).getEndNode().getnum() ==
endNode.getnum()) {

      nodes[node.getnum()].getEdges().get(i).getEndNode().setParent(node);
                        series = new LinkedList<>();
                        series.add(getPath(srcNode, node));
                        series.add(endNode);
                        Path path = new Path(series, gain *
nodes[node.getnum()].getEdges().get(i).getGainValue());
                        paths.add(path);
                  } else if
(!visited[nodes[node.getnum()].getEdges().get(i).getEndNode().getnum()]) {

      nodes[node.getnum()].getEdges().get(i).getEndNode().setParent(node);
                        gain *=
nodes[node.getnum()].getEdges().get(i).getGainValue();

      DFS(nodes[node.getnum()].getEdges().get(i).getEndNode(), srcNode,
endNode, paths, nodes, gain);
                        gain /=
nodes[node.getnum()].getEdges().get(i).getGainValue(); // backtracking
                  }
            }
            visited[node.getnum()] = false;// backtracking
      }
```

- Algorithm used to calculate forward paths gains depends mainly on 3 nested loops:

```
private double calculateForwardPathDelta(IPath forwardPath) {
            double result = 1;
            for (int CombinationIndex = 0; CombinationIndex <
NonTouchingLoopsCombinations.size();
                        CombinationIndex++) {
                  ArrayList<int[]> CombinedLoops =
NonTouchingLoopsCombinations.get(CombinationIndex);
                  double sumOfProducts = 0;
                  for (int i = 0; i < CombinedLoops.size(); i++) {
                        double Product = 1;
                        for (int j = 0; j < CombinedLoops.get(i).length;
j++) {
                              if
(!isPathTouchingWithLoop(forwardPath,loops.get(CombinedLoops.get(i)[j]))) {
                                    Product *=
loops.get(CombinedLoops.get(i)[j]).getPathGain();
                              } else {
                                    Product=0;
                                    break;
                              }
                        }
                        sumOfProducts += Product;
                  }
                  if (CombinationIndex % 2 == 0)result -= sumOfProducts;
                  else result += sumOfProducts;
            }
            forwardPathsDeltas.add(result);
            return result;
      }
```

- Bit Masking Method used in getting the loops combinations:

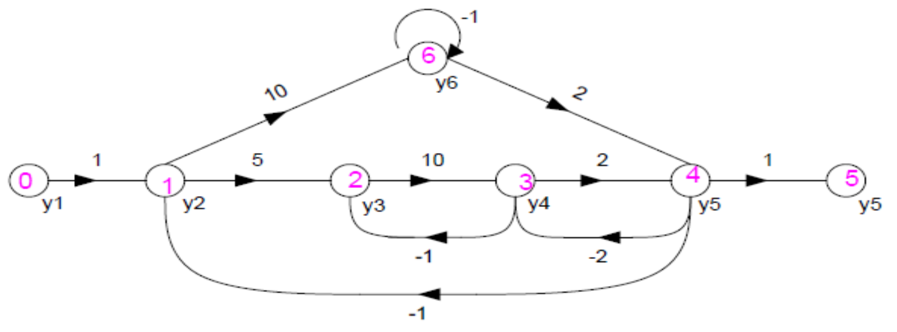
```
public int bitMasking() {
            int Mask=0;
            for (int i=0;i<path.size();i++)
                  Mask|=(1<<path.get(i).getnum());
            return Mask;
      }
```

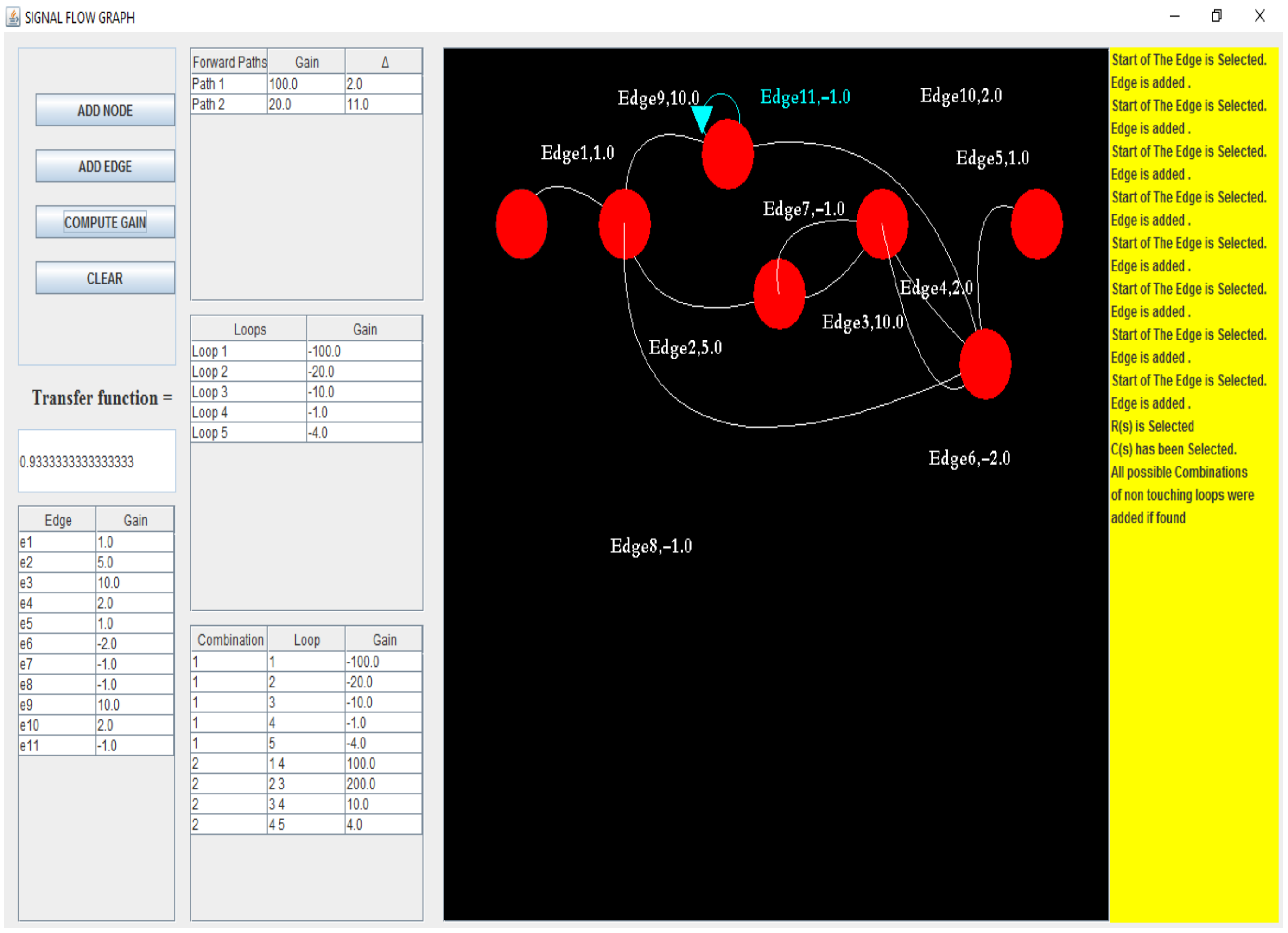
VII. Sample runs:

For testing, problem 3 in sheet 4 was used:

The result by hand was 14/15 =0.93333

[3]. Find the gains $\frac{y_5}{y_1}$ for the signal flow graph in the following graph.





Forward Paths	Gain	Δ
Path 1	100.0	2.0
Path 2	20.0	11.0

Loops	Gain
Loop 1	-100.0
Loop 2	-20.0
Loop 3	-10.0
Loop 4	-1.0
Loop 5	-4.0

Combination	Loop	Gain
1	1	-100.0
1	2	-20.0
1	3	-10.0
1	4	-1.0
1	5	-4.0
2	1 4	100.0
2	2 3	200.0
2	3 4	10.0
2	4 5	4.0

Transfer function =

0.9333333333333333

Edge	Gain
e1	1.0
e2	5.0
e3	10.0
e4	2.0
e5	1.0
e6	-2.0
e7	-1.0
e8	-1.0
e9	10.0
e10	2.0
e11	-1.0

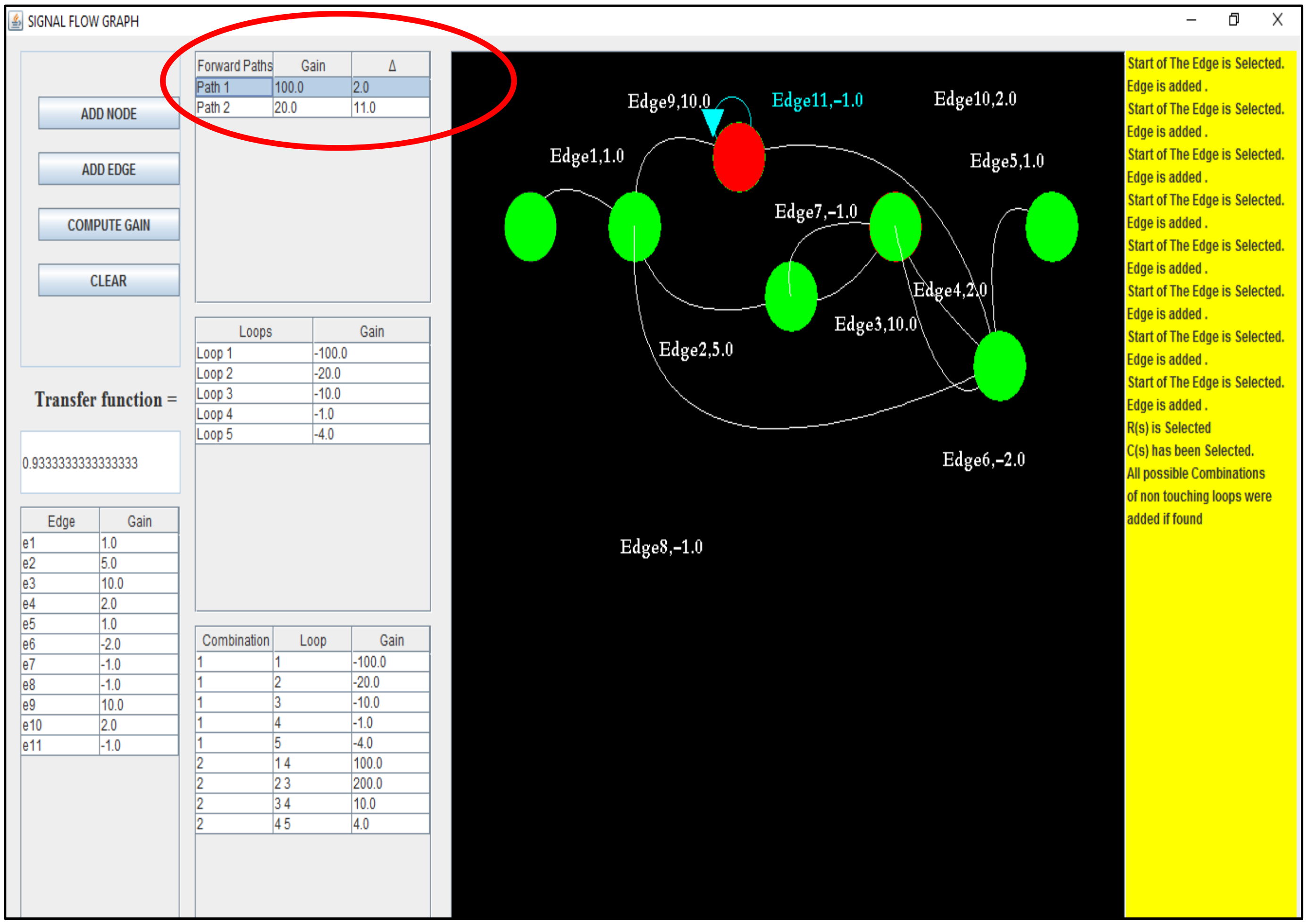
```
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
```

By referring to the solution of this problem that was solved in the sheet:

There were 2 forward paths and 5 loops as detected by the program

By clicking any row in the tables (forward paths, loops or combinations) the color of the nodes in the chosen path are set to green

Example 1: clicking on path 1:



Forward Paths	Gain	Δ
Path 1	100.0	2.0
Path 2	20.0	11.0

Loops	Gain
Loop 1	-100.0
Loop 2	-20.0
Loop 3	-10.0
Loop 4	-1.0
Loop 5	-4.0

Combination	Loop	Gain
1	1	-100.0
1	2	-20.0
1	3	-10.0
1	4	-1.0
1	5	-4.0
2	1 4	100.0
2	2 3	200.0
2	3 4	10.0
2	4 5	4.0

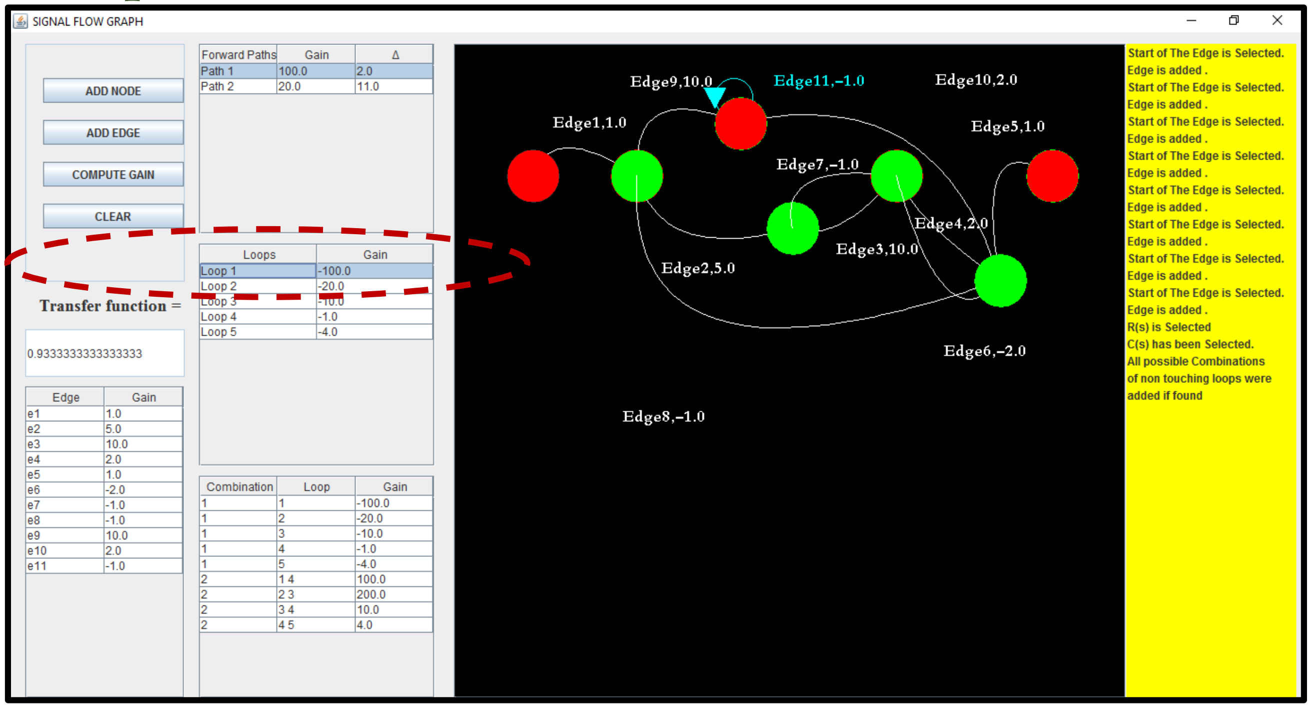
Transfer function =

0.9333333333333333

Edge	Gain
e1	1.0
e2	5.0
e3	10.0
e4	2.0
e5	1.0
e6	-2.0
e7	-1.0
e8	-1.0
e9	10.0
e10	2.0
e11	-1.0

```
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
```

***Example 2**: clicking on loop 1 makes all the nodes in this loop green:*



Forward Paths	Gain	Δ
Path 1	100.0	2.0
Path 2	20.0	11.0

Loops	Gain
Loop 1	-100.0
Loop 2	-20.0
Loop 3	-10.0
Loop 4	-1.0
Loop 5	-4.0

Transfer function =

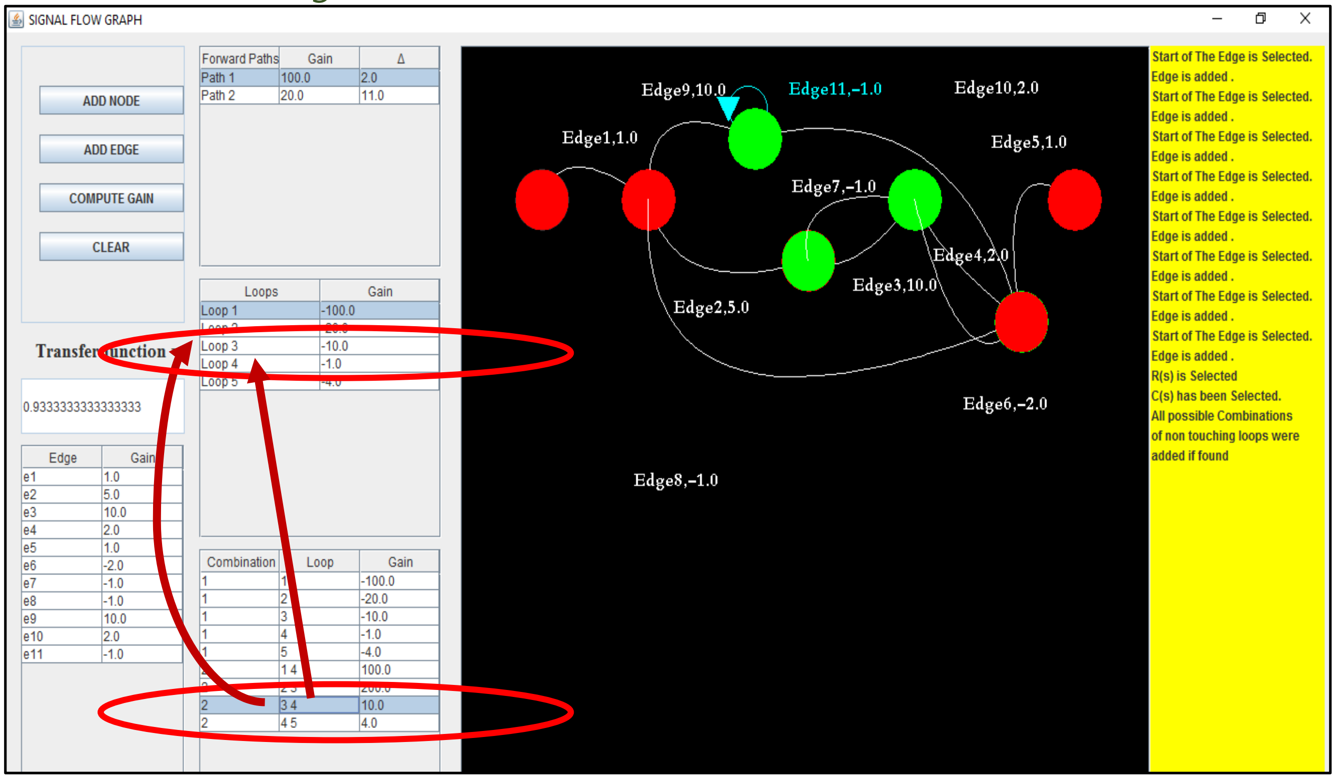
0.9333333333333333

Edge	Gain
e1	1.0
e2	5.0
e3	10.0
e4	2.0
e5	1.0
e6	-2.0
e7	-1.0
e8	-1.0
e9	10.0
e10	2.0
e11	-1.0

Combination	Loop	Gain
1	1	-100.0
1	2	-20.0
1	3	-10.0
1	4	-1.0
1	5	-4.0
2	1 4	100.0
2	2 3	200.0
2	3 4	10.0
2	4 5	4.0

```
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
```

***Example 3:** clicking on combination formed of 2 loops which are LOOP 3 and LOOP 4 in the above list of loops makes all the nodes in this combination green:*



Forward Paths	Gain	Δ
Path 1	100.0	2.0
Path 2	20.0	11.0

Loops	Gain
Loop 1	-100.0
Loop 2	-20.0
Loop 3	-10.0
Loop 4	-1.0
Loop 5	-4.0

Transfer function

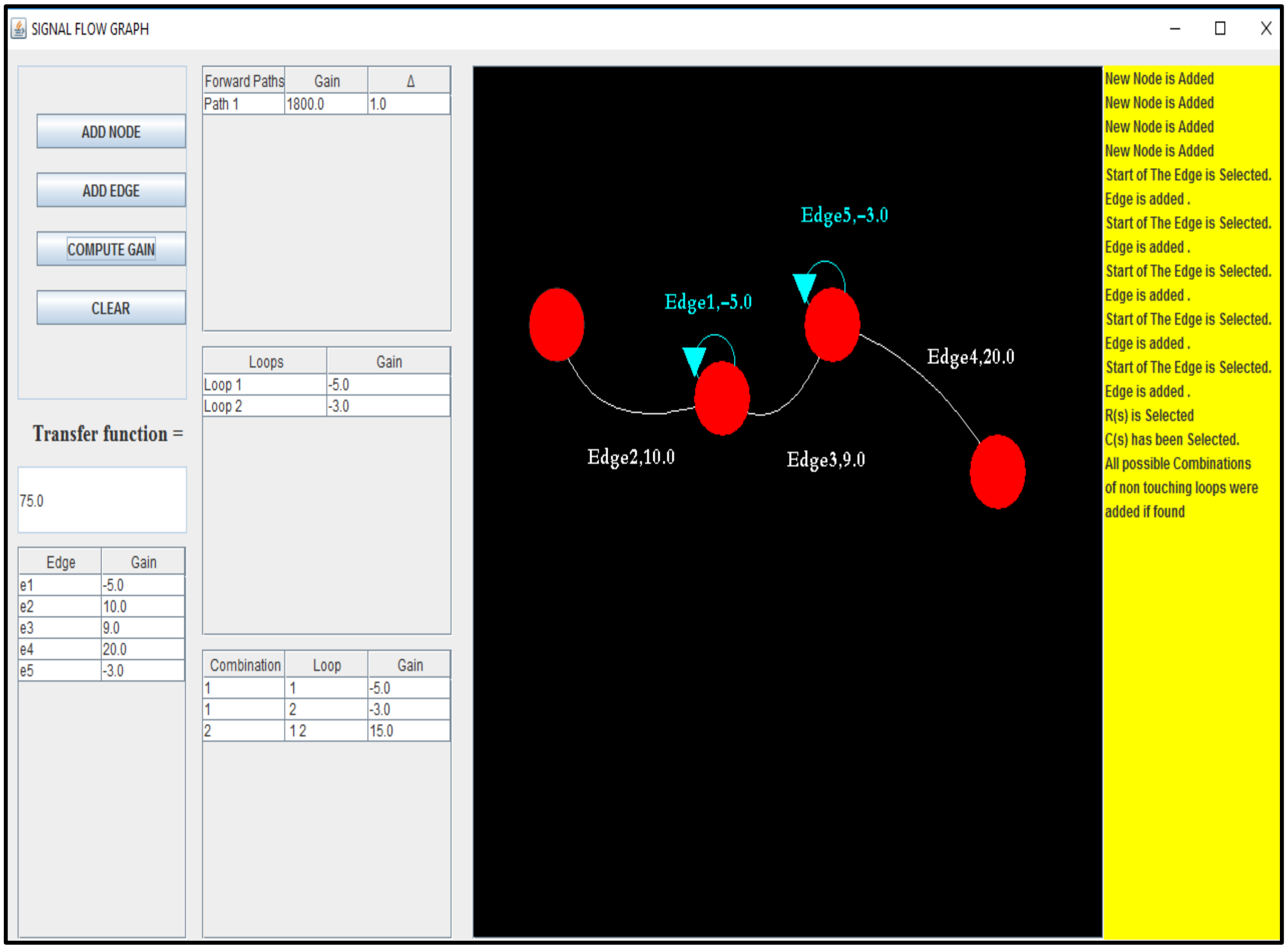
0.9333333333333333

Edge	Gain
e1	1.0
e2	5.0
e3	10.0
e4	2.0
e5	1.0
e6	-2.0
e7	-1.0
e8	-1.0
e9	10.0
e10	2.0
e11	-1.0

Combination	Loop	Gain
1	1	-100.0
1	2	-20.0
1	3	-10.0
1	4	-1.0
1	5	-4.0
2	1 4	100.0
2	2 3	200.0
2	3 4	10.0
2	4 5	4.0

```
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
```

Another Sample Run:



Transfer function =

75.0

Forward Paths	Gain	Δ
Path 1	1800.0	1.0

Loops	Gain
Loop 1	-5.0
Loop 2	-3.0

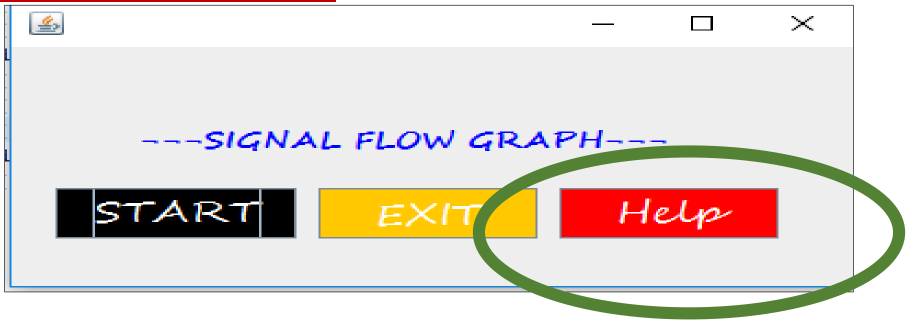
Combination	Loop	Gain
1	1	-5.0
1	2	-3.0
2	1 2	15.0

Edge	Gain
e1	-5.0
e2	10.0
e3	9.0
e4	20.0
e5	-3.0

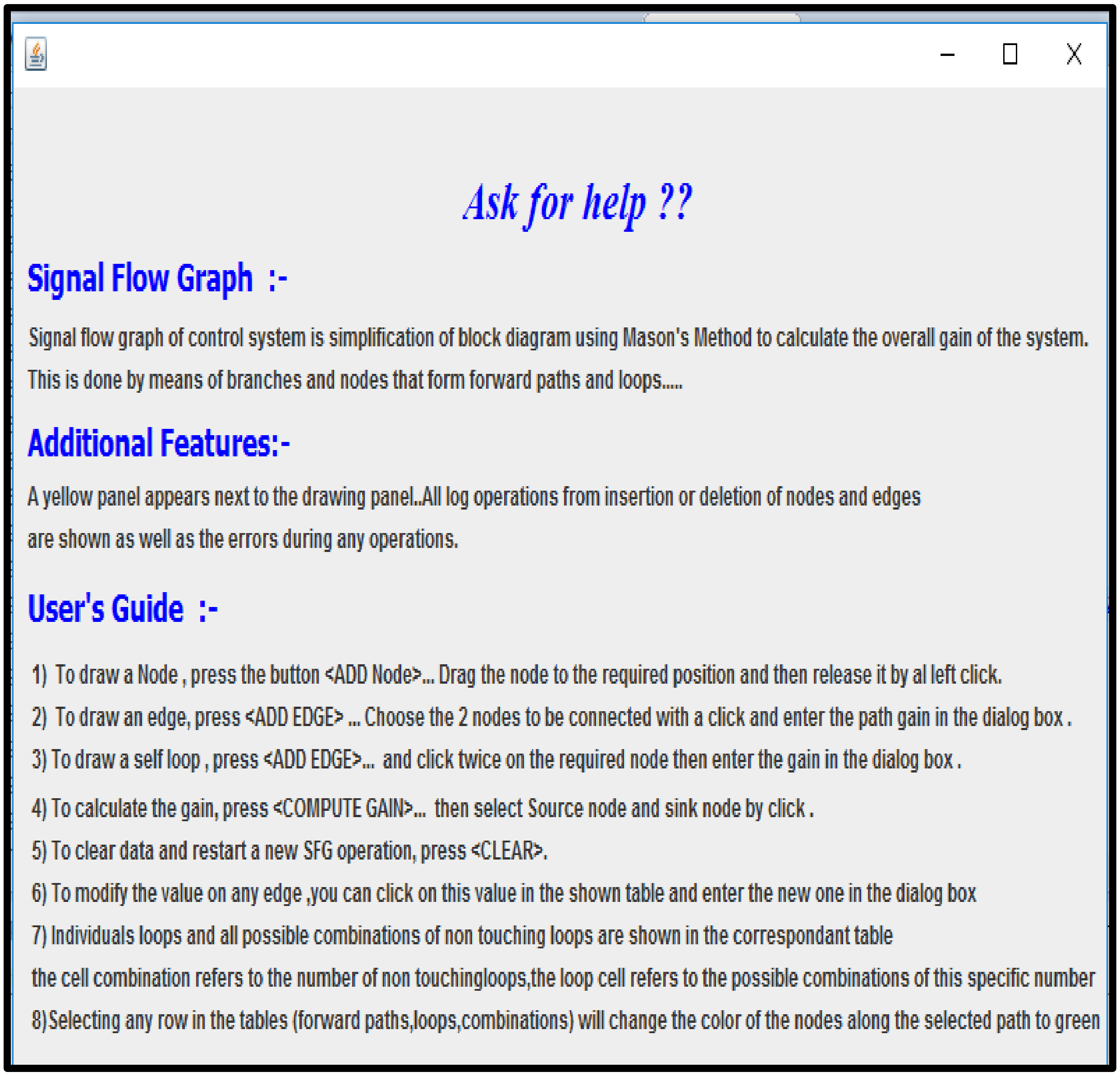
```
New Node is Added
New Node is Added
New Node is Added
New Node is Added
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
Start of The Edge is Selected.
Edge is added .
R(s) is Selected
C(s) has been Selected.
All possible Combinations
of non touching loops were
added if found
```

VIII. Simple user guide:

- ***Main features of the program and additional features can be used as refrences for additional info***



Once HELP is clicked:



Ask for help ??

Signal Flow Graph :-

Signal flow graph of control system is simplification of block diagram using Mason's Method to calculate the overall gain of the system.
This is done by means of branches and nodes that form forward paths and loops.....

Additional Features:-

A yellow panel appears next to the drawing panel..All log operations from insertion or deletion of nodes and edges
are shown as well as the errors during any operations.

User's Guide :-

1) To draw a Node , press the button <ADD Node>... Drag the node to the required position and then release it by al left click.
2) To draw an edge, press <ADD EDGE> ... Choose the 2 nodes to be connected with a click and enter the path gain in the dialog box .
3) To draw a self loop , press <ADD EDGE>... and click twice on the required node then enter the gain in the dialog box .
4) To calculate the gain, press <COMPUTE GAIN>... then select Source node and sink node by click .
5) To clear data and restart a new SFG operation, press <CLEAR>.
6) To modify the value on any edge ,you can click on this value in the shown table and enter the new one in the dialog box
7) Individuals loops and all possible combinations of non touching loops are shown in the correspondant table
the cell combination refers to the number of non touchingloops,the loop cell refers to the possible combinations of this specific number
8)Selecting any row in the tables (forward paths,loops,combinations) will change the color of the nodes along the selected path to green