CHAPTER 5
IMPLEMENTATION AND TESTING

5.1 Implementation

The project's data structure is tree and using Heap Sort as the algorithm. Tree is used to store the data, data represented into binary tree. The sorting process happens in tree or heap and split up into two trees. Here are some code sections of the program:

```
1. DisplayThread dt = new DisplayThread();
2.       public  void  Minheap(Node  root,  Node  prev,  PaintTree
   paintTree)
3.       {
4.             if(root==null)
5.             return;
6.
7.             try{
8.
9.                   minheap(root.left,root,paintTree);
10.                  minheap(root.right,root,paintTree);
11.
12.            if( prev != null && root.data <prev.data){
13.                              swapper(root,prev);
14.                              prev = root;
15.                              minheap(prev, prev,paintTree);
16.                        }
17.                  }catch (Exception e) {
18.                        e.printStackTrace();
19.                  }
20.
21.            }
```

Following code above is used to compare each nodes and swap them. Root's value must be the smallest than its child. The method called MinHeap, code on line 4 and 5 is for returning value of root if it is null. Code on line 9 to 15 is used for swapping each nodes, the root's value must be the smallest than its child, the swapping process runs recursively.

```
  public Node[] dualHeap(Node root){
22.                 Node data = root;
23.                 Node result[] = new Node[2];
24.
25.                 Tree tree1 = new Tree();
26.                 tree1.root = root.left;
27.                 Tree tree2 = new Tree();
28.                 tree2.root = root.right;
29.
30.                 result[0] = tree1.root;
31.                 result[1] = tree2.root;
32.
33.                 cetakHeap(root);
34.       System.out.println("\n===Tree Split Pertama===");
                  tree1.cetakHeap(tree1.root);

35.      System.out.println("\n===Tree Split Kedua===");
36.                 tree2.cetakHeap(tree2.root);
37.
38.                 return result;
39.           }
```

The code section above is a method to split up the tree into 2 trees to implement sorting using multiple heaps. Code on line 22 is used to make data equals root, the data taken from class Node.java. The code underneath used to make new node which become the result. Code line 25 to 28 is the main code, it's split up tree into tree1 and tree2. Code on line 33 to 36 display the splitted tree, tree1 become the left tree and tree2 become the right tree.

```
   private void displayTree(Graphics g, Node root, int x, int y,
  int hGap)[5] {
40.       g.drawOval(x-radius, y-radius, 2 * radius, 2 * radius);
41.           g.drawString(root.data + "", x - 6, y + 4);
42.
43.             if (root.left != null) {
44.             connectLeftChild(g, x - hGap, y + vGap, x, y);
45.     displayTree(g, root.left, x - hGap, y + vGap, hGap / 2);
46.
47.           }
48.
49.           if (root.right != null) {
50.             connectRightChild(g, x + hGap, y + vGap, x, y);
51.     displayTree(g, root.right, x + hGap, y + vGap, hGap / 2);
52.           }
53.     }
```

The following code above is function to display binary tree starts from draw oval and draw string. The method called displayTree, inside the method there are many variabels such as variabel for graphics, root, coordinates, and gap between nodes. Code on line 40 draws oval which become the node, x and y are the coordinates each nodes. Then code on line 41 draws string, value inside node.

The code on line 43 to 45 draws left parent recursively and connecting its child with the gap between them. The rest of code also draws right parent recursively and connecting its child with the gap.

5 http://www.ecs.csun.edu/~cputnam/Comp%20110/Liang%208th%20ed%20PP_Slides/html/BinaryTreeView.html

```
54.     if(root.right != null){
55.     g.drawString(root.data + "", x - 6, y + 4);
56.     g.setColor(Color.red);
57.
58.                 }else{
59.                 g.setColor(Color.blue);
60.     }
61.
62.     if(root.left !=null){
63.     g.drawString(root.data + "", x - 6, y + 4);
64.     g.setColor(Color.magenta);
65.
66.                 }else{
67.                 g.setColor(Color.green);
68.     }
```

The function of code above is to colouring tree. Code on line 54 to 59 is used to colouring the right side of tree recursively. If the value of right root doesn't equals null then it coloured red, or else it's blue. The code on line 62 to 68 also used to colouring the left side of tree resursively. If the value of left root doesn't equals null then it coloured magenta, or else it's green.

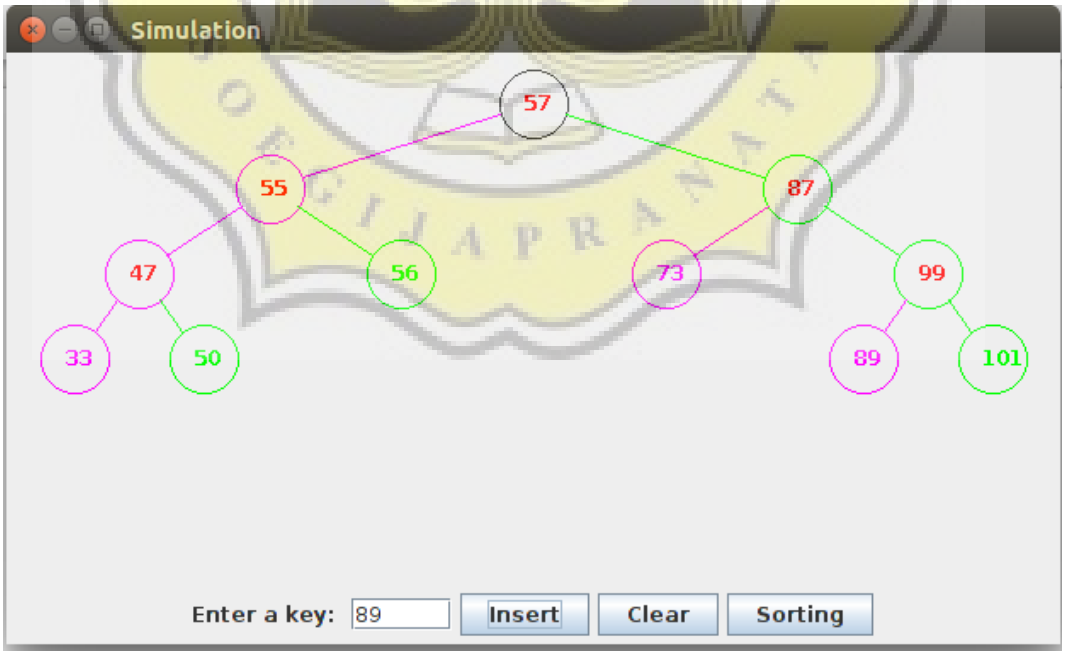

Illustration 5.1: Heap Sort Visualization.

5.2 Testing

There are five samples of Heap Sort visualization that will be used for testing on this project. Each sample has different values of numbers. The testing goal is to find out time complexity of Heap Sort.

	Data 1	**Data 2**	**Data 3**	**Data 4**	**Data 5**	**Data 6**	**Data 7**	**Data 8**	**Data 9**	**Data 10**
	21	51	15	51	43	70	43	57	30	66
	19	20	7	43	35	67	20	54	39	69
	15	13	5	37	27	55	17	33	23	67
	20	4	24	20	13	69	15	22	50	53
	35	15	27	15	30	68	19	24	46	44
	27	27		25	39	34	34	43	67	39
	41	55		40	58	28	27	35	59	21
		53		73	55	25	56	47	13	77
		76		61	57	47	49	55		80
				55	63	41	66	69		95
				63		50	88	59		11
				77		97		58		25
						89		60		33
						77		78		
						71		77		
						93				
						90				
						105				
						101				
						99				
						134				
Data Total	**7**	**9**	**5**	**12**	**10**	**21**	**11**	**15**	**8**	**13**
Time (sec)	**9**	**11**	**8**	**15**	**10**	**27**	**10**	**16**	**9**	**14**

Table 5.1: Data Testing.

Based on table above, time that Heap Sort needs to process the data quiet fast. Average time of Heap Sort is 1.4 seconds per data is need to be sorted. Comparing the least and the most data which are data 3 and 6, the amount of data has no effect on Heap Sort performance.

The program can colouring tree correctly, each node has different colour with the next node. The problem is colour limitation, the program could only draw tree in 4 colours, red, green, magenta, and blue. Other problem is the line and oval shape also come coloured.