`

1. Introduction

1.1 Introduction

The first step towards an understanding of why the study and knowledge of algorithms are so important is to define exactly what we mean by an algorithm. According to the popular algorithms textbook Introduction to Algorithms (Second Edition by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest , Clifford Stein), "an algorithm is any well-defined computational procedure that takes some value, or set of values, as input and produces some value, or set of values as output." In other words, algorithms are like road maps for accomplishing a given, well-defined task. So, a chunk of code that calculates the terms of the Fibonacci sequence is an implementation of a particular algorithm. Even a simple function for adding two numbers is an algorithm in a sense, albeit a simple one.

Some algorithms, like those that compute the Fibonacci sequences, are intuitive and may be innately embedded into our logical thinking and problem solving skills. However, for most of us, complex algorithms are best studied so we can use them as building blocks for more efficient logical problem solving in the future. In fact, you may be surprised to learn just how many complex algorithms people use every day when they check their e-mail or listen to music on their computers. This article will introduce some basic ideas related to the analysis of algorithms, and then put these into practice with a few examples illustrating why it is important to know about algorithms.

Sorting provides a good example of an algorithm that is very frequently used by computer scientists. The simplest way to sort a group of items is to start by removing the smallest item from the group, and put it first. Then remove the next smallest, and put it next and so on. Unfortunately, this algorithm is O(N2), meaning that the amount of time it takes is proportional to the number of items squared. If you had to sort a billion things, this algorithm would take around 10^{18} operations. To put this in perspective, a desktop PC can do a little bit over 109 operations per second, and would take years to finish sorting a billion things this way.

Luckily, there are a number of better algorithms (quick sort, heapsort and merge sort, for example) that have been devised over the years, many of which have a runtime of O(N * Log(N)). This brings the number of operations required to sort a billion items down to a

`

reasonable number that even a cheap desktop could perform. Instead of a billion squared operations (10^{18}) these algorithms require only about 10 billion operations (10^{10}), a factor of 100 million faster.

As a computer scientist, it is important to understand all of these types of algorithms so that one can use them properly. If you are working on an important piece of software, you will likely need to be able to estimate how fast it is going to run. Such an estimate will be less accurate without an understanding of runtime analysis. Furthermore, you need to understand the details of the algorithms involved so that you'll be able to predict if there are special cases in which the software won't work quickly, or if it will produce unacceptable results.

Of course, there are often times when you'll run across a problem that has not been previously studied. In these cases, you have to come up with a new algorithm, or apply an old algorithm in a new way. The more you know about algorithms in this case, the better your chances are of finding a good way to solve the problem. In many cases, a new problem can be reduced to an old problem without too much effort, but you will need to have a fundamental understanding of the old problem in order to do this.

As an example of this, let's consider what a switch does on the Internet. A switch has N cables plugged into it, and receives packets of data coming in from the cables. The switch has to first analyze the packets, and then send them back out on the correct cables. A switch, like a computer, is run by a clock with discrete steps – the packets are send out at discrete intervals, rather than continuously. In a fast switch, we want to send out as many packets as possible during each interval so they don't stack up and get dropped. The goal of the algorithm we want to develop is to send out as many packets as possible during each interval, and also to send them out so that the ones that arrived earlier get sent out earlier. In this case it turns out that an algorithm for a problem that is known as "stable matching" is directly applicable to our problem, though at first glance this relationship seems unlikely. Only through pre-existing algorithmic knowledge and understanding can such a relationship be discovered.

`

1.2 Aim and Objectives

The use of interactive technology in learning for the students is as natural as using a pencil and paper were to past generations. The aim of the project is to provide an interactive learning Application using Java FX libraries and Java graphic libraries. The application contains three modules consisting of Learning, Quiz and Theory. Since Interactive Learning is a pedagogical approach that incorporates social networking and urban computing into intercourse design and delivery. Interactive Learning has evolved out of the hyper-growth in the use of digital technology and virtual communication, particularly by students. The basis of the project is to make an individual capable enough to learn sorting techniques and graph traversal methods without the help of teachers through an interactive learning process.

The main objective of the project is to make the user comfortable with sorting techniques and Graph Traversal through an interactive learning process. With the help of three modules in the project namely Learning, quiz and theory. A learning environment is “interactive” in the sense that a person can navigate through it, select relevant information, respond to questions using computer input devices such as a keyboard, mouse, touch screen, or voice command system, solve problems, complete challenging tasks, create knowledge representations, collaborate with others near or at a distance, or otherwise engage in meaningful learning activities.

1.3 Scope

The game is an interactive learning tool used to help players learn various sorting algorithms with the help of Java graphic primitives. The game has two characters a father and a child. The father is the Computer and the program and the child is the user which interacts with the computer software. We aim to teach the users the functioning of the algorithms while they are playing the game. The user will then be tested about the learning of the algorithms. The user can play fun games if he gets bored while he is learning the algorithms.

`

2. Review of Literature

2.1 Domain Explanation

Domain: Application Development in JAVA

Java FX is a Java library used to build Rich Internet Applications. The applications written using this library can run consistently across multiple platforms. The applications developed using Java FX can run on various devices such as Desktop Computers, Mobile Phones, TVs, Tablets, etc To develop GUI Applications using Java programming language, the programmers rely on libraries such as Advanced Windowing Toolkit and Swings. After the advent of JavaFX, these Java programmers can now develop GUI applications effectively with rich content.

2.1.1 JavaFx

2.1.1.1 Stage

A stage (a window) contains all the objects of a JavaFX application. It is represented by stage class of the package java fx stage. The primary stage is created by the platform itself. The created stage object is passed as an argument to the start() method of the application class

A stage has two parameters determining its position namely width and height It is divided as Content Area and Decorations (Title Bar and Borders).

There are five types of stages available −

- Decorated
- Undecorated
- Transparent
- Unified
- Utility

You have to call the show**()** method to display the contents of a stage.

`

2.1.1.2 Scene

A scene represents the physical contents of a JavaFX application. It contains all the contents of a scene graph. The class Scene of the package javafx.scene represents the scene object. At an instance, the scene object is added to only one stage.

You can create a scene by instantiating the Scene Class. You can opt for the size of the scene by passing its dimensions (height and width) along with the root node to its constructor.

Scene Graph and Nodes:-

A scene graph is a tree-like data structure (hierarchical) representing the contents of a scene. In contrast, a node is a visual/graphical object of a scene graph.

A node may include −

- Geometrical (Graphical) objects (2D and 3D) such as − Circle, Rectangle, Polygon, etc.
- UI Controls such as − Button, Checkbox, Choice Box, Text Area, etc.
- Containers (Layout Panes) such as Border Pane, Grid Pane, Flow Pane, etc.
- Media elements such as Audio, Video and Image Objects.

The Node Class of the package javafx.scene represents a node in JavaFX, this class is the super class of all the nodes.

As discussed earlier a node is of three types −

Root Node − The first Scene Graph is known as the Root node.

Branch Node/Parent Node:-The nodes with child nodes are known as branch/parent nodes. The abstract class named Parent of the package javafx.scene is the base class of all the parent nodes, and those parent nodes will be of the following types −

Group − A group node is a collective node that contains a list of children nodes. Whenever the group node is rendered, all its child nodes are rendered in order. Any transformation, effect state applied on the group will be applied to all the child nodes.

`

Region: - It is the base class of all the JavaFX Node based UI Controls, such as Chart, Pane and Control.

Web View: - This node manages the web engine and displays its contents.

- **Leaf Node** − the node without child nodes is known as the leaf node. For example, Rectangle, Ellipse, Box, Image View, MediaView are examples of leaf nodes.

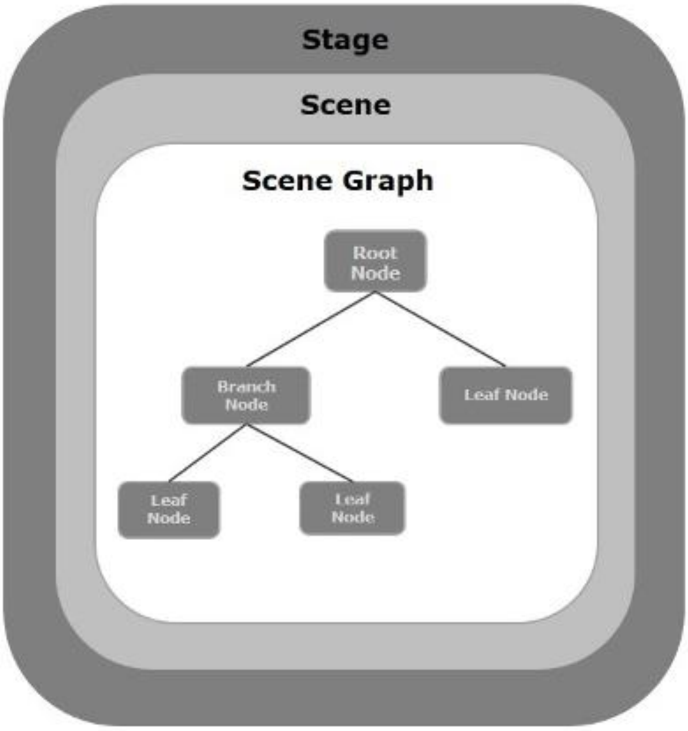


Fig: 2.1

2.1.2 Java Swing

Java Swing tutorial is a part of Java Foundation Classes (JFC) that is *used to create window-based applications*. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java. Unlike AWT, Java Swing provides platform-independent and lightweight components. The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc. Java swing components are platform-independent. Swing is light weight Swing supports pluggable look and feel. Swing provides more powerful components Swing follows MVC.

`

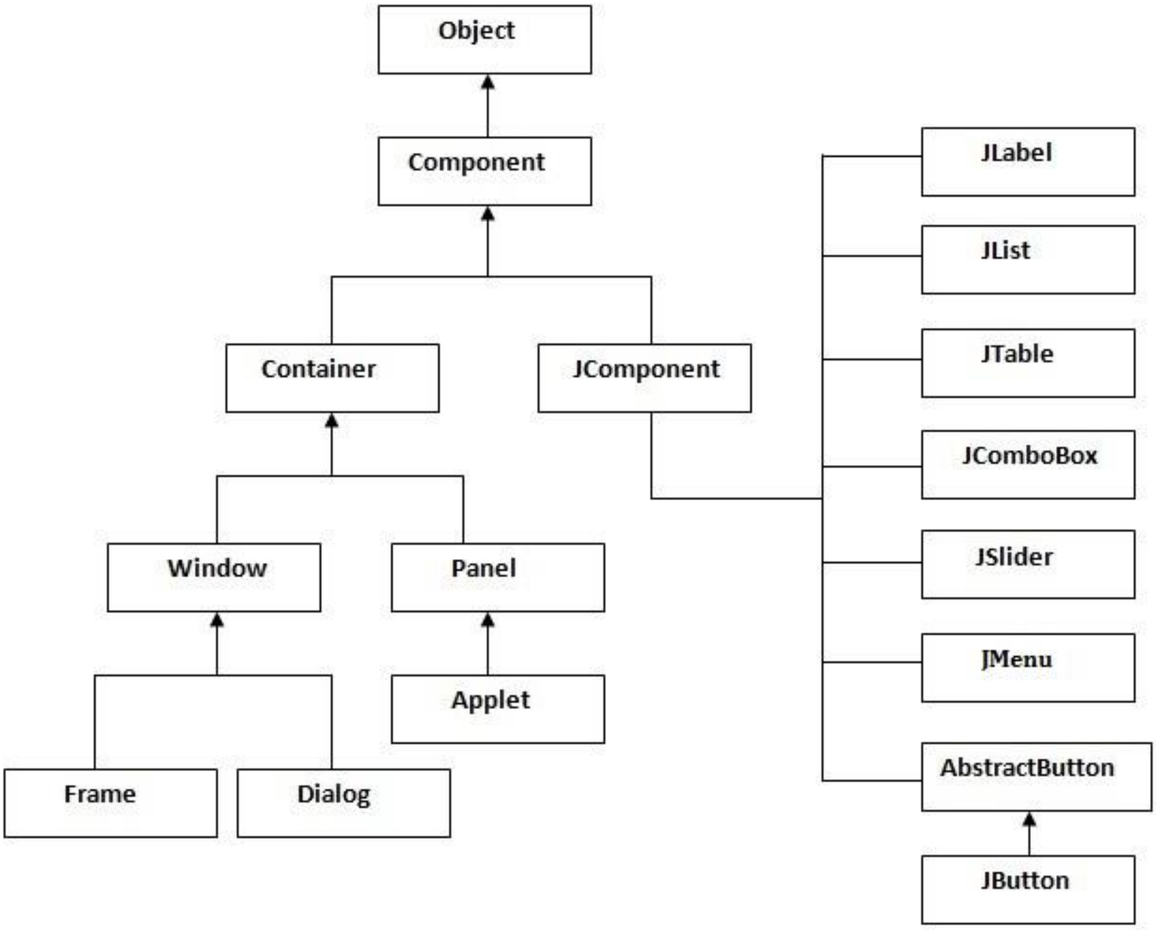


Figure 2.2

2.1.3 Java JFrame

A frame, implemented as an instance of the **JFrame** class, is a window that has decorations such as a border, a title, and supports button components that close or iconify the window. Applications with a GUI usually include at least one frame. Applets sometimes use frames, as well.

. . .

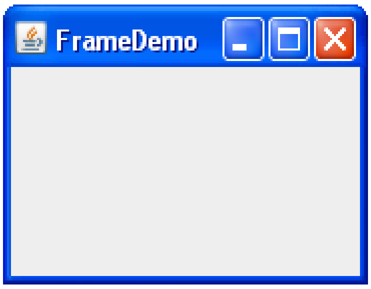


Figure 2.3

`

The following FrameDemo code shows how to create and set up a frame.

1. Create the frame.
Jframe frame = new Jframe("FrameDemo");

2. Optional: What happens when the frame closes?
frame.setDefaultCloseOperation (JFrame.EXIT_ON_CLOSE);

3. Create components and put them in the frame.
...create empty Label...
frame.getContentPane ().add (Empty Label, BorderLayout.CENTER);

4. Size the frame.
frame.pack();

5. Show it.
frame.setVisible(true);

2.1.4 CSS

JavaFX provides you the facility of using CSS to enhance the look and feel of the application. The package **javafx.css** contains the classes that are used to apply CSS for JavaFX applications.

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document.

A style rule is made of three parts, which are −

Selector − A selector is an HTML tag at which a style will be applied. This could be any tag like **<h1>** or **<table>**, etc.

Property − A property is a type of attribute of the HTML tag. In simpler terms, all the HTML attributes are converted into CSS properties. They could be colour, **border**, etc.

Value − Values are assigned to properties. For example, a color property can have value either **red** or **#F1F1F1**, etc.

`

You can put CSS Style Rule Syntax as follows −

Selector{ Property: Value;}

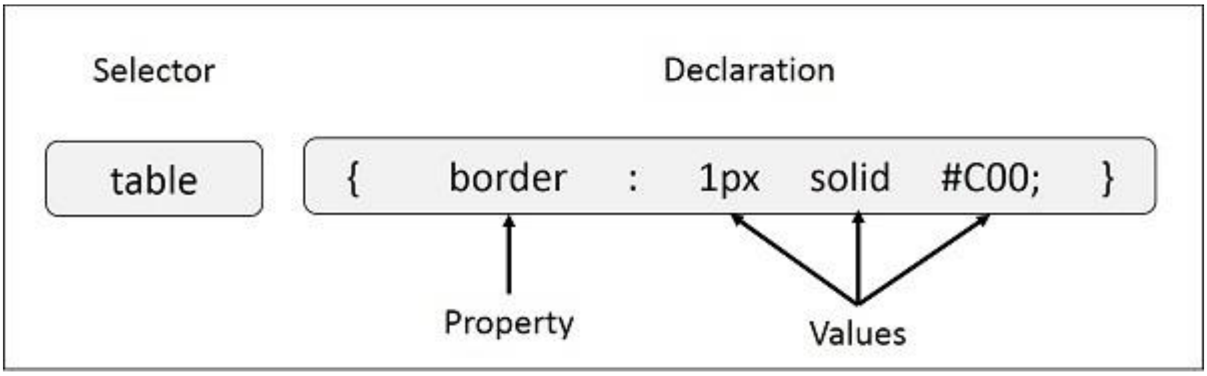


Figure 2.4

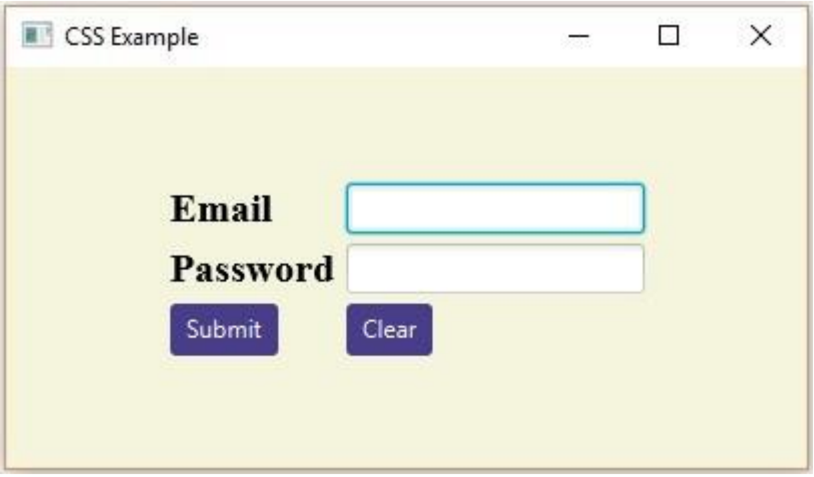


Figure 2.5

2.2 Existing Solution

2.2.1 Adobe After Effects

It is a digital visual effects, motion graphics, and compositing application developed by Adobe Systems and used in the post-production process of film making and television production. Among other things, After Effects can be used for keying, tracking, compositing and animation. It also functions as a very basic non-linear editor, audio editor and media transcoder.

`

Figure 2.6

2.2.2 Go Animate

Go animate (formally known as Go! Animate until 2013) is a cloud-based, animated video creation platform. It is designed to allow business people with no background in animation to quickly and easily create animated videos.

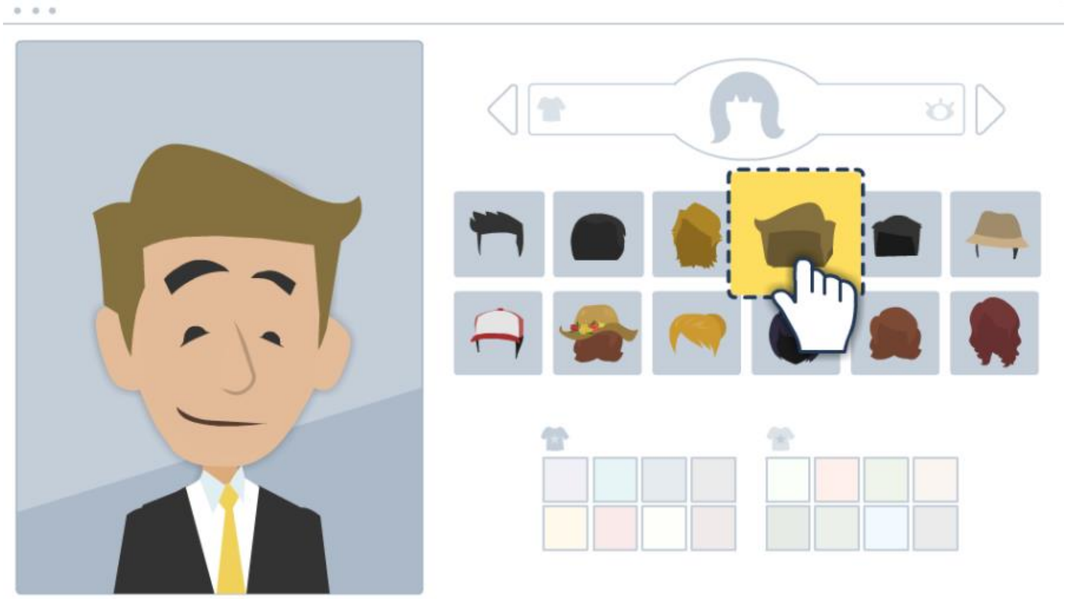


Figure 2.7

2.2.3 XtraNormal

XtraNormal Technology, Inc. was a digital entertainment company that produced do-it-yourself animation software for the web and desktop and turned words from a script into an animated movie using text-to-speech and animation technologies.

`

Figure 2.8

2.2.4 Android Visualizer

An audio virtualizer is a general name for an effect to specialise audio channels. The exact behaviour of this effect is dependent on the number of audio input channels and the types and number of audio output channels of the device. For example, in the case of a stereo input and stereo headphone output, a stereo widening effect is used when this effect is turned on.

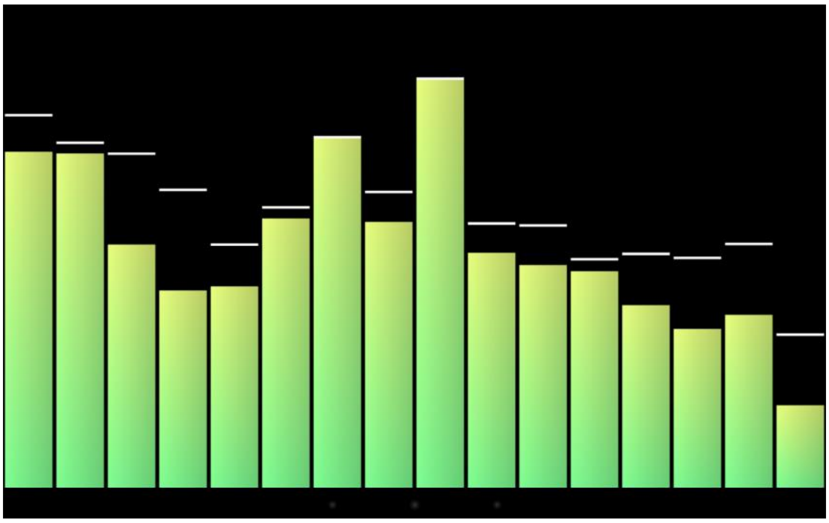


Figure 2.9

`

2.3 Hardware and Software Requirements

2.3.1 Hardware Requirements

JavaFX 1.2 Requirements

Operating Systems

1) Windows:-Windows XP with Service Pack 3 or Windows Vista Home Premium, Business, Ultimate, or Enterprise (certified for 32-bit editions) or Windows 7 Ultimate (certified for 32-bit editions)

2) Mac:- Mac OS X 10.4.10 minimum

3) Linux:-Ubuntu 8.04 LTE

4) Solaris:- Open Solaris 2009.06

Memory

1) Windows:- 512 MB of RAM (2 GB recommended)

2) Mac:- 512 MB of RAM (2 GB recommended)

3) Linux:- 512 MB of RAM (2 GB recommended)

4) Solaris:- 512 MB of RAM (2 GB recommended)

Processor:-

1) Windows:- Intel Pentium 4, Intel Centrino, Intel Xeon, or Intel Core Duo (or compatible) 1.8 GHz minimum (2.6 GHz Intel Pentium 4 or equivalent recommended)

2) Mac:-Dual-Core Intel, PowerPC G5

3) Linux:- Ntel Pentium 4, Intel Centrino, Intel Xeon, or Intel Core Duo (or compatible) 1.8 GHz minimum (2.6 GHz Intel Pentium 4 or equivalent recommended)

4) Solaris:- Intel Pentium 4, Intel Centrino, Intel Xeon, or Intel Core Duo (or compatible) 1.8 GHz minimum (2.8 GHz recommended)

Disk Space

1) Windows:-778 MB of free disk space (1 GB recommended)

2) Mac:-778 MB of free disk space (1 GB recommended**)**

3) Linux:-778 MB of free disk space (1 GB recommended)

4) Solaris:- 778 MB of free disk space (1 GB recommended)

`

2.3.2 Software Requirements

1) Java 8 system requirements

Operating system

Windows

- Windows 10 (8u51 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 R2 SP1 (64-bit)
- Windows Server 2012 and 2012 R2 (64-bit)
- RAM: 128 MB
- Disk space: 124 MB for JRE; 2 MB for Java Update
- Processor: Minimum Pentium 2 266 MHz processor
- Browsers: Internet Explorer 9 and above, Firefox

Linux

- Oracle Linux 5.5+[1]
- Oracle Linux 6.x (32-bit), 6.x (64-bit)[2]
- Oracle Linux 7.x (64-bit)[2] (8u20 and above)
- Red Hat Enterprise Linux 5.5+[1], 6.x (32-bit), 6.x (64-bit)[2]
- Red Hat Enterprise Linux 7.x (64-bit)[2] (8u20 and above)
- Suse Linux Enterprise Server 10 SP2+, 11.x
- Suse Linux Enterprise Server 12.x (64-bit)[2] (8u31 and above)
- Ubuntu Linux 12.04 LTS, 13.x
- Ubuntu Linux 14.x (8u25 and above)
- Ubuntu Linux 15.04 (8u45 and above)
- Ubuntu Linux 15.10 (8u65 and above)
- Browsers: Firefox

Mac

- Intel-based Mac running Mac OS X 10.8.3+, 10.9+
- Administrator privileges for installation
- 64-bit browser

`

Solaris

See supported Java 8 System Configurations for information about supported platforms, operating systems, desktop managers, and browsers.

2) Java 7 system requirements

Operating system

Windows

- Windows 10 (7u85 and above)
- Windows 8.x (Desktop)
- Windows 7 SP1
- Windows Vista SP2
- Windows Server 2008 SP2 and 2008 R2 SP1 (64-bit)
- Windows Server 2012 (64-bit) and 2012 R2 (64-bit)
- RAM: 128 MB; 64 MB for Windows XP (32-bit)
- Disk space: 124 MB
- Browsers: Internet Explorer 7.0 and above, Firefox 3.6 and above

Linux

- Oracle Linux 5.5+
- Oracle Linux 6.x (32-bit), 6.x (64-bit)[3]
- Oracle Linux 7.x (64-bit)[3] (7u67 and above)
- Red Hat Enterprise Linux 5.5+, 6.x (32-bit), 6.x (64-bit)[3]
- Red Hat Enterprise Linux 7.x (64-bit)[3] (7u67 and above)
- Suse Linux Enterprise Server 10 SP2, 11.x
- Suse Linux Enterprise Server 12.x (7u75 and above)
- Ubuntu Linux 10.04 and above
- Browsers: Firefox 3.6 and above

Mac

- Intel-based Mac running Mac OS X 10.7.3 (Lion) or later.
- Administrator privileges for installation
- 64-bit browser

`

2. Design and Implementation

3.1 Design Considerations

Our project comprises of three modules, namely:

I. Learn

II. Quiz

III. Theory

3.1.1 Learn

In learn module the users interacts with the system through the interface to learn different sorting algorithms (Bubble Sort, Selection Sort, Insertion Sort and Quick Sort)and graph traversal methods (BFS, DFS, Dijkstra's Algorithm and Prim's Algorithms) The application provides two forms of methods to study difference sorting techniques i.e with the help of circle and bars. The user provides the numbers differentiated by space as the first input to the application. Then he chooses between circle and bars. And next which algorithm to study is the third input to the application. The output of this module gives the actual working of these algorithms in graphical presentation that helps the user to easily understand algorithms. And similarly for Graph traversal methods. The nodes form the input to the application. And the output is again a graphical representation of the graph showing the shortest path or the path from the starting node to the destination node.

3.1.2 Quiz

In this section of the application the user test its knowledge through an interactive multiple-choice question quiz. The quiz consists of variety of different questions based on sorting and graph traversal methods.

`

3.1.3 Theory

Algorithm: - An algorithm is an effective method that can be expressed within a finite amount of space and time and in a well-defined formal language for calculating a function. Starting from an initial state and initial input (perhaps empty) the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.

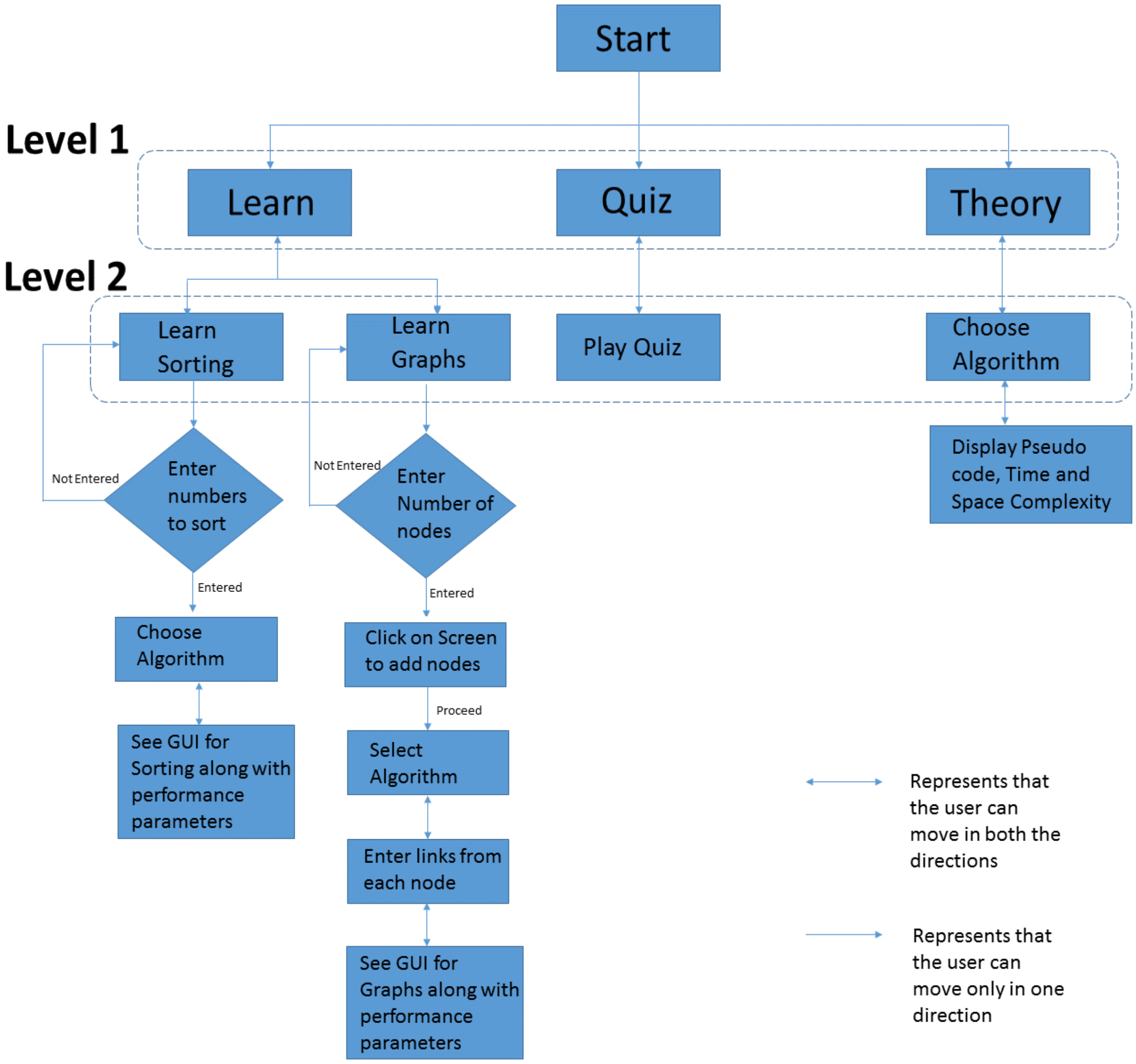
Time complexity :-In computer science, the of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the string representing the input The time complexity of an algorithm is commonly expressed using big O notation, which excludes coefficients and lower order terms. When expressed this way, the time complexity is said to be described *asymptotically*, i.e., as the input size goes to infinity. For example, if the time required by an algorithm on all inputs of size n is at most $5n^3 + 3n$ for any n (bigger than some n_0), the asymptotic time complexity is O (n^3).

Space Complexity:-is a measure of the amount of working storage an algorithm needs. That means how much memory, in the worst case, is needed at any point in the algorithm. As with time complexity, we're mostly concerned with how the space needs grow, in big-Oh terms, as the size N of the input problem grows.

`

3.2 Design Details

3.2.1 Utility Diagram for the Application



Start
Level 1
Learn
Quiz
Theory
Level 2
Learn Sorting
Learn Graphs
Play Quiz
Choose Algorithm
Display Pseudo code, Time and Space Complexity
Not Entered
Enter numbers to sort
Entered
Choose Algorithm
See GUI for Sorting along with performance parameters
Not Entered
Enter Number of nodes
Entered
Click on Screen to add nodes
Proceed
Select Algorithm
Enter links from each node
See GUI for Graphs along with performance parameters
Represents that the user can move in both the directions
Represents that the user can move only in one direction

Figure 3.1

`

3.2.2 Utility Diagram for Learn Module

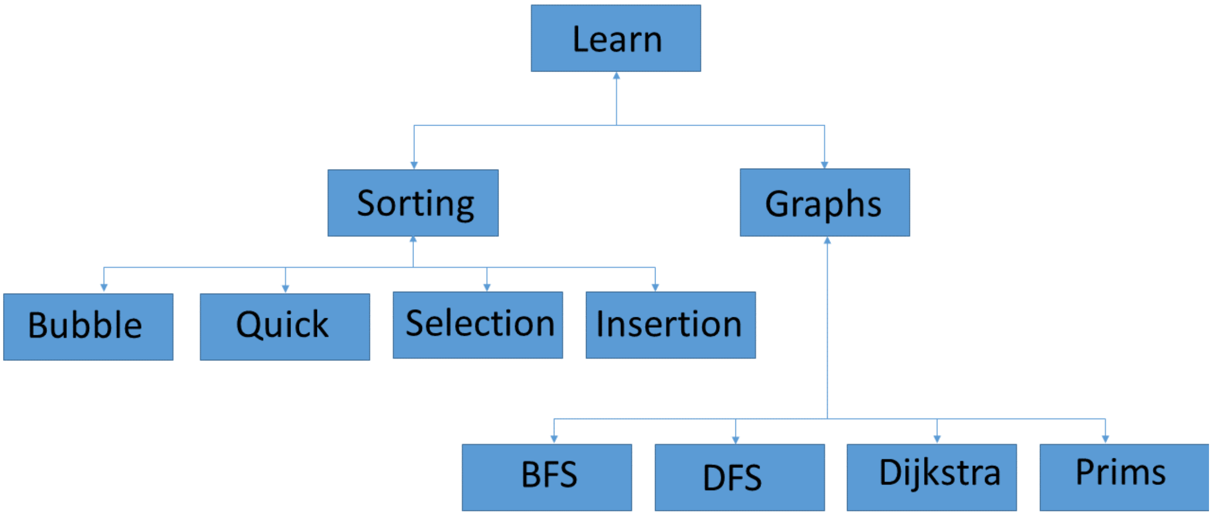


Figure 3.2

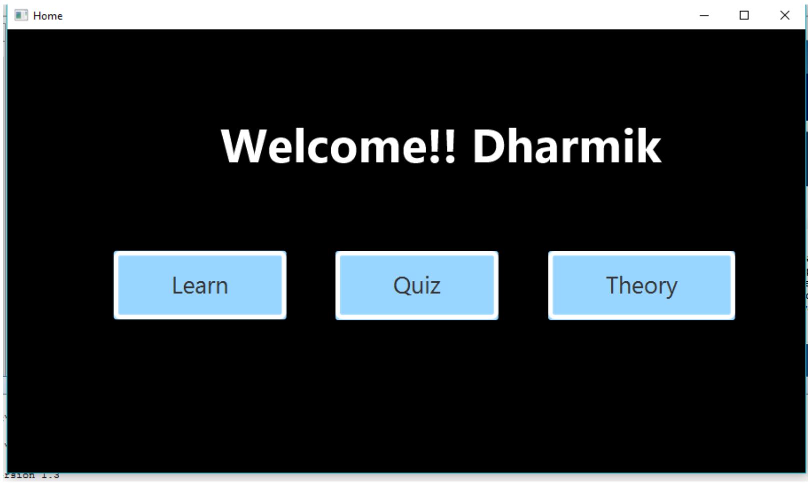
3.3 GUI Design

The GUI of the application is designed using multiple graphic libraries of java such as javafx, java swing, java awt and JPanel. The look and feel of the application is designed keeping into mind all the branches of the engineering stream. Without proper ornamentation and styling the user will not be able to understand the concepts. The main aim of the application was to give user visual perception of the sorting algorithms and graph traversals.

The styling to the application is done by CSS. The CSS is embedded in the javafx and implemented in the application. CSS is used to give background, text decorations, button decorations, etc.

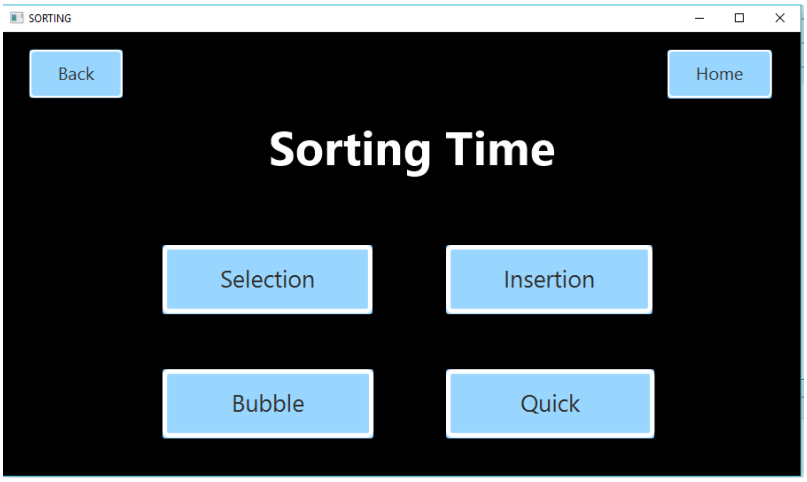
The Quiz pages and the theory pages are designed using JFrame. JFrame allows us to add labels, buttons, text areas to give a clean look to the page.

`



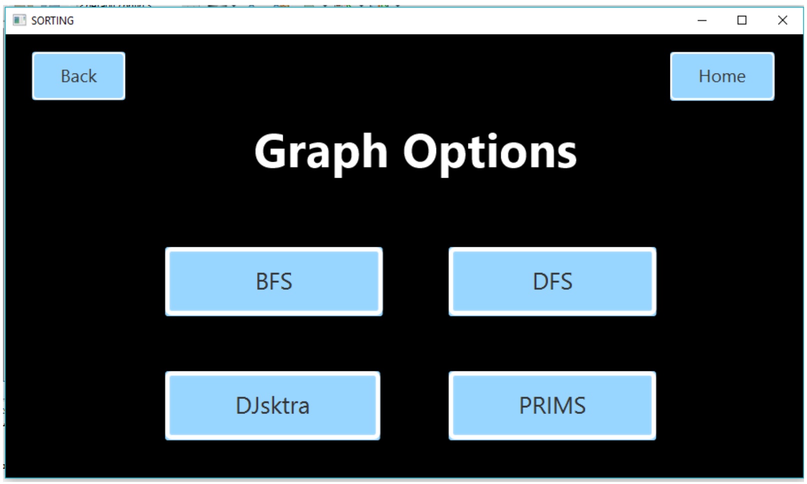
Welcome!! Dharmik

Figure 3.3



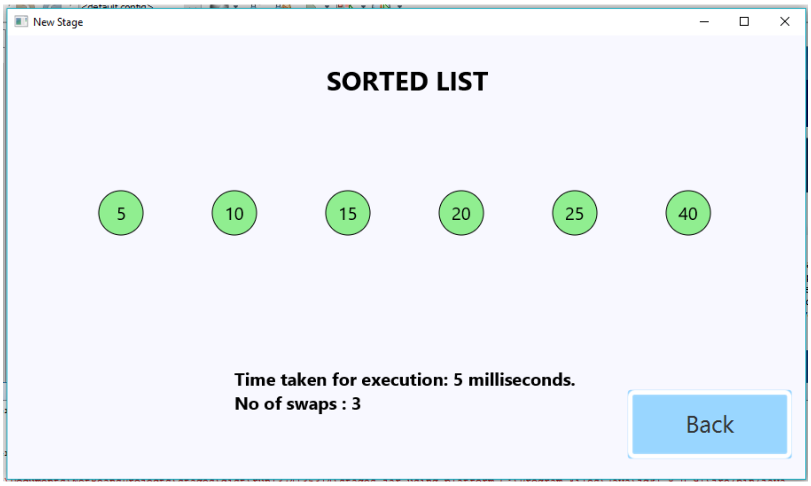
Sorting Time

Figure 3.4



Graph Options

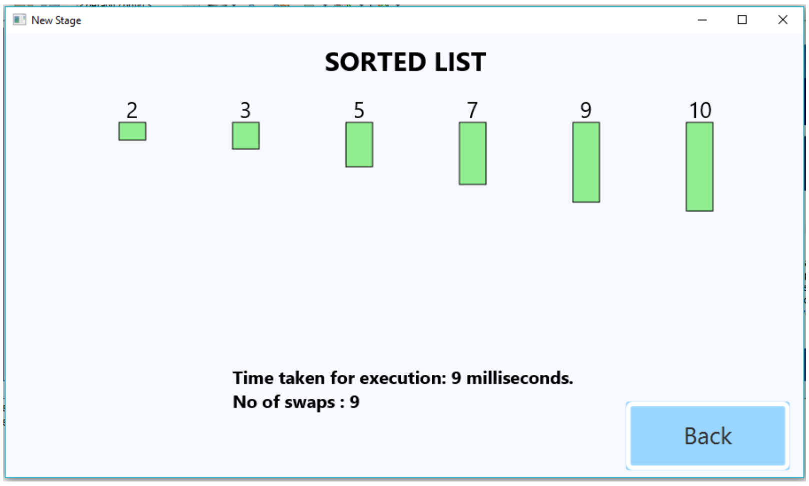
Figure 3.5



SORTED LIST

Time taken for execution: 5 milliseconds.
No of swaps : 3

Figure 3.6



SORTED LIST

Time taken for execution: 9 milliseconds.
No of swaps : 9

Figure 3.7

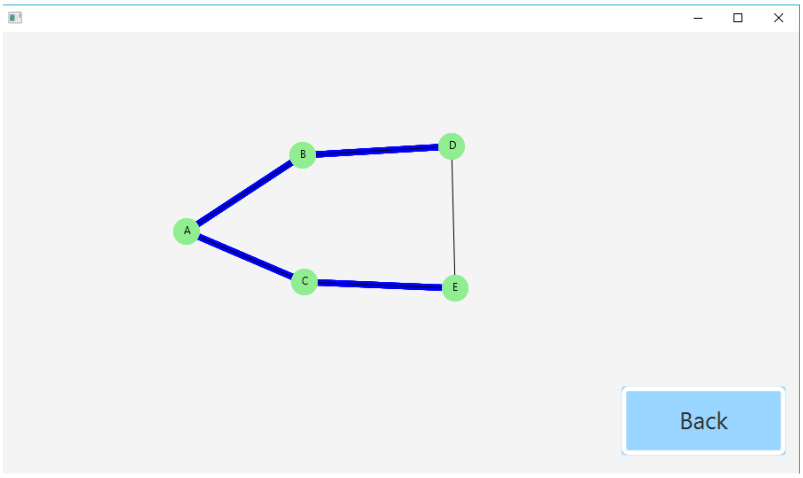
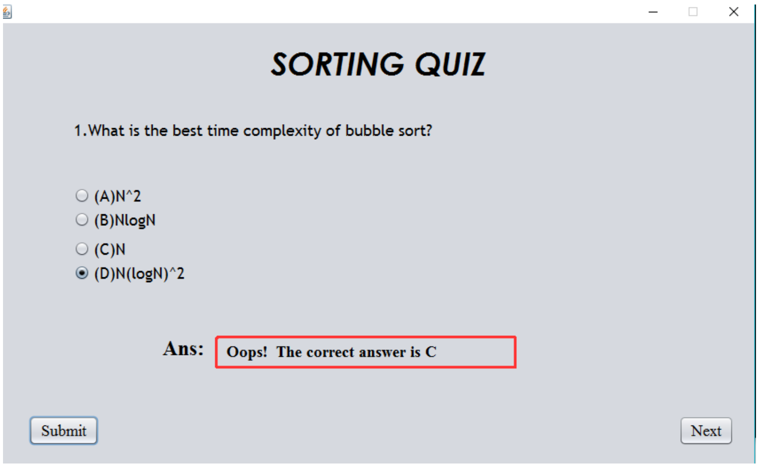


Figure 3.8

`



SORTING QUIZ

1.What is the best time complexity of bubble sort?

○ (A)N^2
○ (B)NlogN
○ (C)N
● (D)N(logN)^2

Ans: Oops! The correct answer is C

Figure 3.9

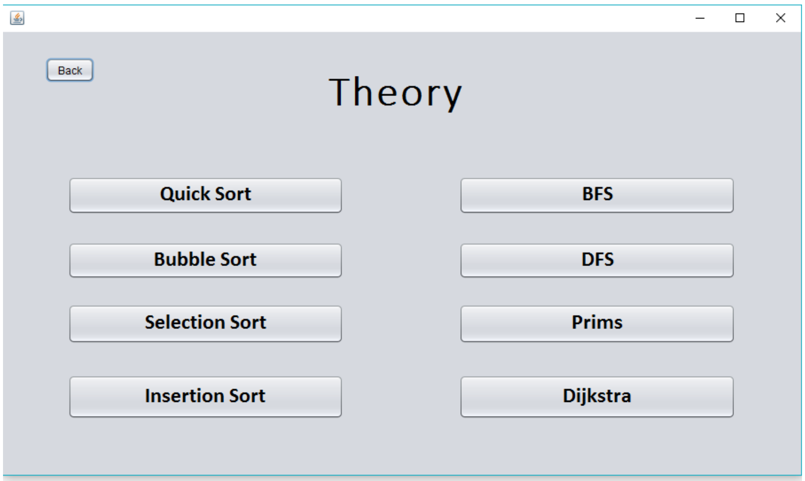


Figure 3.10

`

3. Results and Implementation

4.1 Implementation Details

The developer of the software needs to have a thorough knowledge of the programing methodologies and paradigms. The algorithms are written in java for which a GUI is displayed. The actual time taken to swap two numbers in the application is less compared to the time taken to show the movement of circles and bars used in the programs. This is because a small delay is added to actually follow the algorithm and understand it while the dynamic animation is going on. There are multiple pages in the application which are linked to each other by storing them in the same package. To dynamically call class during the run time the function is called on the fly by calling its class name followed by the function name.

The JFrame Panel is used to add GUI to the application. To create a fast responsive design of the template the user need to know the working of the Net Beans IDE. A simple JFrame Panel is used to add buttons, text areas, JLabels, etc.

4.2 Results and Evaluation

The application aims at making users learn a number of sorting as well as graphs algorithms. It provides with showing the actual working of sorting algorithms to sort a group of numbers. The result is provided in two forms. One is circle form where every circle represents a number from the group of entered numbers and sorts it. The second working is in the form of bars where length of each bar is represented by the weight of the bar i.e. the number the bar represents. Thus, making it easy to visualize the working of sorts. At the end after sorting, the program displays the time taken to sort the numbers along with the number of swaps done for sorting. Graphs help to determine how the various graphs algorithms work eg. Bfs, dfs, Dijkstra and prims. It shows how the nodes are accessed in a tree graph in various algorithms. It takes the source and destination of the nodes and shows the working of graph algorithms to reach to the destination from the source.

There are also some quiz questions kept for the learners to test what they learnt based on the algorithms present in the applications learn module. There is also a part of theory for each of the algorithms present which displays the algorithm pseudo-code as well as the time complexities of the algorithms.

`

Appendix

Sorting Algorithms

A. Calling method for sorting algorithms

```
call()
    {
               Stage subStage = new Stage();
               subStage.setTitle("New Stage");
       rootGroup = new Group();
               FlowPane root = new FlowPane();
               root.setAlignment(Pos.CENTER);
               Scene scene = new Scene(rootGroup, 900, 500, Color.GHOSTWHITE);
               Button btn=new Button("Back");
       btn.setLayoutX(700);
               btn.setLayoutY(400);
               rootGroup.getChildren().add(btn);
               btn.setOnAction(eve-> {subStage.close();});
        root.getChildren().add(new Button("New Stage"));
        subStage.setScene(scene);
        subStage.show();
        if(sort.test==1)
            selection(rootGroup);
        else if(sort.test==2)
            insertion(rootGroup);
        else if(sort.test==3)
            bubble(rootGroup);
        else
            quick(rootGroup);

    }
```

`

B. Insertion sort

```
void insertion(final Group group, int[] nos)
   {
       for(int i=0;i<nos.length;i++)
       {
              circle[i]=new Circle(x,y,radius);
              group.getChildren().add(circle[i]);
       }
       int[] arr=nos;
       int[] temp1=new int[nos.length];
       for(int i=0;i<nos.length;i++)
       {
              temp1[i]=i+1;
       }
       int temp=0;
              int swap=0;
       long tex;
              long t1=System.currentTimeMillis();
       for(int i=1; i < nos.length; i++)
       {
                for(int j=i; j > 0; j--)
              {
                            if(arr[j] < arr[j-1])
                     {
                            final Path path = generatePath(circle[j],circle[j-1]);
                            generatePathTransition(circle[temp1[j-1]-1],path1);
                            generatePathTransition(text[temp1[j-1]-1],path1);

                                          swap++;
                                          //swap elements
                                          temp = arr[j-1];
                                          arr[j-1] = arr[j];
                                          arr[j] = temp;

                            temp=temp1[j-1];
                            temp1[j-1]=temp1[j];
                            temp1[j]=temp;
                     }
                      }
              }
               long t2=System.currentTimeMillis();
              tex=t2-t1;
       Text time=new Text(1001,0,"Time taken for execution: "+tex+" milliseconds. \nNo of
       swaps : "+swap);
`
                             time.setStyle("-fx-font-size:20px;-fx-color:RED;");
                             time.setFont(Font.font("Comic-Sans", FontWeight.BOLD, 0));
                             group.getChildren().add(time);

                             final Path pathcurrent=new Path();
                             pathcurrent.getElements().add(new MoveTo(450,50));
                             pathcurrent.getElements().add(new LineTo(451,50));
                             generatePathTransition(end,pathcurrent);

                             final Path pathcurrent1=new Path();
                             pathcurrent1.getElements().add(new MoveTo(450,400));
                             pathcurrent1.getElements().add(new LineTo(451,400));
                             generatePathTransition(time,pathcurrent1);
   }
```

`

C. Selection Sort

```
void selection(final Group group, int[] nos)
   {
       for(int i=0;i<nos.length;i++)
       {
              circle[i]=new Circle(x,y,radius);
              group.getChildren().add(circle[i]);
       }
       int[] arr=nos;
       int[] temp1=new int[nos.length];
       for(int i=0;i<nos.length;i++)
       {
              temp1[i]=i+1;
       }
       int temp=0;
              int swap=0;
       long tex;
              long t1=System.currentTimeMillis();
       for(int i=0; i < nos.length-1; i++)
       {
              int index=i;
              int test=0;
                     for(int j=i+1; j < nos.length; j++)
              {
                                   if(arr[j] < arr[index])
                     {
                                          index=j;
                                          test=1;
                                   }
                     }
                             if(test==1)
                     {
                            final Path path1 = generatePath(circle[i],circle[j]);
                            generatePathTransition(circle[temp1[i]-1],path1);
                            generatePathTransition(text[temp1[i]-1],path1);

                                          swap++;

                                          //swap elements
                                          temp = arr[index];
                                          arr[index] = arr[i];
                                          arr[i] = temp;

                            temp=temp1[index];
`
                            temp1[index]=temp1[i];
                            temp1[i]=temp;
                     }
                     long t2=System.currentTimeMillis();
                     tex=t2-t1;
                       Text time=new Text(1001,0,"Time taken for execution: "+tex+"
milliseconds. \nNo of swaps : "+swap);
                                 time.setStyle("-fx-font-size:20px;-fx-color:RED;");
                                 time.setFont(Font.font("Comic-Sans", FontWeight.BOLD, 0));
                                 group.getChildren().add(time);

                                 final Path pathcurrent=new Path();
                                 pathcurrent.getElements().add(new MoveTo(450,50));
                                 pathcurrent.getElements().add(new LineTo(451,50));
                                 generatePathTransition(end,pathcurrent);

                                 final Path pathcurrent1=new Path();
                                 pathcurrent1.getElements().add(new MoveTo(450,400));
                                 pathcurrent1.getElements().add(new LineTo(451,400));
                                 generatePathTransition(time,pathcurrent1);
       }
}
```

`

D. Bubble Sort

```
void bubble(final Group group, int[] nos)
   {
       for(int i=0;i<nos.length;i++)
       {
               circle[i]=new Circle(x,y,radius);
               group.getChildren().add(circle[i]);
       }

       int[] arr=nos;
       int[] temp1=new int[nos.length];
       for(int i=0;i<nos.length;i++)
       {
               temp1[i]=i+1;
       }

       int temp=0;
               int swap=0;
       long tex;
               long t1=System.currentTimeMillis();

       for(int i=0; i < nos.length; i++)
       {
                 for(int j=1; j < (nos.length-i); j++)
               {
                                    if(arr[j] < arr[j-1])
                      {
                             final Path path = generatePath(circle[j],circle[j-1]);
                             generatePathTransition(circle[temp1[j-1]-1],path1);
                             generatePathTransition(text[temp1[j-1]-1],path1);

                                            swap++;

                                            //swap elements
                                            temp = arr[j-1];
                                            arr[j-1] = arr[j];
                                            arr[j] = temp;

                             temp=temp1[j-1];
                             temp1[j-1]=temp1[j];
                             temp1[j]=temp;
                      }
                       }
               }
```

`

```
                 long t2=System.currentTimeMillis();
                tex=t2-t1;

                                 Text time=new Text(1001,0,"Time taken for execution: 
"+tex+" milliseconds. \nNo of swaps : "+swap);

                                 time.setStyle("-fx-font-size:20px;-fx-color:RED;");
                                 time.setFont(Font.font("Comic-Sans", FontWeight.BOLD, 0));
                                 group.getChildren().add(time);

                                 final Path pathcurrent=new Path();
                                 pathcurrent.getElements().add(new MoveTo(450,50));
                                 pathcurrent.getElements().add(new LineTo(451,50));
                                 generatePathTransition(end,pathcurrent);

                                 final Path pathcurrent1=new Path();
                                 pathcurrent1.getElements().add(new MoveTo(450,400));
                                 pathcurrent1.getElements().add(new LineTo(451,400));
                                 generatePathTransition(time,pathcurrent1);
   }
```

`

E. Quick Sort

```
void quicksort(int arr[],int temp1[],int low,int high)
{
        int i = low, j = high;
        int temp;
        int pivot = arr[(low + high) / 2];
        while (i <= j)
        {
               while (arr[i] < pivot)
                              i++;
               while (arr[j] > pivot)
                               j--;
               if (i <= j)
               {
                              if(!(arr[i]==arr[j]))
                              {
                                     final Path path1 = 
generateCurvyPath(circle[i],circle[j]);
                                     generatePathTransition(circle[temp1[i]-1],path1);
                              generatePathTransition(text[temp1[i]-1],path1);
                                     swap1++;
                              /** swap **/
                              temp = arr[i];
                              arr[i] = arr[j];
                              arr[j] = temp;

                              temp=temp1[i];
                       temp1[i]=temp1[j];
                       temp1[j]=temp;
                              }
                       i++;
                       j--;
               }
        }
        /** recursively sort lower half **/
        if (low < j)
            quicksort(arr,temp1, low, j);
        /** recursively sort upper half **/
        if (i < high)
            quicksort(arr, temp1,i, high);
   }

void quick(final Group group, int[] nos)
   {
```

`

```
        for(int i=0;i<nos.length;i++)
        {
                circle[i]=new Circle(x,y,radius);
                group.getChildren().add(circle[i]);
        }
        int[] arr=nos;
        int[] temp1=new int[nos.length];
        int temp=0;
                int swap=0;
        long tex;
                long t1=System.currentTimeMillis();
        quicksort(nos,temp1,0,nos.length-1);
                long t2=System.currentTimeMillis();
                tex=t2-t1;
                                   Text time=new Text(1001,0,"Time taken for execution: 
"+tex+" milliseconds. \nNo of swaps : "+swap1);
                                   time.setStyle("-fx-font-size:20px;-fx-color:RED;");
                                   time.setFont(Font.font("Comic-Sans", FontWeight.BOLD, 0));
                                   group.getChildren().add(time);

                                   final Path pathcurrent=new Path();
                                   pathcurrent.getElements().add(new MoveTo(450,50));
                                   pathcurrent.getElements().add(new LineTo(451,50));
                                   generatePathTransition(end,pathcurrent);
}
```

`

Graphs

F. BFS

```
public void bfs_code(int src)
        {
            int p,i;
            add(src);
            vis[src]=1;
            p=delete();
            while(p!=-1)
            {
                for(i=0;i<nodes;i++)
                    if((dist[p][i]!=0)&&(vis[i]==0))
                    {
                             add(i);
                             Path path1 = createPath(circle[p],circle[i]);
                             Animation animation = createPathAnimation(path1, 
Duration.seconds(2));
                             animation.play();
                             vis[i]=1;
                    }
                      p=delete();
            }
            for(i=0;i<nodes;i++)
                if(vis[i]==0)
            bfs_code(i);
        }
```

`

G. DFS

```
void dfs_code(int src)
        {
            int p,i;
            push(src);
            vis[src]=1;
            p=pop();
                for(i=0;i<nodes;i++)
                {
                    if((dist[src][i]!=0)&&(vis[i]==0))
                    {
                        push(i);
                        Path path1 = createPath(circle[src],circle[i]);
                        Animation animation = createPathAnimation(path1,
Duration.seconds(2));
                      animation.play();
                        vis[i]=1;
                        p=pop();
                        dfs_code(p);
                    }
                }

        }
```

`

H. Dijkstra

```
void dijkstra(int src)
{
    int dist[]=new int[nodes];  // The output array. dist[i] will hold
                  // the shortest distance from src to i

    // sptSet[i] will true if vertex i is included / in shortest
    // path tree or shortest distance from src to i is finalized
    boolean sptSet[]=new boolean[nodes];

    // Parent array to store shortest path tree
    int parent[]=new int[nodes];
    // Initialize all distances as INFINITE and stpSet[] as false
    for (int i = 0; i < nodes; i++)
    {
        parent[i] = -1;
        dist[i] = max;
        sptSet[i] = false;
    }
    // Distance of source vertex from itself is always 0
    dist[src] = 0;
    // Find shortest path for all vertices
    for (int count = 0; count < nodes-1; count++)
    {
        // Pick the minimum distance vertex from the set of
        // vertices not yet processed. u is always equal to src
        // in first iteration.
        int u = minDistance(dist, sptSet);

        // Mark the picked vertex as processed
        sptSet[u] = true;
        for (int v = 0; v < nodes; v++)
            if (!sptSet[v] && dist1[u][v]!=0 && ((dist[u] + dist1[u][v]) < dist[v]))
            {
                parent[v]  = u;
                dist[v] = dist[u] + dist1[u][v];
            }
    }
       printPath(parent, end);
       temp();
}
    void printPath(int parent[], int j)
{
    // Base Case : If j is source
`
    if (parent[j]==-1)
    {
        //q[q1++]=j;
        return;
    }
    printPath(parent, parent[j]);
    q[q1++]=j;
}

void temp()
 {
     drawcolorline(circle[start],circle[q[0]]);
     System.out.println("outside" +q1);
     for(int i=q2;i<q1-1;i++)
     {
         System.out.println("inside");
         drawcolorline(circle[q[i]],circle[q[i+1]]);
     }
 }
```

`

I. Prims

```
public void prims_code()
        {
        for(int i=0;i<nodes;i++)
        {
            circle[i]=new Circle(x,y,],radius);
             root.getChildren().add(circle[i]);
             circle[i].setFill(Color.LIGHTGREEN);
        }
               int empty=is_empty();
               vis[start]=1;
               while(empty==0)
               {
                      int min_index=cal_min(start);
                             vis[min_index]=1;
                      Path path1=createPath(circle[src],circle[min_index);
                             Animation animation = createPathAnimation(path1,
Duration.seconds(2));
                             animation.play();
                        int new_start=0;
                        int new_dist=10000;
                        for(int i=0;i<nodes;i++)
                        {
                            if(vis[i]==1)
                            {
                               if(new_dist>dist[i][cal_min(i)])
                               {
                                   new_dist=dist[i][cal_min(i)];
                                   new_start=i;
                               }
                            }
                        }
                        start=new_start;
                        empty=is_empty();
               }
        }
       int cal_min(int num)
       {
               int min_index=0;
               int min_dist=10000;
               for(int i=0;i<nodes;i++)
               {
                      if(i!=num && vis[i]==0 && dist[num][i]!=0)
                      {
```

`

```
            if(dist[num][i]<min_dist)
            {
                min_dist=dist[num][i];
                min_index=i;
            }
        }
    }
    return min_index;
}

int is_empty()
{
    int flag=0;
    for(int i=0;i<nodes;i++)
        if(vis[i]==0)
        {
            flag=1;
            break;
        }
    if(flag==0)
        return 1;
    else
        return 0;
}
```

`

J. Theory

```
public class Bubble_theory extends javax.swing.JFrame {

    public Bubble_theory() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jLabel2 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();
        jLabel5 = new javax.swing.JLabel();
        jLabel6 = new javax.swing.JLabel();
        jLabel7 = new javax.swing.JLabel();
        jButton1 = new javax.swing.JButton();
        jScrollPane1 = new javax.swing.JScrollPane();
        jTextArea1 = new javax.swing.JTextArea();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);

        jLabel1.setFont(new java.awt.Font("Calibri Light", 1, 36)); // NOI18N
        jLabel1.setText("Bubble Sort");

        jLabel2.setFont(new java.awt.Font("Calibri", 0, 12)); // NOI18N
        jLabel2.setText("Algorithm :");

        jLabel4.setText("Time Complexity :");

        jLabel5.setText("Space Complexity :");

        jLabel6.setText("O(n)");

        jLabel7.setText("O(1)");

        jButton1.setText("Back");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
```

```
`
        jScrollPane1.setFont(new java.awt.Font("Calibri", 0, 12)); // NOI18N

        jTextArea1.setEditable(false);
        jTextArea1.setColumns(20);
        jTextArea1.setRows(5);
        jTextArea1.setText("procedure bubbleSort( list : array of items )\n  loop = 
list.count; \n   for i = 0 to loop-1 do:\n        swapped = false\t\n   for j = 0 to loop-1 
do:  \n      if list[j] > list[j+1] then\n        swap( list[j], list[j+1] )\t\t \n        
swapped = true\n        end if\n      end for\n      if(not swapped) then\n         break\n      
end if\n   end for   \nend procedure return list ");
        jScrollPane1.setViewportView(jTextArea1);

        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(65, 65, 65)
                
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jButton1)
                        .addGap(201, 201, 201)
                        .addComponent(jLabel1))
                    .addGroup(layout.createSequentialGroup()
                        
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel2)
                            .addComponent(jLabel4)
                            .addComponent(jLabel5))
                        .addGap(44, 44, 44)
                        
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                            .addComponent(jLabel7)
                            .addComponent(jLabel6)
                            .addComponent(jScrollPane1, 
javax.swing.GroupLayout.PREFERRED_SIZE, 469, javax.swing.GroupLayout.PREFERRED_SIZE))))
                .addContainerGap(230, Short.MAX_VALUE))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(33, 33, 33)
`
                        .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 58,
javax.swing.GroupLayout.PREFERRED_SIZE))
                    .addGroup(layout.createSequentialGroup()
                        .addContainerGap()
                        .addComponent(jButton1)))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel2)
                    .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
295, javax.swing.GroupLayout.PREFERRED_SIZE))
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 26,
Short.MAX_VALUE)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .addComponent(jLabel6))
                .addGap(18, 18, 18)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel5)
                    .addComponent(jLabel7))
                .addGap(24, 24, 24))
        );

        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>

    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new Theory();
        dispose();
        // TODO add your handling code here:
    }

    public static void al() {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
                    javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex) {
```

`

```
java.util.logging.Logger.getLogger(Bubble_theory.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Bubble_theory.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Bubble_theory.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Bubble_theory.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Bubble_theory().setVisible(true);
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JButton jButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel2;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JLabel jLabel5;
    private javax.swing.JLabel jLabel6;
    private javax.swing.JLabel jLabel7;
    private javax.swing.JScrollPane jScrollPane1;
    private javax.swing.JTextArea jTextArea1;
    // End of variables declaration
}
```

`

K. QUIZ

```
public class bs_quiz2 extends javax.swing.JFrame {
    public bs_quiz2() {
        initComponents();
    }
    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        buttonGroup1 = new javax.swing.ButtonGroup();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jLabel3 = new javax.swing.JLabel();
        jRadioButton1 = new javax.swing.JRadioButton();
        jRadioButton2 = new javax.swing.JRadioButton();
        jRadioButton3 = new javax.swing.JRadioButton();
        jRadioButton4 = new javax.swing.JRadioButton();
        jButton1 = new javax.swing.JButton();
        ans_lbl = new javax.swing.JLabel();
        jButton2 = new javax.swing.JButton();
        jLabel1 = new javax.swing.JLabel();
        jLabel4 = new javax.swing.JLabel();

        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
        setCursor(new java.awt.Cursor(java.awt.Cursor.DEFAULT_CURSOR));
        setResizable(false);
        jLabel3.setFont(new java.awt.Font("Times New Roman", 1, 24)); // NOI18N
        jLabel3.setText("Ans:");

        buttonGroup1.add(jRadioButton1);
        jRadioButton1.setFont(new java.awt.Font("Trebuchet MS", 0, 18)); // NOI18N
        jRadioButton1.setText("(A)N^2");
        buttonGroup1.add(jRadioButton2);
        jRadioButton2.setFont(new java.awt.Font("Trebuchet MS", 0, 18)); // NOI18N
        jRadioButton2.setText("(B)NlogN");
        jRadioButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jRadioButton2ActionPerformed(evt);
            }
        });
        buttonGroup1.add(jRadioButton3);
        jRadioButton3.setFont(new java.awt.Font("Trebuchet MS", 0, 18)); // NOI18N
        jRadioButton3.setText("(C)N");
        jRadioButton3.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
```

`

```
                jRadioButton3ActionPerformed(evt);
            }
        });
        buttonGroup1.add(jRadioButton4);
        jRadioButton4.setFont(new java.awt.Font("Trebuchet MS", 0, 18)); // NOI18N
        jRadioButton4.setText("(D)N(logN)^2");
        jButton1.setFont(new java.awt.Font("Times New Roman", 0, 18)); // NOI18N
        jButton1.setText("Next");
        jButton1.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton1ActionPerformed(evt);
            }
        });
        ans_lbl.setFont(new java.awt.Font("Times New Roman", 1, 18)); // NOI18N
        ans_lbl.setBorder(new javax.swing.border.LineBorder(new java.awt.Color(255, 51,
51), 3, true));

        jButton2.setFont(new java.awt.Font("Times New Roman", 0, 18)); // NOI18N
        jButton2.setText("Submit");
        jButton2.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {
                jButton2ActionPerformed(evt);
            }
        });
        jLabel1.setFont(new java.awt.Font("Trebuchet MS", 0, 18)); // NOI18N
        jLabel1.setText("<html>1.What is the best time complexity of bubble sort?</html>");
        jLabel4.setFont(new java.awt.Font("Century Gothic", 3, 36)); // NOI18N
        jLabel4.setText("SORTING QUIZ");
        javax.swing.GroupLayout layout = new javax.swing.GroupLayout(getContentPane());
        getContentPane().setLayout(layout);
        layout.setHorizontalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(38, 38, 38)
                .addComponent(jButton2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jButton1)
                .addGap(23, 23, 23))
            .addGroup(layout.createSequentialGroup()
                .addGap(89, 89, 89)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jRadioButton4)
                    .addComponent(jRadioButton3)
                    .addComponent(jRadioButton2)
```

`

```
                    .addComponent(jRadioButton1)
                    .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 560,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(layout.createSequentialGroup()
                        .addGap(100, 100, 100)
                        .addComponent(jLabel3)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                        .addComponent(ans_lbl, javax.swing.GroupLayout.PREFERRED_SIZE, 340,
javax.swing.GroupLayout.PREFERRED_SIZE)))
                .addContainerGap(207, Short.MAX_VALUE))
            .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
layout.createSequentialGroup()
                .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
                .addComponent(jLabel4)
                .addGap(302, 302, 302))
        );
        layout.setVerticalGroup(
            layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .addGroup(layout.createSequentialGroup()
                .addGap(25, 25, 25)
                .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE, 38,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 91,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addGap(18, 18, 18)
                .addComponent(jRadioButton1)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jRadioButton2)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
                .addComponent(jRadioButton3)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(jRadioButton4)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 60,
Short.MAX_VALUE)
          .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jLabel3)
                    .addComponent(ans_lbl, javax.swing.GroupLayout.PREFERRED_SIZE, 37,
javax.swing.GroupLayout.PREFERRED_SIZE))
                .addGap(53, 53, 53)

.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jButton1)
                    .addComponent(jButton2))
                .addGap(22, 22, 22))
```

```
`
        );
        pack();
        setLocationRelativeTo(null);
    }// </editor-fold>
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        new bs_quiz().q2();
    dispose();
    }
    private void jRadioButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        // TODO add your handling code here:
    }
    private void jRadioButton3ActionPerformed(java.awt.event.ActionEvent evt) {
            // TODO add your handling code here:
    }
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
        if(jRadioButton3.isSelected())
            ans_lbl.setText(" Your answer is Correct!");
        else
            ans_lbl.setText("  Oops!  The correct answer is C");
// TODO add your handling code here:
    }
    public static void q2() {
        try {
            for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
                if ("Nimbus".equals(info.getName())) {
javax.swing.UIManager.setLookAndFeel(info.getClassName());
                    break;
                }
            }
        } catch (ClassNotFoundException ex)
{java.util.logging.Logger.getLogger(bs_quiz2.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (InstantiationException ex)
{java.util.logging.Logger.getLogger(bs_quiz2.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (IllegalAccessException ex)
{java.util.logging.Logger.getLogger(bs_quiz2.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        } catch (javax.swing.UnsupportedLookAndFeelException ex)
{java.util.logging.Logger.getLogger(bs_quiz2.class.getName()).log(java.util.logging.Level.S
EVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new bs_quiz2().setVisible(true);
`
            }
        });
    }
    // Variables declaration - do not modify
    private javax.swing.JLabel ans_lbl;
    private javax.swing.ButtonGroup buttonGroup1;
    private javax.swing.JButton jButton1;
    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JLabel jLabel3;
    private javax.swing.JLabel jLabel4;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JRadioButton jRadioButton3;
    private javax.swing.JRadioButton jRadioButton4;
    private javax.swing.JTabbedPane jTabbedPane1;
    // End of variables declaration
}
```

`

Conclusion and Future Work

With the help of this application built on Java FX and Java graphical libraries we were able to provide user the GUI interface through which it could interact with the application. The user was able to learn different sorting algorithms and Graph Traversal methods, test its knowledge through the quiz module of the application and do the comparative study of different sorting algorithms based on time and space complexity.

The idea of mapping each of every step of the algorithm while running the output was not achieved in this application due to time constraints.

In future we would like to expand this application on larger scale by adding more sorting algorithms, new search techniques and more graph traversal methods. In addition to this we would also like to add voice over outputs which could benefit for deaf and dumb users and enhance GUI interface to improve its efficiency and performance.

`

References

[1] Ortiz, C. Enrique; Giguere, Eric, “Mobile Information Device Profile for Java 2”, Micro Edition: Developer's Guide, John Wiley & Sons, 2012.

[2] "Oracle JDK 7 and JRE 7 Certified System Configurations”, Oracle.com, 2016.

[3] R. Chapman, Java applet, awt and util class reference, 1st ed. Seattle, WA: Specialized Systems Consultants, 1996.

[4] R. Chapman, *Java applet, awt and util class reference,* 1st ed. Seattle, WA: Specialized Systems Consultants, 1996.

[5] H. Ebbers, *Mastering JavaFX 8 Controls*, 1st ed. New York: McGraw-Hill Education, 2014.

[6] Paul Anderson. Gail Anderson., *JavaFX Rich Client Programming on the NetBeans Platform*, 1st ed. Addison-Wesley Professional, 2014.

[7] J. Zukowski, *The definitive guide to Java Swing*, 1st ed. Berkeley, CA: Apress, 2005.

[8] "JavaFX Application", www.tutorialspoint.com, 2017. [Online]. Available: https://www.tutorialspoint.com/javafx/javafx_application.htm. [Accessed: 30- Apr- 2017].

[9] "PathTransition (JavaFX 2.2)", Docs.oracle.com, 2017. [Online]. Available: https://docs.oracle.com/javafx/2/api/javafx/animation/PathTransition.html. [Accessed: 30- Apr- 2017].

[10] Tutorial, "JavaFx-Creating Multiple Stage - JavaFx Tutorial", JavaFx Tutorial, 2017. [Online]. Available: http://hajsoftutorial.com/javafx-tutorial/javafx-creating-multiple-stage/. [Accessed: 30- Apr- 2017].

[11] "CSS Tutorial", *W3schools.com*, 2017. [Online]. Available: https://www.w3schools.com/css/. [Accessed: 30- Apr- 2017].

`

Acknowledgement

We would like to express our gratitude towards Dr. G.T. THAMPI for allowing us to narrow down an interesting topic and helping us. He has been the much needed driving force for us. His continuous guidance and suggestions helped us to coordinate well and enhanced our scope and knowledge for the project.

We would also like to thank Dr. MADHURI RAO, Head of Department, Information Technology Department, Thadomal Shahani Engineering College for the support and encouragement and also for providing us with the necessary infrastructure and facilities in this college.

We would also like to thank our faculty members and non-teaching staff of Thadomal Shahani Engineering College for giving us their valuable time and patience.