

B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



Course – JAVA PROGRAMMING
Course Code – 19IS4PCDJAV
AY 2019-20

Final Report on Project work

SIMULATION OF SORTING ALGORITHMS

Submitted to – Mamtha M
(Associate Professor)

Submitted by –

Prateek Jain
1BM18IS057

Anupam Kumar Boruah
1BM18IS013

B.M.S College of Engineering
P.O. Box No.: 1908 Bull Temple Road,
Bangalore-560 019

DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING



CERTIFICATE

Certified that the Project has been successfully presented at **B.M.S College Of Engineering** by **PRATEEK JAIN, ANUPAM KUMAR BORUAH** bearing **USN: 1BM18IS149, 1BM18IS013**, in partial fulfillment of the requirements for the IV Semester degree in **Bachelor of Engineering in Information Science & Engineering** of **Visvesvaraya Technological University, Belgaum** as a part of the course **JAVA programming (19IS4PCJAV)** during academic year 2019-2020.

Faculty Name – Mamtha M.

Designation – ASSISTANT PROFESSOR

Department of ISE, BMSCE

TABLE OF CONTENTS

Contents	Page No
Front Cover	1
Certificate	2
Table of Contents	3
Abstract	4
Introduction	4
Problem Statement	4
Implementation/Code	5
Result	30
Java Concepts Covered	33
Tools Used	33
High Level Design	34
References	35

ABSTRACT:

Java programming is used to create animation to visualize and simulate two sorting algorithms: Bubble Sort and Quick Sort. Time Complexity of both the algorithms have been discussed. — Sorting is one of the most fundamental problems in computer science, as it is used in most software applications. Data-driven algorithms especially use sorting to gain an efficient access to data. Many sorting algorithms with distinct properties for different architectures have been developed. Searching for items and sorting through items are tasks that we do every day.

INTRODUCTION:

Sorting a list of items into ascending or descending order can help either a human or a computer find items on that list quickly. This is implemented using sorting algorithms. Searching for items and sorting through items are tasks that we do every day. Searching and sorting are also common tasks in computer programs. We search for all occurrences of a word in a file in order to replace it with another word. We sort the items on a list into alphabetical or numerical order. Because searching and sorting are common computer tasks, we have well-known algorithms, for doing searching and sorting

There are several sorting algorithms. The two main criteria to judge which algorithm is better than the other have been:

1. Time taken to sort the given data.
2. Memory Space required to do so.

PROBLEM STATEMENT:

Simulation of two sorting techniques: Bubble Sort and Quick Sort and comparison between them.

The number of sorting algorithms considered in this study for complexity measurement is limited to bubble, insertion, and selection sorting.

The goal for this will be to adopt the most efficient sorting technique in the development of job scheduler for grid computing community.

API's used in the Project:

`java.util.Random` – is used to create random numbers.

`javax.swing.JButton` – is used to create buttons in `JFrame`.

`javax.swing.JComboBox` – is used to create combo box in `JFrame`.

`javax.swing.JOptionPane` – is used to create option pane in `JFrame`.

`javax.swing.JTextField` – is used to create in text field `JFrame`.

IMPLEMENTATION/CODE:

Bubble Sort:

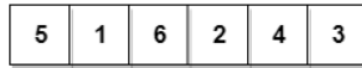
Following are the steps involved in bubble sort(for sorting a given array in ascending order):

1. Starting with the first element(index = 0), compare the current element with the next element of the array.
2. If the current element is greater than the next element of the array, swap them.
3. If the current element is less than the next element, move to the next element. Repeat Step 1.

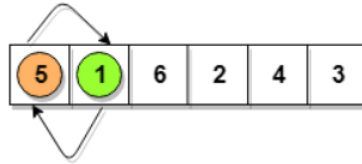
Let's consider an array with values {5, 1, 6, 2, 4, 3}

Below, we have a pictorial representation of how bubble sort will sort the given array.

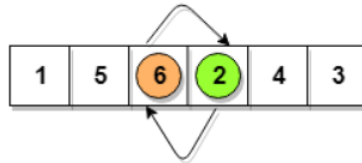
5>1
so interchange



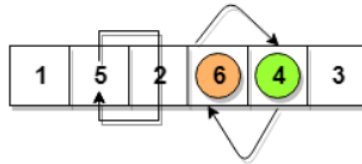
5<6
No swapping



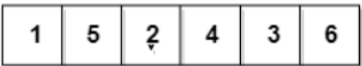
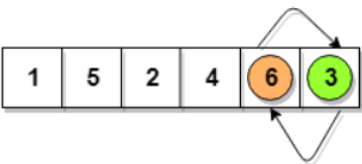
6>2
so interchange



6>4
so interchange



6>3
so interchange



This is first insertion

similarly, after all the iterations, the array gets sorted

Following are the Time and Space complexity for the Bubble Sort algorithm.

- Worst Case Time Complexity [Big-O]: $O(n^2)$
- Best Case Time Complexity [Big-omega]: $O(n)$
- Average Time Complexity [Big-theta]: $O(n^2)$
- Space Complexity: $O(1)$

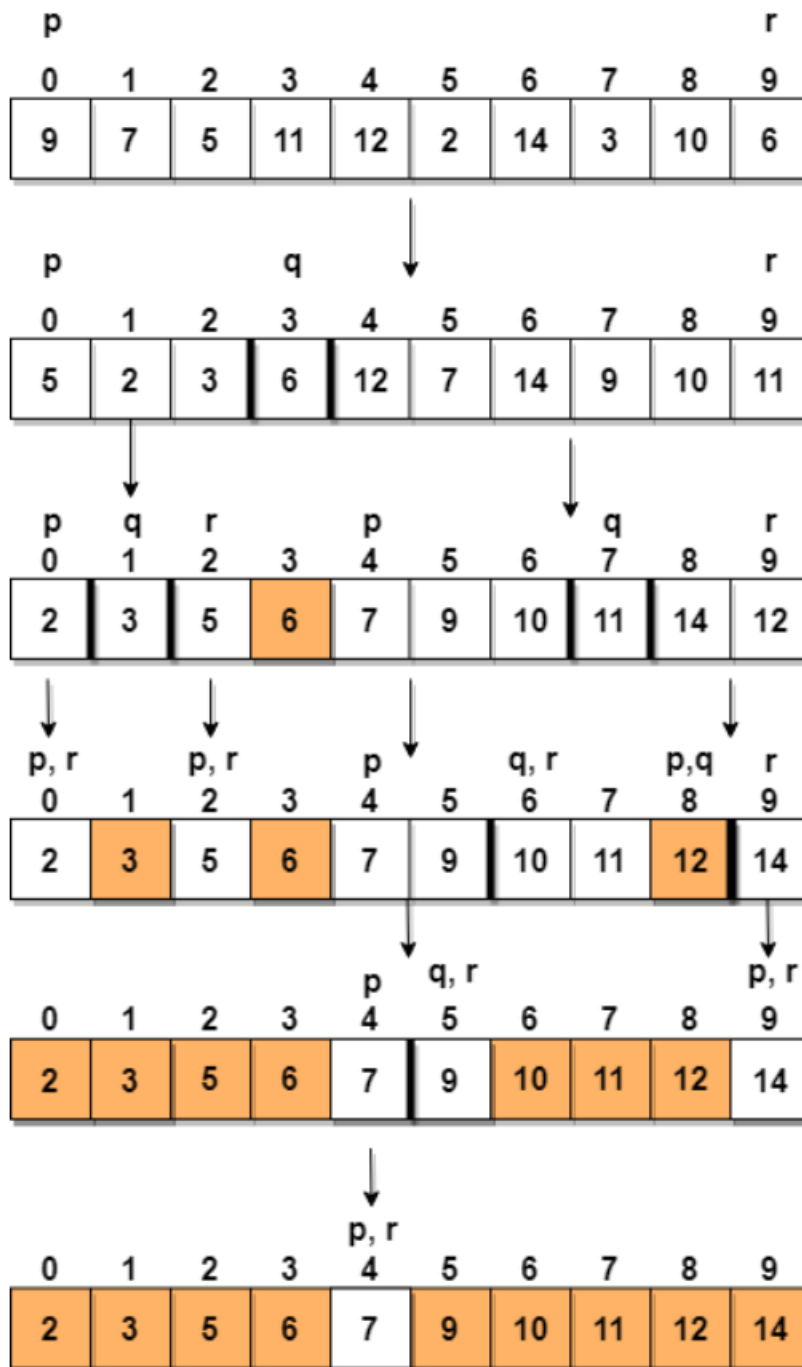
Quick Sort:

Following are the steps involved in quick sort algorithm:

1. After selecting an element as pivot, which is the last index of the array in our case, we divide the array for the first time.
2. In quick sort, we call this partitioning. It is not simple breaking down of array into 2 subarrays, but in case of partitioning, the array elements are so positioned that all the elements smaller than the pivot will be on the left side of the pivot and all the elements greater than the pivot will be on the right side of it.
3. And the pivot element will be at its final sorted position.
4. The elements to the left and right, may not be sorted.
5. Then we pick subarrays, elements on the left of pivot and elements on the right of pivot, and we perform partitioning on them by choosing a pivot in the subarrays.

Let's consider an array with values {9, 7, 5, 11, 12, 2, 14, 3, 10, 6}

Below, we have a pictorial representation of how quick sort will sort the given array.



Following are the Time and Space complexity for the Quick Sort algorithm:

Worst Case Time Complexity [Big-O]: $O(n^2)$

Best Case Time Complexity [Big-omega]: $O(n \log n)$

Average Time Complexity [Big-theta]: $O(n \log n)$

Space Complexity: $O(n \log n)$

Code:

```
import java.util.Random;
import javax.swing.JButton;
import javax.swing.JOptionPane;

/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

/**
 *
 * @author NIKHIL
 */
public class simulation extends javax.swing.JFrame implements Runnable {

    Thread t;

    JButton[] button_array;

    int[] A;
```

```
int choice;
```

```
/**
```

```
 * Creates new form simulation
```

```
 */
```

```
public simulation() {
```

```
    initComponents();
```

```
    t = new Thread(this);
```

```
    jLabel2.setVisible(false);
```

```
    jLabel1.setVisible(false);
```

```
    jButton2.setVisible(false);
```

```
    jButton3.setVisible(false);
```

```
    jButton4.setVisible(false);
```

```
    jButton5.setVisible(false);
```

```
    jButton6.setVisible(false);
```

```
    jButton7.setVisible(false);
```

```
}
```

```
/**
```

```
 * This method is called from within the constructor to initialize the form.
```

```
 * WARNING: Do NOT modify this code. The content of this method is always
```

```
 * regenerated by the Form Editor.
```

```
 */
```

```
@SuppressWarnings("unchecked")
```

```
// <editor-fold defaultstate="collapsed" desc="Generated Code">
```

```
private void initComponents() {
```

```

jButton1 = new javax.swing.JButton();
jTextField1 = new javax.swing.JTextField();
jTextField2 = new javax.swing.JTextField();
jTextField3 = new javax.swing.JTextField();
jTextField4 = new javax.swing.JTextField();
jTextField5 = new javax.swing.JTextField();
jTextField6 = new javax.swing.JTextField();
jButton2 = new javax.swing.JButton();
jButton3 = new javax.swing.JButton();
jButton4 = new javax.swing.JButton();
jButton5 = new javax.swing.JButton();
jButton6 = new javax.swing.JButton();
jButton7 = new javax.swing.JButton();
jComboBox1 = new javax.swing.JComboBox();
jButton8 = new javax.swing.JButton();
jTextField7 = new javax.swing.JTextField();
jLabel1 = new javax.swing.JLabel();
jLabel2 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

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

jTextField1.setMaximumSize(new java.awt.Dimension(79, 25));

```

```
textField1.setMinimumSize(new java.awt.Dimension(35, 22));

textField2.setMinimumSize(new java.awt.Dimension(35, 22));

textField3.setMinimumSize(new java.awt.Dimension(35, 22));

textField4.setMinimumSize(new java.awt.Dimension(35, 22));

textField5.setMinimumSize(new java.awt.Dimension(35, 22));

textField6.setMinimumSize(new java.awt.Dimension(35, 22));

button2.setMinimumSize(new java.awt.Dimension(35, 22));
button2.setPreferredSize(new java.awt.Dimension(50, 25));

button3.setMinimumSize(new java.awt.Dimension(35, 22));
button3.setPreferredSize(new java.awt.Dimension(50, 25));

button4.setMinimumSize(new java.awt.Dimension(35, 22));
button4.setPreferredSize(new java.awt.Dimension(50, 25));

button5.setMinimumSize(new java.awt.Dimension(35, 22));
button5.setPreferredSize(new java.awt.Dimension(50, 25));

button6.setMinimumSize(new java.awt.Dimension(35, 22));
button6.setPreferredSize(new java.awt.Dimension(50, 25));

button7.setMinimumSize(new java.awt.Dimension(35, 22));
button7.setPreferredSize(new java.awt.Dimension(50, 25));

comboBox1.setModel(new javax.swing.DefaultComboBoxModel(new String[] { "Select",
"Bubble Sort", "Quick Sort" }));

comboBox1.addActionListener(new java.awt.event.ActionListener() {
```

```

        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jComboBox1ActionPerformed(evt);
        }
    });

    jButton8.setText("Time Complexity");
    jButton8.addActionListener(new java.awt.event.ActionListener() {
        public void actionPerformed(java.awt.event.ActionEvent evt) {
            jButton8ActionPerformed(evt);
        }
    });

    jLabel1.setText("jLabel1");

    jLabel2.setText("jLabel2");

    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(33, 33, 33)
                .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                    .addComponent(jButton8)
                    .addGroup(layout.createSequentialGroup()
                        .addComponent(jTextField7, javax.swing.GroupLayout.PREFERRED_SIZE, 81,
                            javax.swing.GroupLayout.PREFERRED_SIZE)
                        .addGap(200, 200, 200))
                    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 59,
                        Short.MAX_VALUE))
                )
            )
    );

```

G)

```

        .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
DING)

        .addComponent(jLabel2)
        .addComponent(jLabel1))
        .addGap(0, 0, Short.MAX_VALUE))
        .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
DING, false)

        .addComponent(jButton2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jTextField1, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
DING, false)

        .addComponent(jTextField2, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jButton3, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
DING)

        .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGroup(layout.createSequentialGroup())
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
TRAILING)

        .addComponent(jButton1)
        .addGroup(layout.createSequentialGroup())

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING, false)

```

```

        .addComponent(jTextField3,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

        .addComponent(jButton4,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))

        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Align
ment.LEADING, false)

        .addComponent(jTextField4,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)

        .addComponent(jButton5,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))

        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)

        .addComponent(jTextField5, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jButton6, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))

        .addGap(18, 18, 18)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.
LEADING, false)

        .addComponent(jTextField6, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)

        .addComponent(jButton7, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))))

        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))))

    );

    layout.setVerticalGroup(

        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

        .addGroup(layout.createSequentialGroup()

```

```

        .addContainerGap()

        .addComponent(jComboBox1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

        .addGap(28, 28, 28)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jTextField1, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jTextField2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jTextField3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jTextField4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jTextField5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jTextField6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))

        .addGap(18, 18, 18)

        .addComponent(jButton1)

        .addGap(46, 46, 46)

        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)

            .addComponent(jButton2, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jButton3, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jButton4, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jButton5, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

            .addComponent(jButton6, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)

```



```

        .addComponent(jButton7, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(46, 46, 46)
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
NE)
        .addComponent(jButton8)
        .addComponent(jTextField7, javax.swing.GroupLayout.PREFERRED_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(18, 18, 18)
        .addComponent(jLabel1)
        .addGap(18, 18, 18)
        .addComponent(jLabel2)
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    );

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

```

```

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

    A = new int[6];

    if(jTextField1.getText().isEmpty()){
        JOptionPane.showMessageDialog(null, "Insert a number", "Warning", 2);
        jTextField1.grabFocus();
    }
    else if(jTextField2.getText().isEmpty()){
        JOptionPane.showMessageDialog(null, "Insert a number", "Warning", 2);
        jTextField2.grabFocus();
    }
}

```

```

}
else if(jTextField3.getText().isEmpty()){
    JOptionPane.showMessageDialog(null, "Insert a number", "Warning", 2);
    jTextField3.grabFocus();
}
else if(jTextField4.getText().isEmpty()){
    JOptionPane.showMessageDialog(null, "Insert a number", "Warning", 2);
    jTextField4.grabFocus();
}
else if(jTextField5.getText().isEmpty()){
    JOptionPane.showMessageDialog(null, "Insert a number", "Warning", 2);
    jTextField5.grabFocus();
}
else if(jTextField6.getText().isEmpty()){
    JOptionPane.showMessageDialog(null, "Insert a number", "Warning", 2);
    jTextField6.grabFocus();
}

else{
    A[0]=Integer.parseInt(jTextField1.getText());
    A[1]=Integer.parseInt(jTextField2.getText());
    A[2]=Integer.parseInt(jTextField3.getText());
    A[3]=Integer.parseInt(jTextField4.getText());
    A[4]=Integer.parseInt(jTextField5.getText());
    A[5]=Integer.parseInt(jTextField6.getText());

    for(int k=0; k<A.length; k++){System.out.println(A[k]);}
}

```

```

jButton2.setVisible(true);
jButton3.setVisible(true);
jButton4.setVisible(true);
jButton5.setVisible(true);
jButton6.setVisible(true);
jButton7.setVisible(true);

jButton2.setText(jTextField1.getText());
jButton3.setText(jTextField2.getText());
jButton4.setText(jTextField3.getText());
jButton5.setText(jTextField4.getText());
jButton6.setText(jTextField5.getText());
jButton7.setText(jTextField6.getText());

button_array = new JButton[]{jButton2,jButton3,jButton4,jButton5,jButton6,jButton7};

t.start();

}

}

private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:
    choice = jComboBox1.getSelectedIndex();
    System.out.println("choice = "+choice);
    System.out.println("Elements before Sorting");
}

```

```

private void jButton8ActionPerformed(java.awt.event.ActionEvent evt)
{
    // TODO add your handling code here:

    jButton2.setText(Integer.toString(A[0]));
    jButton3.setText(Integer.toString(A[1]));
    jButton4.setText(Integer.toString(A[2]));
    jButton5.setText(Integer.toString(A[3]));
    jButton6.setText(Integer.toString(A[4]));
    jButton7.setText(Integer.toString(A[5]));

    Random rd = new Random(); // creating Random object
    int no=Integer.parseInt(jTextField7.getText());
    jLabel2.setVisible(true);
    jLabel1.setVisible(true);
    jLabel1.setText("Generated " + no +" random numbers.");
    System.out.println("Generated " + no +" random numbers.");
    int[] arr = new int[no];

    for (int i = 0; i < arr.length; i++) {
        arr[i] = rd.nextInt();
    }
    if(choice==1){
        long startTime = System.currentTimeMillis();
        bubble(arr);
        long endTime = System.currentTimeMillis();

        long time = endTime - startTime;

        jLabel2.setText("Time taken to sort using Bubble Sort = "+time+" milliseconds");
    }
}

```

```
System.out.println("Time taken to sort using Bubble Sort = "+time+" milliseconds");
```

```
}
```

```
if(choice==2){
```

```
    long startTime = System.currentTimeMillis();
```

```
    qS(arr,0, arr.length-1);
```

```
    long endTime = System.currentTimeMillis();
```

```
long time = endTime - startTime;
```

```
System.out.println("Time taken to sort using Quick Sort = "+time+" milliseconds");
```

```
jLabel2.setText("Time taken to sort using Quick Sort = "+time+" milliseconds");
```

```
}
```

```
}
```

```
/**
```

```
 * @param args the command line arguments
```

```
 */
```

```
public static void main(String args[]) {
```

```
    /* Set the Nimbus look and feel */
```

```
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting code (optional) ">
```

```
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the default look and feel.
```

```
     * For details see http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
```

```
    */
```

```
    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(simulation.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (InstantiationException ex) {

    java.util.logging.Logger.getLogger(simulation.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (IllegalAccessException ex) {

    java.util.logging.Logger.getLogger(simulation.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

} catch (javax.swing.UnsupportedLookAndFeelException ex) {

    java.util.logging.Logger.getLogger(simulation.class.getName()).log(java.util.logging.Level.SEVERE, null, ex);

}

//</editor-fold>

/* Create and display the form */
java.awt.EventQueue.invokeLater(new Runnable() {
    public void run() {
        new simulation().setVisible(true);
    }
});
}

public void run(){

```

```

if(choice==1)
{

    System.out.println("Starting Bubble Sort");
    int i,j,n,temp;
    n= A.length;

    for(i=0; i<n; i++){

        for(j=0; j<(n-i); j++){
            try{ if(A[j] > A[j+1])
            {
                temp = A[j];
                A[j]=A[j+1];
                A[j+1]=temp;

                try{
                    swap2(j, j+1);
                }
                catch(InterruptedException e){ System.out.println("error");}
            }
        }
        catch(Exception e){//System.out.println("error-arrow...!!");}
    }
}

for(int k=0; k<A.length; k++){System.out.println(A[k]);}

```

```
}
```

```
if(choice == 2){  
    System.out.println("Starting Quick Sort");  
    quickSort(A,0,A.length-1);  
    for(int k=0; k<A.length; k++){System.out.println(A[k]);}  
}
```

```
}
```

```
public void bubble(int A[]){  
    int i,j,n,temp;  
    n= A.length-1;  
    for(i=0; i<n; i++){  
  
        for(j=0; j<(n-i); j++){  
            if(A[j] > A[j+1])  
            {  
                temp = A[j];  
                A[j]=A[j+1];  
                A[j+1]=temp;  
            }  
  
        }  
  
    }  
}
```



```

    }
}

public int parttt (int arr[], int low,int  high)
{
    // pivot (Element to be placed at right position)
    int pivot = arr[high];
    int temp;
    int i = (low - 1); // Index of smaller element

    for (int j = low; j <= high- 1; j++)
    {
        // If current element is smaller than the pivot
        if (arr[j] < pivot)
        {
            i++; // increment index of smaller element
            //swap arr[i] and arr[j]
            try{
                swap2(i, j);
            }
            catch(InterruptedExcepTion e){System.out.println("error");}

            temp = arr[i];
            arr[i]=arr[j];
            arr[j]=temp;

        }
    }

    //swap arr[i + 1] and arr[high])

```

```

try{
    swap2(i+1, high);
}
catch(InterruptedException e){System.out.println("error");}

temp = arr[i+1];
arr[i+1]=arr[high];
arr[high]=temp;

return (i + 1);
}

```

```

public void quickSort(int a[], int beg, int end)

```

```

{

    int loc;
    if(beg<end)
    {
        loc = parttt(a, beg, end);
        quickSort(a, beg, loc-1);
        quickSort(a, loc+1, end);
    }
}

```

```

public void qS(int a[], int beg, int end)

```

```

{

    int loc;
    if(beg<end)
    {

```

```

        loc = part(a, beg, end);
        qS(a, beg, loc-1);
        qS(a, loc+1, end);
    }
}

public int part(int a[], int beg, int end)
{

    int left, right, temp, loc, flag;
    loc = left = beg;
    right = end;
    flag = 0;
    while(flag != 1)
    {
        while((a[loc] <= a[right]) && (loc!=right))
            right--;
        if(loc==right)
            flag = 1;
        if(a[loc]>a[right])
        {
            temp = a[loc];
            a[loc] = a[right];
            a[right] = temp;
            loc = right;
        }
        if(flag!=1)
        {
            while((a[loc] >= a[left]) && (loc!=left))

```

```

    left++;
    if(loc==left)
    flag =1;
    if(a[loc] <a[left])
    {
        temp = a[loc];
        a[loc] = a[left];
        a[left] = temp;
        loc = left;
    }
}
}
return loc;
}

```

```

public void swap2(int num1, int num2) throws InterruptedException{

    for(int i=0; i<30;i++){

        button_array[num1].setLocation(button_array[num1].getLocation().x,
button_array[num1].getLocation().y+1);

        button_array[num2].setLocation(button_array[num2].getLocation().x,
button_array[num2].getLocation().y-1);

        t.sleep(10);
    }

    int len = button_array[num2].getX()- button_array[num1].getX();

```

```

    for (int i=0;i<len;i++){

        button_array[num1].setLocation(button_array[num1].getLocation().x+1,
button_array[num1].getLocation().y);

        button_array[num2].setLocation(button_array[num2].getLocation().x-1,
button_array[num2].getLocation().y);

        t.sleep(10);
    }

    for (int i=0;i<30;i++){

        button_array[num1].setLocation(button_array[num1].getLocation().x,
button_array[num1].getLocation().y-1);

        button_array[num2].setLocation(button_array[num2].getLocation().x,
button_array[num2].getLocation().y+1);

        t.sleep(10);
    }

    JButton temp = button_array[num1];
    button_array[num1] = button_array[num2];
    button_array[num2] = temp;
    t.sleep(10);
}

// Variables declaration - do not modify
private javax.swing.JButton jButton1;
private javax.swing.JButton jButton2;
private javax.swing.JButton jButton3;
private javax.swing.JButton jButton4;
private javax.swing.JButton jButton5;

```

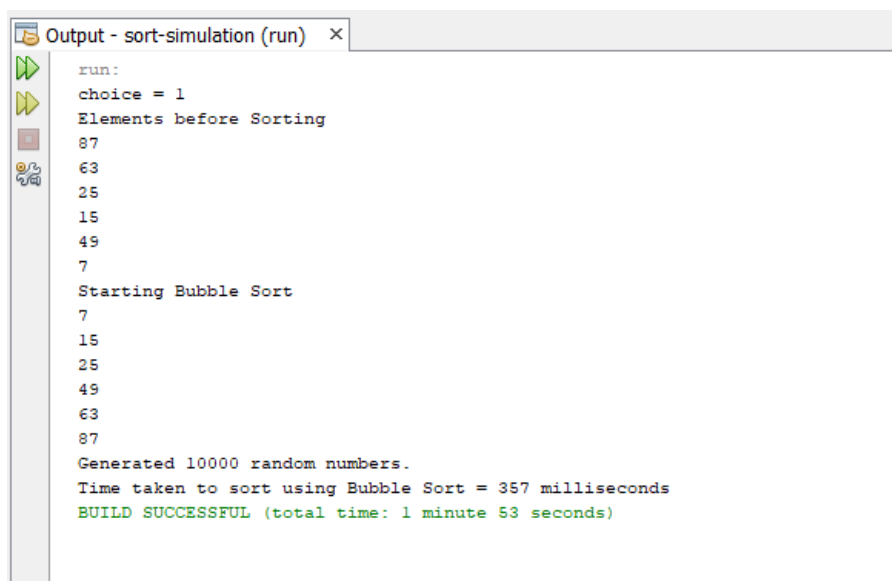
```

private javax.swing.JButton jButton6;
private javax.swing.JButton jButton7;
private javax.swing.JButton jButton8;
private javax.swing.JComboBox jComboBox1;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JTextField jTextField1;
private javax.swing.JTextField jTextField2;
private javax.swing.JTextField jTextField3;
private javax.swing.JTextField jTextField4;
private javax.swing.JTextField jTextField5;
private javax.swing.JTextField jTextField6;
private javax.swing.JTextField jTextField7;

// End of variables declaration
}

```

RESULTS:



```

run:
choice = 1
Elements before Sorting
87
63
25
15
49
7
Starting Bubble Sort
7
15
25
49
63
87
Generated 10000 random numbers.
Time taken to sort using Bubble Sort = 357 milliseconds
BUILD SUCCESSFUL (total time: 1 minute 53 seconds)

```

```
Output - sort-simulation (run) ×
run:
choice = 2
Elements before Sorting
56
23
78
12
96
30
Starting Quick Sort
12
23
30
56
78
96
Generated 10000 random numbers.
Time taken to sort using Quick Sort = 7 milliseconds
BUILD SUCCESSFUL (total time: 40 seconds)
```

Bubble Sort ▼

45 87 63 12 92 5

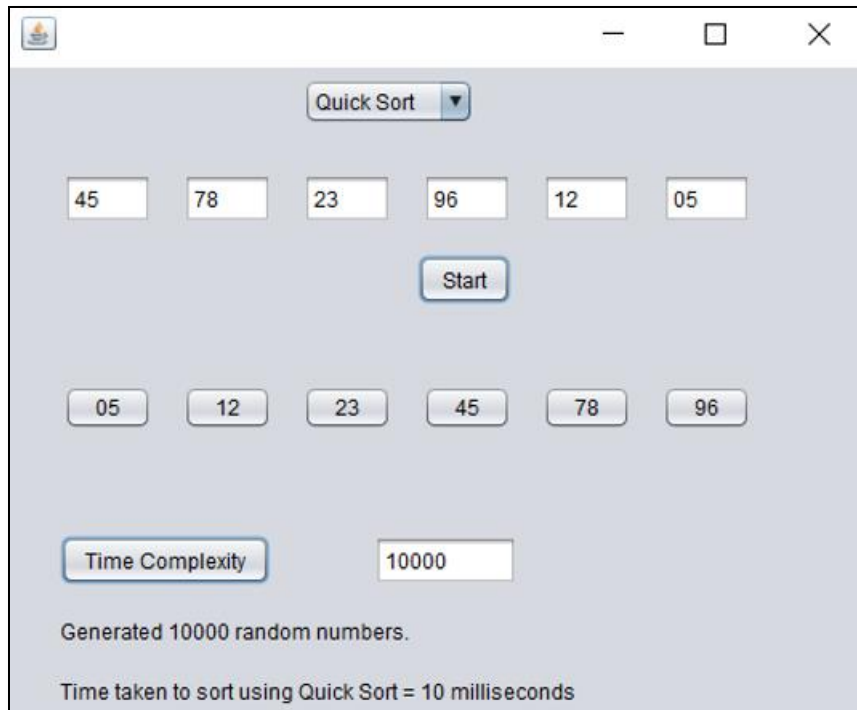
Start

5 12 45 63 87 92

Time Complexity 10000

Generated 10000 random numbers.

Time taken to sort using Bubble Sort = 197 milliseconds



A screenshot of a Java Swing window titled "Quick Sort". The window has a standard title bar with a minimize button, a maximize button, and a close button. The main content area is light gray and contains the following elements:

- A dropdown menu at the top center with the text "Quick Sort" and a downward arrow.
- A row of six white rectangular boxes containing the numbers: 45, 78, 23, 96, 12, and 05.
- A blue "Start" button with white text, centered below the first row of numbers.
- A row of six light gray rounded rectangular boxes containing the numbers: 05, 12, 23, 45, 78, and 96.
- A label "Time Complexity" in a light gray rounded rectangular box, followed by a white rectangular box containing the number "10000".
- Two lines of text at the bottom: "Generated 10000 random numbers." and "Time taken to sort using Quick Sort = 10 milliseconds".

Java concepts covered:

- Methods and classes
- Inheritance
- Packages
- Exception Handling
- Java I/O basics
- File Handling

Tools Required:

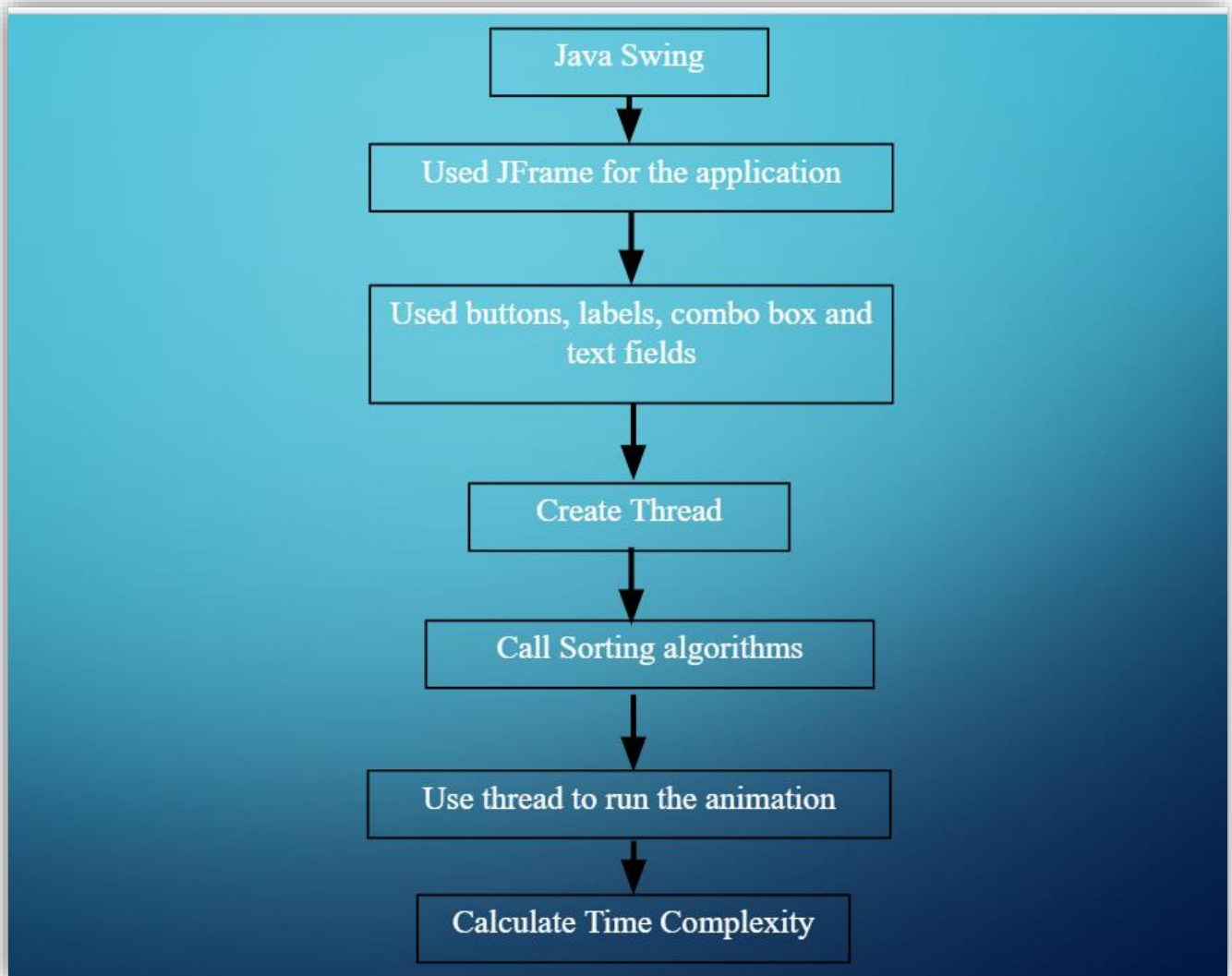
Software requirements:

- **Operating System:**
 1. Linux
 2. Windows
- **Language Used:**
 1. Java
- **Editor:**
 1. JGrasp
 2. Visual Studio
 3. Atom
- **Java Development Kit:**
 1. JDK SE 8
 2. JDK SE 9

Hardware requirements:

1. Minimum RAM 512 MB
2. 2 MB Secondary memory or above
3. x86 or above processor

HIGH LEVEL DESIGN:



References:

- www.wikipedia.org
- www.geeksforgeeks.com
- www.w3resources.com
- www.javatutorialspoint.com
- Primer for JAVA- E. Balaguruswamy
- Stack overflow
- Stack exchange
- Web developers