Criterion C: Development

There were several functions that demonstrated the complexity of the algorithm. These are listed in order of complexity.

1. **Resolving Order and adding the data into the inventory**: It needed few for loops functions and search for equipment that needed to be resolved.
2. **Search Function:** It is required for loops functions inside a for loop functions to search for the equipment depending on the subject and the topic the user enters.
3. **Adding data to the Inventory data and making the Inventory file:**  This applies to both the inventory and order functions. This code relies strongly on being consistent to the procedural inbuilt functions to create the file and add data into the file.

**Libraries Used:**

I imported Jxl Libraries since Apache POI Libraries was incompatible with my development PC.

Below are the libraries I imported:

import java.io.File;

import java.io.FileNotFoundException;

import java.io.FileOutputStream;

import java.io.IOException;

import java.util.Iterator;

import jxl.Sheet;

import jxl.Workbook;

import java.util.ArrayList;

import java.util.Locale;

import javax.swing.table.DefaultTableModel;

import jxl.read.biff.BiffException;

import jxl.write.Label;

import jxl.write.WritableSheet;

import jxl.write.WritableWorkbook;

import jxl.write.WriteException;

import java.util.Iterator;

import java.util.Scanner;

import java.util.logging.Level;

import java.util.logging.Logger;

**Database: Excel**

To record the data, I used Excel as a database for the application. It serves in a simple yet important role in this program by being able to read the recorded data and use the data for major sorting methods.

**Major Jxl methods used:**

1. Creating an excel file:

File nameoffile= new File("nameofexcelfile.xls"); (Kumar, 2019)

WritableWorkbook nameofworkbook= Workbook.createWorkbook(nameoffile); (Kumar, 2019)

1. Creating a sheet:

WritableSheet nameofsheet=nameofworkbook.createSheet("titleforsheet", 0); (Kumar, 2019)

1. Adding a new row:

nameofsheet.addCell(new Label(rowno,columno, Inputdata)); (Lars Vogel, 2019)

1. Reading a specific cell:

nameofsheet.getCell(rowno,columno).getContents(); (Lars Vogel, 2019)

1. Update cell:

nameofSheet.addCell(new Label(rowno,columno,Inputdata)); (Lars Vogel, 2019)

**GUI:**

The GUI classes handled all the functions of input of data and viewing the sorted data.

**Using JFrames and JPanels**

public class MainUI extends javax.swing.JFrame {

private javax.swing.JButton jButton1; //Search Button

private javax.swing.JLabel jLabel1; // “Science Lab Equipment Register”

private javax.swing.JLabel jLabel2; // “Welcome to UWC-USA Science Laboratory”

private javax.swing.JLabel jLabel3; // Image for the MainUI

private javax.swing.JMenu jMenu2; // Record Menu

private javax.swing.JMenu jMenu3; // Order Menu

private javax.swing.JMenuBar jMenuBar1; //The Menu Bar

private javax.swing.JMenuItem jMenuItem2; // View Record Menu

private javax.swing.JMenuItem jMenuItem3; // Add Record Menu

private javax.swing.JMenuItem jMenuItem4; // Resolve Order Menu

private javax.swing.JMenuItem jMenuItem5; // Add Order Menu

private javax.swing.JPanel jPanel1; // JPanel for Search Button and JLabel1

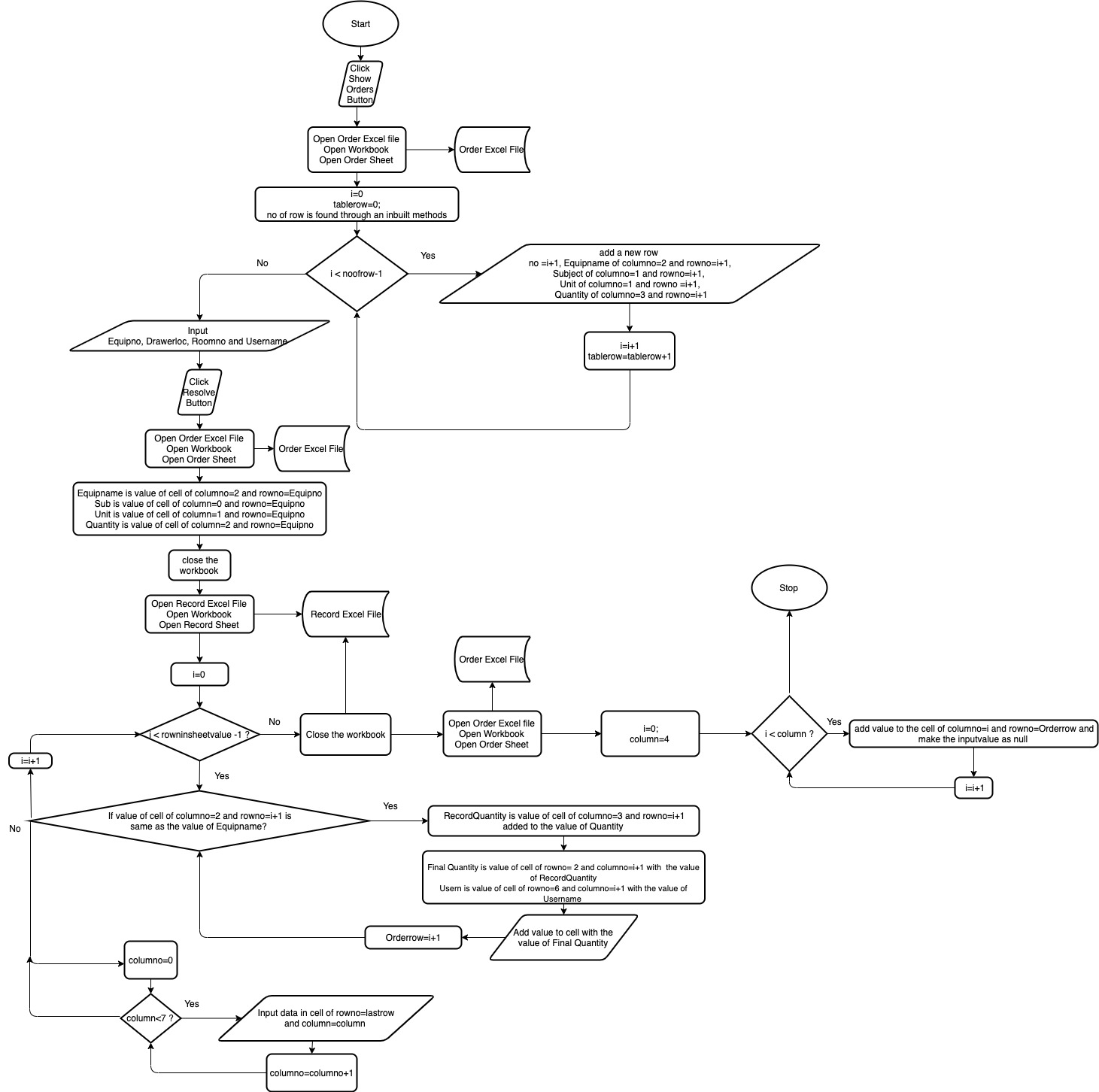
private javax.swing.JPanel jPanel2; // JPanel for the Image

// End of variables declaration

}

**Resolve Order Flowchart:**

* Take the order number input, users, locations displayed in the table in the ResolveorderGui.
* It then checks the Order excel file and deletes the order. It then records the info or changes the quantity info if there is the same equipment in stock.



**Java code for the algorithm:**

private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {

// Show Orders Button

Workbook workbook = null;

try {

workbook = Workbook.getWorkbook(new File("Order.xls")); (Datsabk, 2019)

// get the workbook from the Order Excel File

} catch (IOException ex) {

Logger.getLogger(ressolvee.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(ressolvee.class.getName()).log(Level.SEVERE, null, ex);

}

Sheet sheet = workbook.getSheet(0); // get the Order Sheet from the Excel file (Datsabk, 2019)

DefaultTableModel model=(DefaultTableModel)jTable1.getModel(); (1bestCsharp,2019)

// Use the Table inbuilt method to input data into it

for(int i=0; i<sheet.getRows()-1; i++){

model.addRow(new Object[]{i+1,sheet.getCell(2,i+1).getContents(),sheet.getCell(0,i+1).getContents(),

sheet.getCell(1,i+1).getContents(),sheet.getCell(3,i+1).getContents()}); (1bestCsharp,2019)

// adds all the data in the jTable1 from the Order Excel file

}

if (workbook!= null) {

workbook.close(); // close the Workbook (Datsabk, 2019)

}

System.out.println("tableshows"); // test code

}

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

// Resolve Order Button

Workbook orderbook = null; (Kumar, 2019)

try {

orderbook = Workbook.getWorkbook(new File("Order.xls")); (Kumar, 2019)

// get the workbook from the Order Excel File

} catch (IOException ex) {

Logger.getLogger(ressolvee.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(ressolvee.class.getName()).log(Level.SEVERE, null, ex);

}

Sheet ordersheet = orderbook.getSheet(0); // get the Order Sheet from the Excel file

(Lars Vogel, 2019)

int no= (int) Float.parseFloat(jTextField1.getText());

// convert value of jTextField1 into Integer value

// get the values from the cells in the sheet

String Equipname=ordersheet.getCell(2,no).getContents(); (Datsabk, 2019)

String Sub=ordersheet.getCell(0,no).getContents(); (Datsabk, 2019)

String Unit=ordersheet.getCell(1,no).getContents(); (Datsabk, 2019)

String Quant=ordersheet.getCell(3,no).getContents(); (Datsabk, 2019)

System.out.println(Sub+Unit+Quant); // test code

// assign variable to the corresponding input

String Drawer=jTextField4.getText();

String Roomno=jTextField3.getText();

String Usern=ordersheet.getCell(4,no).getContents();

Newrow[0]=Equipname;

Newrow[1]=Sub;

Newrow[2]=Unit;

Newrow[3]=Roomno;

Newrow[4]=Drawer;

Newrow[5]=Quant;

Newrow[6]=Usern;

orderbook.close(); (Lars Vogel, 2019)

System.out.println("order info's retrieved"); // test code

// delete the orders resolved from the order excel file

Workbook clearorder = null; (Kumar, 2019)

try {

clearorder = Workbook.getWorkbook(new File("Order.xls")); (Lars Vogel, 2019)

//call the workbook to access its sheet.

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

WritableWorkbook Copy = null; //create a new writeableWorkbook (Lars Vogel, 2019)

try {

Copy = clearorder.createWorkbook(new File("Order.xls"), clearorder); (Lars Vogel, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet CopySheet = Copy.getSheet(0); //get the record sheet (Lars Vogel, 2019)

int lastrow=CopySheet.getRows(); //get the last row (Lars Vogel, 2019)

System.out.println("nextinput"); //check code

Scanner in=new Scanner(System.in); //check code

//Adding cell data with the input value related to each column

for(int i=0; i<5; i++){

try {

CopySheet.addCell(new Label(i,no,"")); (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

Copy.write(); //write the data (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

try {

Copy.close(); //close the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

} catch (WriteException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

System.out.println("order info cleared from order excel file");

//open record excel file and compare if there are same equipments

String finalquant = null;

int row=0;

String newinputfinalquant=null;

Workbook recordcheck = null; (Datsabk, 2019)

try {

recordcheck = Workbook.getWorkbook(new File("Record.xls")); (Datsabk, 2019)

// get the workbook from the Order Excel File

} catch (IOException ex) {

Logger.getLogger(ressolvee.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(ressolvee.class.getName()).log(Level.SEVERE, null, ex);

}

Sheet recordsheet = recordcheck.getSheet(0); // get the Order Sheet from the Excel file

(Datsabk, 2019)

for(int k=0; k<recordsheet.getRows()-1; k++){

// compares the Equipname and the cell data in the record sheet of Equipment column in //lowercase

System.out.println(recordsheet.getCell(0,k+1).getContents().toLowerCase()); //testcode

System.out.println(Equipname.toLowerCase()); (Singh, 2019)

System.out.println(recordsheet.getCell(0,k+1).getContents().toLowerCase().equals(Equipname.toLowerCase())==true); // test code (Singh, 2019)

if(recordsheet.getCell(0,k+1).getContents().toLowerCase().equals(Equipname.toLowerCase())==true) {

// compares name of the equipments in the record sheet and Ordersheet (Singh, 2019)

System.out.println(Integer.valueOf(Quant));

System.out.println(Integer.valueOf(recordsheet.getCell(5,k+1).getContents()));

int updatedquant=Integer.valueOf(Quant)+Integer.valueOf(recordsheet.getCell(5,k+1).getContents());

// quantity updated (GeeksforGeeks, 2019)

System.out.println(updatedquant);

finalquant=String.valueOf(updatedquant); (GeeksforGeeks, 2019)

System.out.println(finalquant);

row=k+1;

newrowinfo=0;

k=recordsheet.getRows()-1; (Datsabk, 2019)

}

}

if(newrowinfo==1){ //if it doesn’t match

System.out.println("new inputs will be recorded");

jTextField5.setText("Enter the room no. and drawer no.");

}

System.out.println("quantity updated or location info input received");

recordcheck.close(); // close the record excel file (Datsabk, 2019)

System.out.println(newrowinfo);

//writing the final updated record file

Workbook finalrecord = null; (Kumar, 2019)

try {

finalrecord = Workbook.getWorkbook(new File("Record.xls")); (Kumar, 2019)

//call the workbook to access its sheet.

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

WritableWorkbook frecord = null; //create a new writeableWorkbook

try {

frecord = finalrecord.createWorkbook(new File("Record.xls"), finalrecord); (Lars Vogel, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet fupdate = frecord.getSheet(0); //get the record sheet (Datsabk, 2019)

int lrow=fupdate.getRows(); //get the last row (Datsabk, 2019)

System.out.println("nextinput"); // test code

//Adding cell data with the input value related to each column

if(newrowinfo==0){ //if there are matching equipments

try {

fupdate.addCell(new Label(5,row,finalquant)); //updating the quantity in the record file

(Datsabk, 2019)

} catch (Exception ex) {

System.out.println("qqqqqqqq"); // test code

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

try {

fupdate.addCell(new Label(6,row,Newrow[6])); // adding the username (Datsabk, 2019)

} catch (Exception ex) {

System.out.println("qqqqqqqq"); // test code

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

System.out.println(finalquant);

System.out.println(row);

}

try {

frecord.write(); //write the data (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

try {

frecord.close(); //close the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

} catch (WriteException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

System.out.println("resolveorder finished");

}

private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {

// move to the main page and closes the current page

dispose(); (Esau,2019)

MainUI M= new MainUI(); (ProgramCreek, 2019)

M.setVisible(true); (ProgramCreek, 2019)

}

private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {

// to add a new equipments from the order into the record file

Workbook finrecord = null; (Datsabk, 2019)

try {

finrecord = Workbook.getWorkbook(new File("Record.xls")); (Datsabk, 2019)

//call the workbook to access its sheet.

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

WritableWorkbook frecord = null; //create a new writeableWorkbook (Datsabk, 2019)

try {

frecord = finrecord.createWorkbook(new File("Record.xls"), finrecord); (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet fupdate = frecord.getSheet(0); //get the record sheet (Datsabk, 2019)

int lrow=fupdate.getRows(); //get the last row (Datsabk, 2019)

System.out.println("nextinput"); // test code

//Adding cell data with the input value related to each column

if(newrowinfo==1) { // new row of record

for(int j=0; j<7; j++){

try {

fupdate.addCell(new Label(j,lrow,Newrow[j])); // adding the equipments informations

(Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

try {

fupdate.addCell(new Label(3,lrow,jTextField3.getText())); // room no (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

try {

fupdate.addCell(new Label(4,lrow,jTextField4.getText())); // drawer no (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

System.out.println(Newrow[j]); // test code

}

}

try {

frecord.write(); //write the data (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

try {

frecord.close(); //close the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

} catch (WriteException ex) {

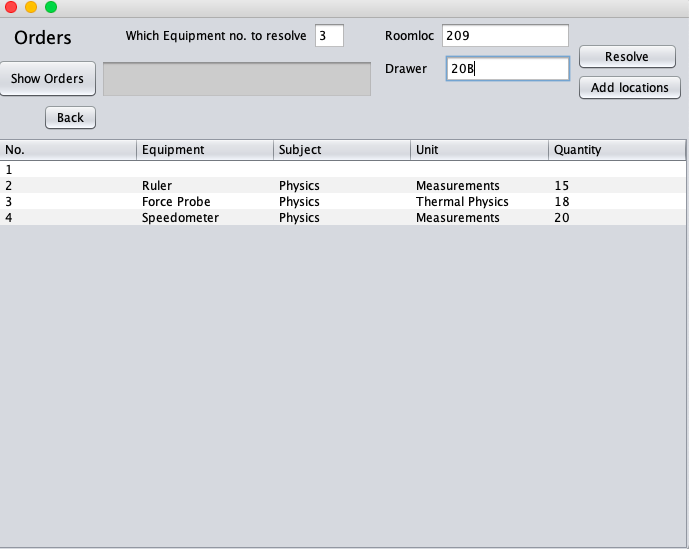
Logger.getLogger(Resolveorder.class.getName()).log(Level.SEVERE, null, ex);

}

System.out.println("resolveorder finished");

}

**Resolve Order Interface**(Sample Input)**:**



**Updated Record Excel File:**

New lines of inventory of equipment are added while Order excel file is cleared of resolved order equipment from its data.

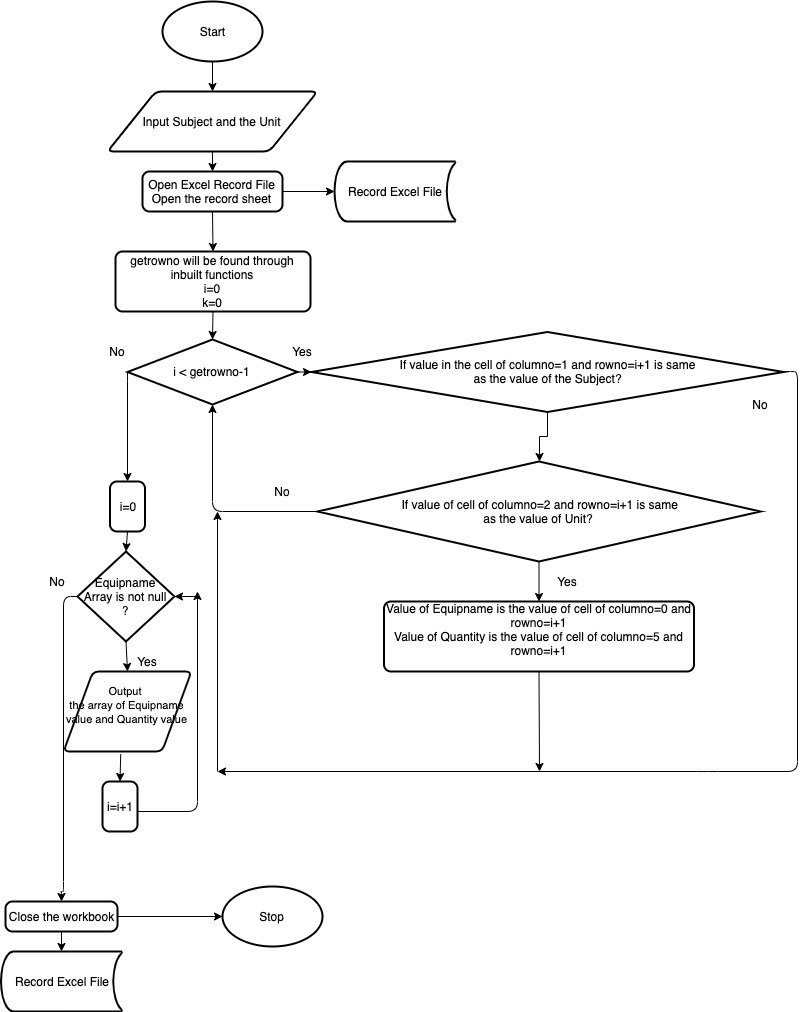


**Updated Order Excel File:**



**Search Algorithm Flowchart:**

* Take the input of the name of the subject and topic from the user
* Open the record file and it checks equipment that matches the subject and the topic.
* The algorithm loops to all the records if necessary to find the equipment which matches the information.
* If there is equipment that matches the information are displayed in the table of SearchGui.
* If there is no matching equipment, it simply does not display any equipment name and its information.



**Java code for the algorithm:**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {

Workbook workbook = null; (Datsabk, 2019)

try {

workbook = Workbook.getWorkbook(new File("Record.xls")); (Datsabk, 2019)

// get the Workbook from the Record File

Sheet sheet = workbook.getSheet(0); // get the record sheet (Datsabk, 2019)

int k=0; // it is used to iterate the array index of Equipname[] and Quantity []

// This algorithm is one of the ingenuities of searching the equipment

for(int i=0; i<sheet.getRows()-1; i++){ (Datsabk, 2019)

if(sheet.getCell(1,i+1).getContents().toLowerCase().equals(jTextField1.getText().toLowerCase())==true){

(Singh, 2019)

// Compares input data of subjects and subject record in the Record file in lower case

// the input has to be exact in spelling with any subject record in the file

if(sheet.getCell(2,i+1).getContents().toLowerCase().equals(jTextField2.getText().toLowerCase())==true){

(Singh, 2019)

// Compares input data of Unit and the Unit record in the Record File in lower case

Equipname[k]=sheet.getCell(0,i+1).getContents(); (Datsabk, 2019)

// stores the name of equipment that matches with the input of Subjects and Units

Quantity[k]=sheet.getCell(5,i+1).getContents(); (Datsabk, 2019)

// stores the no of Quantity of the Equipment which matches with the input of Subjects and Units

k++;

}

}

}

} catch (IOException e) {

e.printStackTrace();

} catch (BiffException e) {

e.printStackTrace();

} finally {

if (workbook != null) {

workbook.close(); // close the workbook

}

}

DefaultTableModel model=(DefaultTableModel)jTable1.getModel(); (1bestCsharp,2019)

// Use the Table inbuilt method to input data into it

int i=0;

while(Equipname[i]!=null){ // The loop goes until the value of Equipment[i]=null model.addRow(new Object[]{Equipname[i],Quantity[i]}); // adds data in the jTable1

(1bestCsharp,2019)

i++;

}

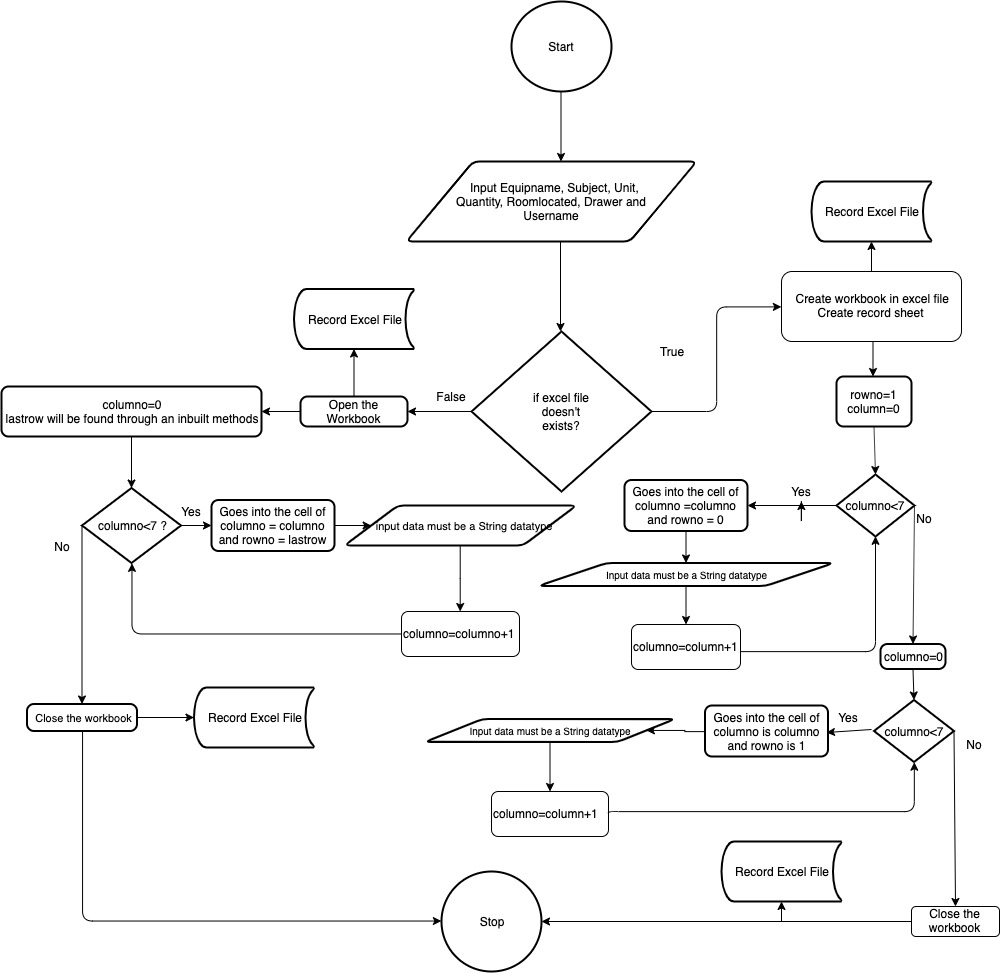
}

**Search Interface**(Sample Input and result)**:**



**Record Algorithm Flowchart:**

* Take the inputs of equipment in stock and its informations from the user.
* Check if the excel file exists.
* Create a workbook and the excel file if it does not exist.
* Writes the inputs of users in the excel file from the last row previously written.
* Close the workbook after the recording operation is complete.



**Java code for this algorithm:**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { // Record Button

String[ ] Invalue= new String[7]; // String Array to store the input datas

String Equipname=jTextField1.getText();

Invalue[0]=Equipname;

String Sub=jTextField2.getText();

Invalue[1]=Sub;

String Unit=jTextField3.getText();

Invalue[2]=Unit;

String Room=jTextField4.getText();

Invalue[3]=Room;

String Drawer=jTextField5.getText();

Invalue[4]=Drawer;

String Quantity=jTextField6.getText();

Invalue[5]=Quantity;

String Username=jTextField7.getText();

Invalue[6]=Username;

File f= new File("Record.xls"); //declare the file and the filename (Datsabk, 2019)

if(!f.exists()){ //checks if the excel Record file exists or not

// Codes if the file doesn't exist.

WritableWorkbook workbook = null; (Datsabk, 2019)

try {

workbook = Workbook.createWorkbook(f); //creates the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet sheet=workbook.createSheet("record", 0); //create the sheet named record.

(Datsabk, 2019)

String[] Inputvalue= new String[7];

System.out.println("The sheet created is named as :"+sheet.getName()); //testcode

String lblEquip="Name of Equipment";

Inputvalue[0]=lblEquip;

String lblSub="Name of Subject";

Inputvalue[1]=lblSub;

String lblUnit="Unit in the Subject";

Inputvalue[2]=lblUnit;

String lblRoomloc="Room Location";

Inputvalue[3]=lblRoomloc;

String lblDrawer="Drawer";

Inputvalue[4]=lblDrawer;

String lblQuantity="Quantity";

Inputvalue[5]=lblQuantity;

String lblUsername="Username";

Inputvalue[6]=lblUsername;

//Adding cell data with the title of each column of cell

for(int i=0; i<7; i++){

try {

sheet.addCell(new Label(i,0,Inputvalue[i])); (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

//Adding cell data with the input value related to each column

for(int i=0; i<7; i++){

try {

sheet.addCell(new Label(i,1,Invalue[i])); (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

//workbook is closed.

try {

workbook.write(); (Datsabk, 2019)

workbook.close(); (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

else{

// If the file exists

Workbook WorkBook = null; (Datsabk, 2019)

try {

WorkBook = Workbook.getWorkbook(new File("Record.xls")); //call the workbook to access its sheet.

(Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

WritableWorkbook Copy = null; //create a new writeableWorkbook (Datsabk, 2019)

try {

Copy = Workbook.createWorkbook(new File("Record.xls"), WorkBook); (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet CopySheet = Copy.getSheet(0); //get the record sheet (Datsabk, 2019)

int lastrow=CopySheet.getRows(); //get the last row (Datsabk, 2019)

System.out.println("nextinput"); // test code

Scanner in=new Scanner(System.in); // test code

//Adding cell data with the input value related to each column

for(int i=0; i<7; i++){

try {

CopySheet.addCell(new Label(i,lastrow,Invalue[i])); (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

Copy.write(); //write the data (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

try {

Copy.close(); //close the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

} catch (WriteException ex) {

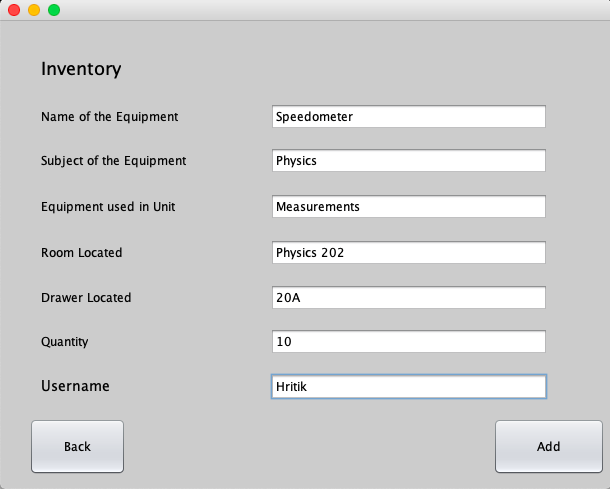
Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

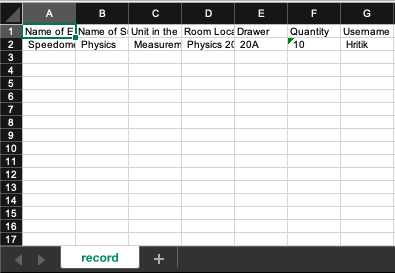
}

}

**Recording Equipments Interface**(Sample Input)**:**

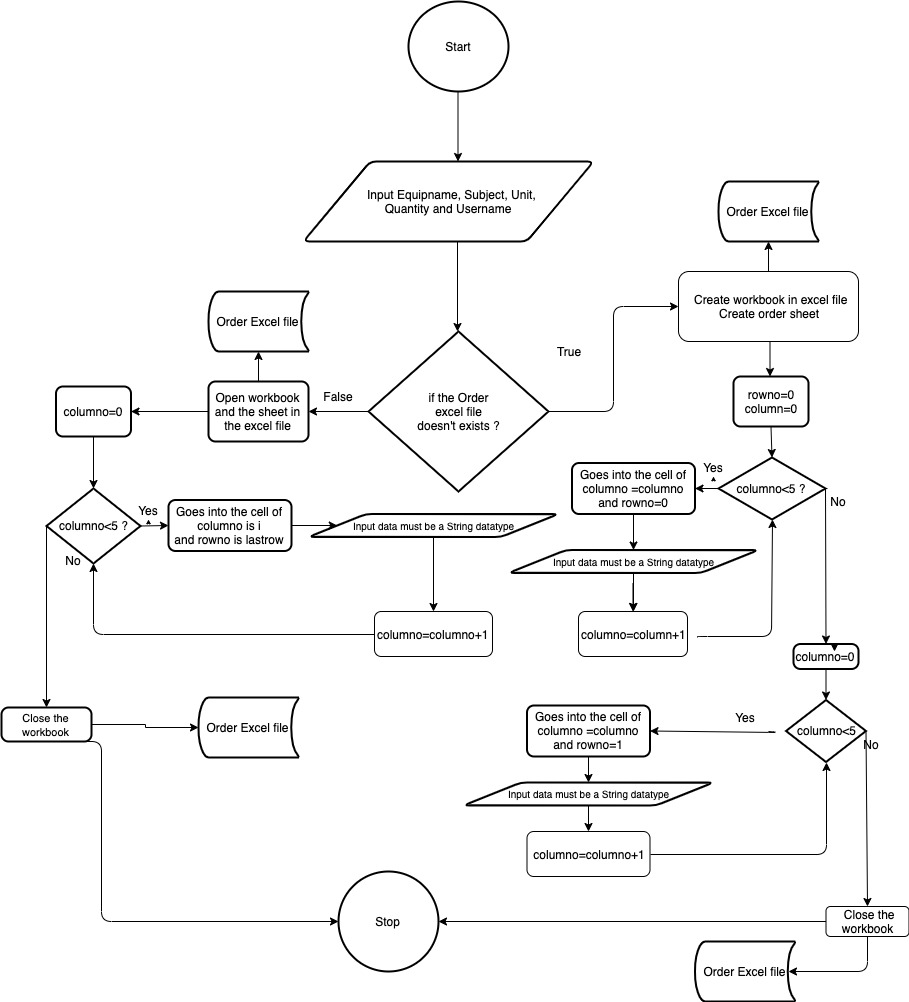


**Sample data in the Record Excel File:**

****

**Order Algorithm Flowchart:**

* Take the order inputs of new equipment from the user.
* Check if the excel file exists.
* Create a workbook and the excel file if it does not exist.
* Writes the inputs of users in the excel file from the last row previously written.
* Close the workbook after the recording operation is complete.



**Java code for this algorithm:**

private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) { // Order button

//get input values through jTextField methods and it is stored in Invalue array

String[] Invalue= new String[5];

String Sub = jTextField1.getText();

Invalue[0]=Sub;

String Unit =jTextField2.getText();

Invalue[1]=Unit;

String Equip =jTextField3.getText();

Invalue[2]=Equip;

String Quant= jTextField4.getText();

Invalue[3]=Quant;

String Username=jTextField5.getText();

Invalue[4]=Username;

// Declare Order Excel File.

File f= new File("Order.xls"); (Datsabk, 2019)

if(!f.exists()){ //if the file doesn't exist

WritableWorkbook workbook = null; (Datsabk, 2019)

try {

workbook = Workbook.createWorkbook(f); //create a workbook in the excel file

(Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet sheet=workbook.createSheet("Order", 0); //creates Order Sheet (Datsabk, 2019)

System.out.println("The sheet created is named as :"+sheet.getName()); //test code

// Inputvalue array is to store the label of the sheet in the excel file.

String[] Inputvalue= new String[5];

String Subj="Sub";

Inputvalue[0]=Subj;

String Uni="Unit";

Inputvalue[1]=Uni;

String Equi="Equipment";

Inputvalue[2]=Equi;

String Quanti="Quantity";

Inputvalue[3]=Quanti;

String User="Username";

Inputvalue[4]=User;

// loop to add the label of the sheet

for(int i=0; i<5; i++){ // i<5 depicts no. of column to add the label in.

try {

sheet.addCell(new Label(i,0,Inputvalue[i])); // adding method (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

//loop to add the input value of the corresponding data in the excel file

for(int i=0; i<5; i++){ // i<5 depicts no. of column to add the input data in.

try {

sheet.addCell(new Label(i,1,Invalue[i])); // adding method (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

workbook.write(); // write the workbook in the file (Datsabk, 2019)

workbook.close(); // close the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

else{ //if the Excel Order file exists

Workbook WorkBook = null; (Datsabk, 2019)

try {

WorkBook = Workbook.getWorkbook(new File("Order.xls")); //get the workbook

(Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

} catch (BiffException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

WritableWorkbook Copy = null; (Datsabk, 2019)

try {

Copy = Workbook.createWorkbook(new File("Order.xls"), WorkBook); (Datsabk, 2019)

//create a workbook to write the data into

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

WritableSheet CopySheet = Copy.getSheet(0); // get the Order sheet (Datsabk, 2019)

int lastrow=CopySheet.getRows(); // get the last row of the Order sheet which already has datas in.

(Datsabk, 2019)

System.out.println("nextinput"); // test code

for(int i=0; i<5; i++){ // i<5 depicts no. of column to add the input data in.

try {

CopySheet.addCell(new Label(i,lastrow,Invalue[i])); // adding method (Datsabk, 2019)

} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

}

try {

Copy.write(); // write it in the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

}

try {

Copy.close(); // close the workbook (Datsabk, 2019)

} catch (IOException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

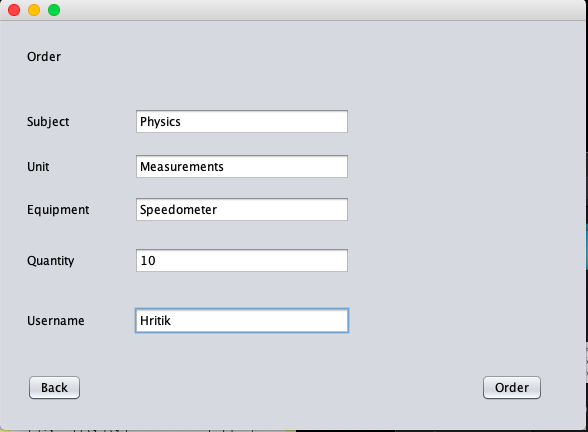
} catch (WriteException ex) {

Logger.getLogger(RecordGui.class.getName()).log(Level.SEVERE, null, ex);

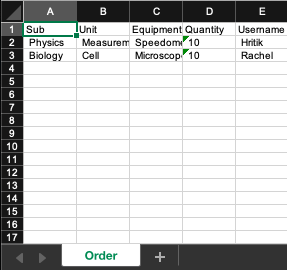
}

}

**Recording Orders Interface**(Sample Input)**:**



**Sample data in the Order Excel File:**



Word Count: 325

Works Cited

1bestCsharp. "C#, JAVA,PHP, Programming ,Source Code: Java - Populate JTable From ArrayList In Java." *C#, JAVA,PHP, Programming ,Source Code*, 1bestcsharp.blogspot.com/2016/03/java-populate-jtable-from-arraylist.html.

Accessed 19 Oct. 2019.

Datsabk. "JExcel API – Reading and Writing Excel File in Java." *Mkyong.com – Java Tutorials, Examples and Articles*, mkyong.com/java/jexcel-api-reading-and-writing-excel-file-in-java/.

Accessed 11 Sept. 2019.

Esaue, Aaron. "How to Programmatically Close a JFrame." *Stack Overflow*, stackoverflow.com/questions/1234912/how-to-programmatically-close-a-jframe. Accessed 10 Oct. 2019.

GeeksforGeeks. "Different Ways for Integer to String Conversions In Java." *GeeksforGeeks*, 9 Oct. 2019, www.geeksforgeeks.org/different-ways-for-integer-to-string-conversions-in-java/. Accessed 7 Nov. 2019.

Kumar, Rajesh. "How to Insert a Date Field in Excel Sheet Using Jexcel." *Free E-Learning Portal for Students and Professionals*, r4r.co.in/java/apis/jexcel/basic/example/jexcel\_basic\_examples.php?qid=771.

Accessed 10 Sept. 2019.

Lars Vogel. "Excel and Java - Read and Write Excel with Java - Tutorial." *Eclipse, Android and Java Training and Support*, www.vogella.com/tutorials/JavaExcel/article.html.

Accessed 12 Oct. 2019.

ProgramCreek. "Java Code Examples Javax.swing.JFrame.setVisible." *ProgramCreek.com*, www.programcreek.com/java-api-examples/?class=javax.swing.JFrame&method=setVisible. Accessed 12 Nov. 2019.

Singh, Chaitanya. "Java - String ToLowerCase() and ToUpperCase() Methods." *Beginnersbook.com*, 28 Dec. 2013, beginnersbook.com/2013/12/java-string-tolowercase-method-example/.

Accessed 28 Oct. 2019.