Criteria C: Development

Introduction

The project I created is a Java program coded on Netbeans. Netbeans was selected because it has a developed graphical user interface for the application that can be made easily with Java's swing tools. The application is a material management system that allows Robotic teams to have easier access to various kinds of materials. Besides that, teachers are able to make better purchases of materials based on the current stocks. By using Netbeans, this complicated program becomes more manageable which provides a better experience to both programmers and users.

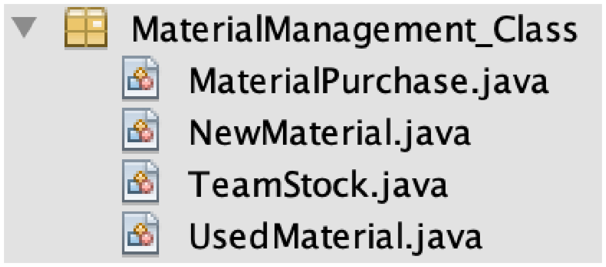
Word Count: 89

Techniques-List of Highlights

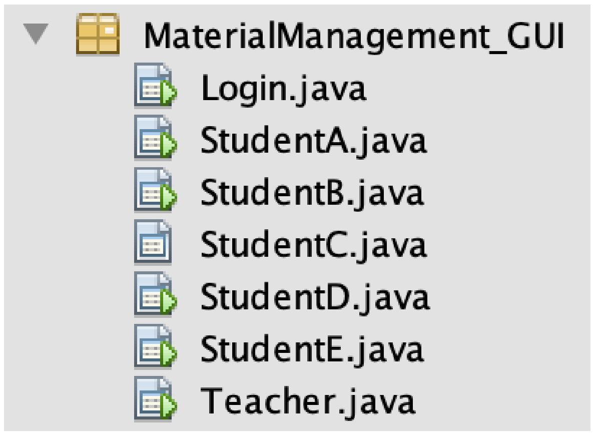
- Self-created algorithms which automatically generate a list of recommended materials based on input information.
- Encapsulation, polymorphism, inheritance
- Jswing.JOptionPane is used to notify the users about their incorrect use of the program.
- Saving and reading local files
- Nested loop
- Array and ArrayList
- Error handling (try, catch)
- The use of external packages
- Parsing value from one data type to another
- Login system which connects multiple GUIs together
- split() method is used to separate a String variable into parts.
- The use of GUI features: Textfield.setText(), ComboBox.getSelectedItem(), etc

Word Count: 0

Structure of the program



In the project I created, teachers and teams need to work with four different types of data which are purchased materials, used materials, new materials, and teams' own stocks. Each class is created for each type of material. By organizing the data in an Object Oriented Programming way, I can easily design special methods for each kind of variable. Besides that, since all the self-written methods are public, the methods can still be used by its objects and instances in other programs.



6 separate accounts are created for teachers and teams in the school. The different positions should have different accessibility to the data. One unique GUI is created for each team which allows them to work and access the information of both used and new materials. Besides the previous functions, the teacher's account can even purchase materials and have access to each team's stocks.

Word Count: 116

Data structure used

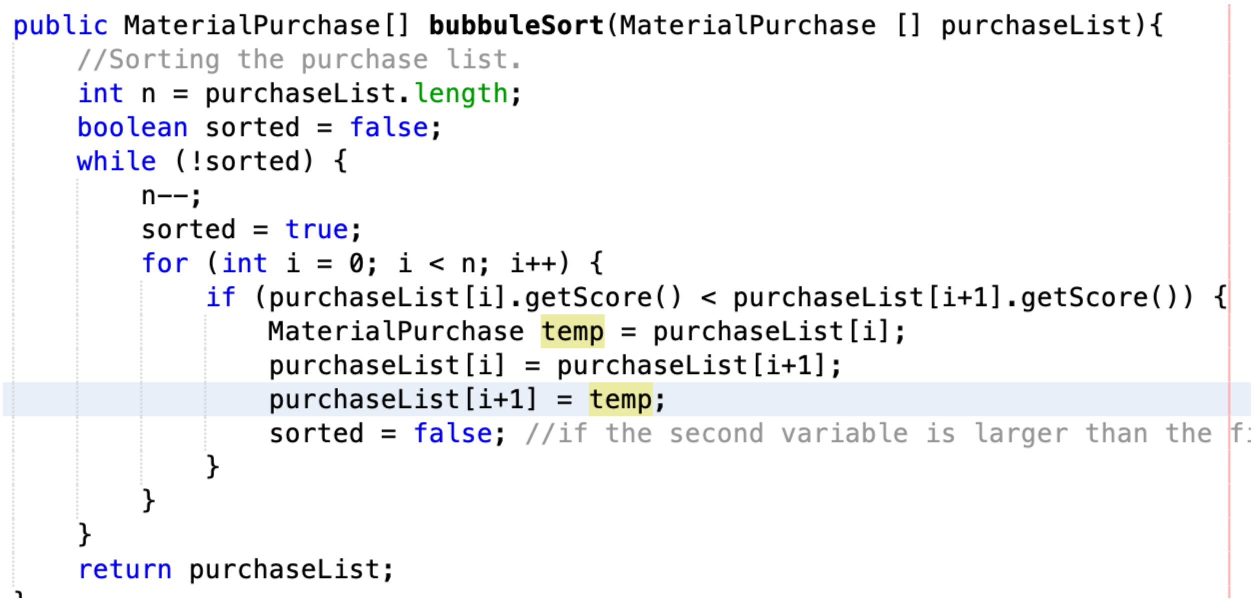
The program uses several types of lists:

ArrayList: The ArrayList is used because it is dynamic in size which makes removing certain variables from the ArrayList much easier. Besides that, this feature also makes the file read possible because it will always create enough space to load all the data from a local file without knowing the size of the data.

- ArrayList<UsedMaterial> usedMaterial: This contains the information of all the used Materials in the club.
- ArrayList<TeamStock> teamStock: This contains the materials belonging to each team. The dynamic feature of the ArrayList allows teamStock to shrink in size based on how many materials each team wants.

Array: The array is used because it allows programmers to have direct access to all the variables in the array. This feature makes the bubble sort possible. The array is used in my bubble sorting method for recommended materials for purchase which allows me to sort my data more efficiently.

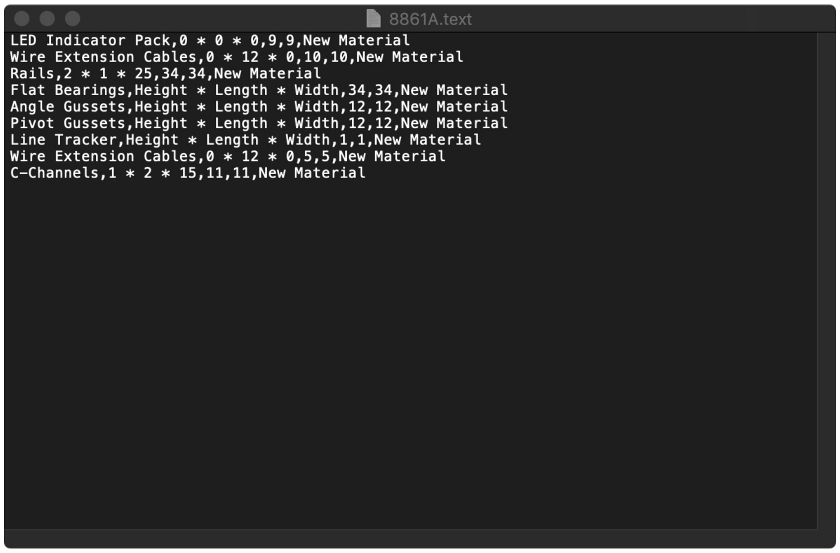
- MaterialPurchase [ ] purchaseList: The array contains the information of the materials that need to be purchased. bubbleSort method will arrange them in sequence based on their priorities represented by their scores.



```java
public MaterialPurchase[] bubbuleSort(MaterialPurchase [] purchaseList){
    //Sorting the purchase list.
    int n = purchaseList.length;
    boolean sorted = false;
    while (!sorted) {
        n--;
        sorted = true;
        for (int i = 0; i < n; i++) {
            if (purchaseList[i].getScore() < purchaseList[i+1].getScore()) {
                MaterialPurchase temp = purchaseList[i];
                purchaseList[i] = purchaseList[i+1];
                purchaseList[i+1] = temp;
                sorted = false; //if the second variable is larger than the f
            }
        }
    }
    return purchaseList;
}
```

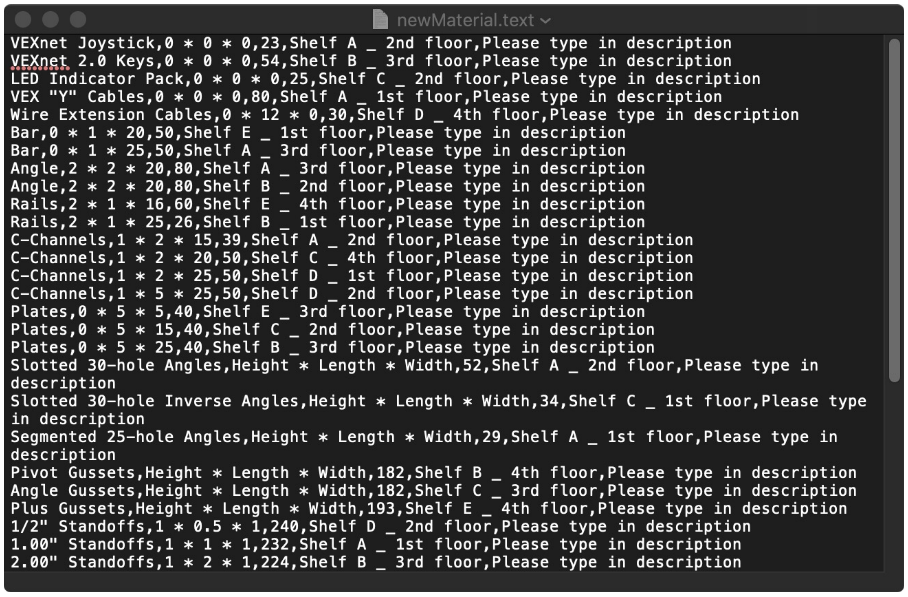
Files: Files are used to save the data of the materials. Files are located in the hard drives which allow them to keep the data after the program is closed.

- 8861A.text: This file contains the information of the 8861A's materials. Local files allow the team to update their stocks based on existing data instead of creating a new table every time.



```
LED Indicator Pack,0 * 0 * 0,9,9,New Material
Wire Extension Cables,0 * 12 * 0,10,10,New Material
Rails,2 * 1 * 25,34,34,New Material
Flat Bearings,Height * Length * Width,34,34,New Material
Angle Gussets,Height * Length * Width,12,12,New Material
Pivot Gussets,Height * Length * Width,12,12,New Material
Line Tracker,Height * Length * Width,1,1,New Material
Wire Extension Cables,0 * 12 * 0,5,5,New Material
C-Channels,1 * 2 * 15,11,11,New Material
```

- newMaterial.text: This file contains information about the new materials in the club. Saving data into a local file allows all the accounts access to the same data and the update made by various accounts can appear in the same file.

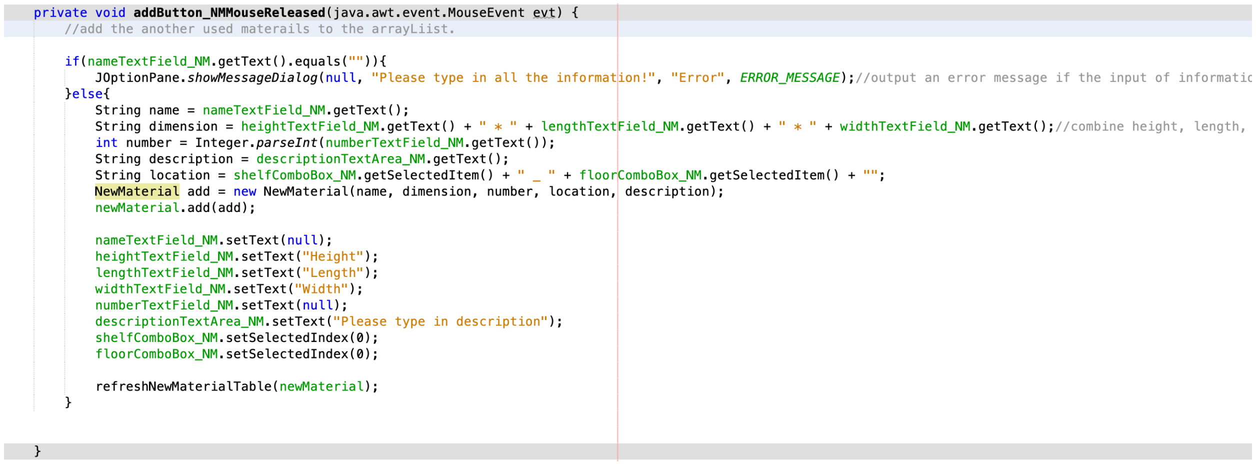


```
VEXnet Joystick,0 * 0 * 0,23,Shelf A _ 2nd floor,Please type in description
VEXnet 2.0 Keys,0 * 0 * 0,54,Shelf B _ 3rd floor,Please type in description
LED Indicator Pack,0 * 0 * 0,25,Shelf C _ 2nd floor,Please type in description
VEX "Y" Cables,0 * 0 * 0,80,Shelf A _ 1st floor,Please type in description
Wire Extension Cables,0 * 12 * 0,30,Shelf D _ 4th floor,Please type in description
Bar,0 * 1 * 20,50,Shelf E _ 1st floor,Please type in description
Bar,0 * 1 * 25,50,Shelf A _ 3rd floor,Please type in description
Angle,2 * 2 * 20,80,Shelf A _ 3rd floor,Please type in description
Angle,2 * 2 * 20,80,Shelf B _ 2nd floor,Please type in description
Rails,2 * 1 * 16,60,Shelf E _ 4th floor,Please type in description
Rails,2 * 1 * 25,26,Shelf B _ 1st floor,Please type in description
C-Channels,1 * 2 * 15,39,Shelf A _ 2nd floor,Please type in description
C-Channels,1 * 2 * 20,50,Shelf C _ 4th floor,Please type in description
C-Channels,1 * 2 * 25,50,Shelf D _ 1st floor,Please type in description
C-Channels,1 * 5 * 25,50,Shelf D _ 2nd floor,Please type in description
Plates,0 * 5 * 5,40,Shelf E _ 3rd floor,Please type in description
Plates,0 * 5 * 15,40,Shelf C _ 2nd floor,Please type in description
Plates,0 * 5 * 25,40,Shelf B _ 3rd floor,Please type in description
Slotted 30-hole Angles,Height * Length * Width,52,Shelf A _ 2nd floor,Please type in
description
Slotted 30-hole Inverse Angles,Height * Length * Width,34,Shelf C _ 1st floor,Please type
in description
Segmented 25-hole Angles,Height * Length * Width,29,Shelf A _ 1st floor,Please type in
description
Pivot Gussets,Height * Length * Width,182,Shelf B _ 4th floor,Please type in description
Angle Gussets,Height * Length * Width,182,Shelf C _ 3rd floor,Please type in description
Plus Gussets,Height * Length * Width,193,Shelf E _ 4th floor,Please type in description
1/2" Standoffs,1 * 0.5 * 1,240,Shelf D _ 2nd floor,Please type in description
1.00" Standoffs,1 * 1 * 1,232,Shelf A _ 1st floor,Please type in description
2.00" Standoffs,1 * 2 * 1,224,Shelf B _ 3rd floor,Please type in description
```

Word Count: 212

Main unique algorithms

Add new materials

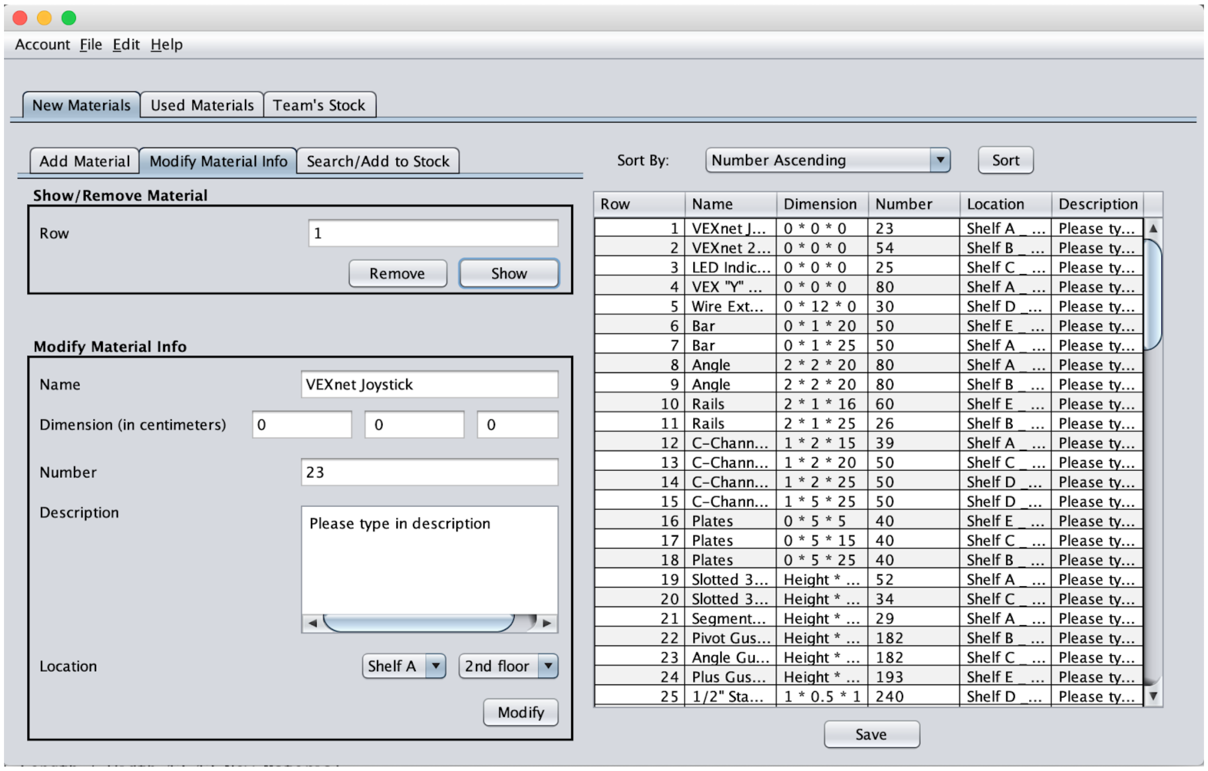


```
    private void addButton_NMMouseReleased(java.awt.event.MouseEvent evt) {
        //add the another used materails to the arrayLiist.

        if(nameTextField_NM.getText().equals("")){
            JOptionPane.showMessageDialog(null, "Please type in all the information!", "Error", ERROR_MESSAGE);//output an error message if the input of informatic
        }else{
            String name = nameTextField_NM.getText();
            String dimension = heightTextField_NM.getText() + " * " + lengthTextField_NM.getText() + " * " + widthTextField_NM.getText();//combine height, length,
            int number = Integer.parseInt(numberTextField_NM.getText());
            String description = descriptionTextArea_NM.getText();
            String location = shelfComboBox_NM.getSelectedItem() + " _ " + floorComboBox_NM.getSelectedItem() + "";
            NewMaterial add = new NewMaterial(name, dimension, number, location, description);
            newMaterial.add(add);

            nameTextField_NM.setText(null);
            heightTextField_NM.setText("Height");
            lengthTextField_NM.setText("Length");
            widthTextField_NM.setText("Width");
            numberTextField_NM.setText(null);
            descriptionTextArea_NM.setText("Please type in description");
            shelfComboBox_NM.setSelectedIndex(0);
            floorComboBox_NM.setSelectedIndex(0);

            refreshNewMaterialTable(newMaterial);
        }


    }
```

This code will create a new instance in the new material ArrayList based on users' input information. The code will firstly take all the needed information from the GUI. Then, variables are created to store the input information. After that, the add() method is used to add the new material into the ArrayList as a new instance. Since the ArrayList is dynamic in size, when a new instance is added into it, its size will be automatically increased by one and instance will be added to the tail of the list.

After the input process is down, the refreshNewMaterialTable() method is implemented to refresh the display table to keep the table updated.

Modify the information of certain material



Show/Remove Material

Row: 1

Modify Material Info

Name: VEXnet Joystick

Dimension (in centimeters): 0 0 0

Number: 23

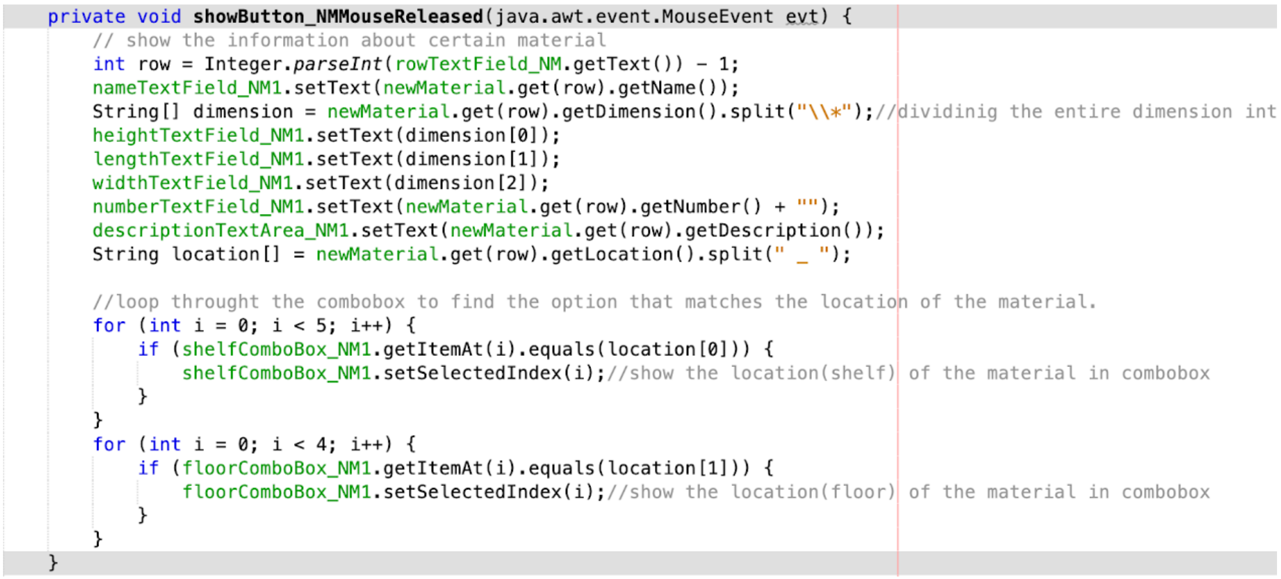
Description: Please type in description

Location: Shelf A, 2nd floor

Sort By: Number Ascending

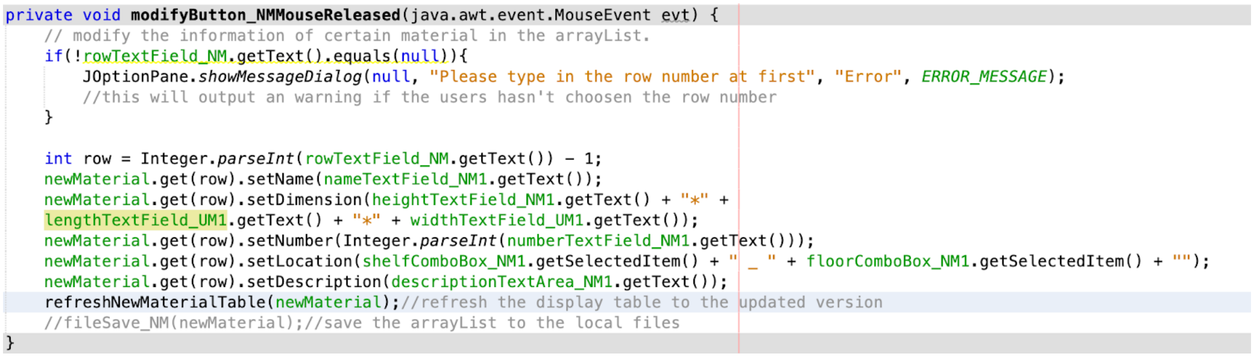
Row	Name	Dimension	Number	Location	Description
1	VEXnet J...	0 * 0 * 0	23	Shelf A _ ...	Please ty...
2	VEXnet 2...	0 * 0 * 0	54	Shelf B _ ...	Please ty...
3	LED Indic...	0 * 0 * 0	25	Shelf C _ ...	Please ty...
4	VEX "Y" ...	0 * 0 * 0	80	Shelf A _ ...	Please ty...
5	Wire Ext...	0 * 12 * 0	30	Shelf D _...	Please ty...
6	Bar	0 * 1 * 20	50	Shelf E _ ...	Please ty...
7	Bar	0 * 1 * 25	50	Shelf A _ ...	Please ty...
8	Angle	2 * 2 * 20	80	Shelf A _ ...	Please ty...
9	Angle	2 * 2 * 20	80	Shelf B _ ...	Please ty...
10	Rails	2 * 1 * 16	60	Shelf E _ ...	Please ty...
11	Rails	2 * 1 * 25	26	Shelf B _ ...	Please ty...
12	C-Chann...	1 * 2 * 15	39	Shelf A _ ...	Please ty...
13	C-Chann...	1 * 2 * 20	50	Shelf C _ ...	Please ty...
14	C-Chann...	1 * 2 * 25	50	Shelf D _...	Please ty...
15	C-Chann...	1 * 5 * 25	50	Shelf D _...	Please ty...
16	Plates	0 * 5 * 5	40	Shelf E _ ...	Please ty...
17	Plates	0 * 5 * 15	40	Shelf C _ ...	Please ty...
18	Plates	0 * 5 * 25	40	Shelf B _ ...	Please ty...
19	Slotted 3...	Height * ...	52	Shelf A _ ...	Please ty...
20	Slotted 3...	Height * ...	34	Shelf C _ ...	Please ty...
21	Segment...	Height * ...	29	Shelf A _ ...	Please ty...
22	Pivot Gus...	Height * ...	182	Shelf B _ ...	Please ty...
23	Angle Gu...	Height * ...	182	Shelf C _ ...	Please ty...
24	Plus Gus...	Height * ...	193	Shelf E _ ...	Please ty...
25	1/2" Sta...	1 * 0.5 * 1	240	Shelf D _...	Please ty...

To make sure that the data for new materials are always updated, users are allowed to manually modify the information of certain material on the display table. After typing in the row number, the users can click the show button to demonstrate the material's information in the following text fields.



```java
private void showButton_NMMouseReleased(java.awt.event.MouseEvent evt) {
    // show the information about certain material
    int row = Integer.parseInt(rowTextField_NM.getText()) - 1;
    nameTextField_NM1.setText(newMaterial.get(row).getName());
    String[] dimension = newMaterial.get(row).getDimension().split("\\*");//dividinig the entire dimension int
    heightTextField_NM1.setText(dimension[0]);
    lengthTextField_NM1.setText(dimension[1]);
    widthTextField_NM1.setText(dimension[2]);
    numberTextField_NM1.setText(newMaterial.get(row).getNumber() + "");
    descriptionTextArea_NM1.setText(newMaterial.get(row).getDescription());
    String location[] = newMaterial.get(row).getLocation().split(" _ ");

    //loop throught the combobox to find the option that matches the location of the material.
    for (int i = 0; i < 5; i++) {
        if (shelfComboBox_NM1.getItemAt(i).equals(location[0])) {
            shelfComboBox_NM1.setSelectedIndex(i);//show the location(shelf) of the material in combobox
        }
    }
    for (int i = 0; i < 4; i++) {
        if (floorComboBox_NM1.getItemAt(i).equals(location[1])) {
            floorComboBox_NM1.setSelectedIndex(i);//show the location(floor) of the material in combobox
        }
    }
}
```

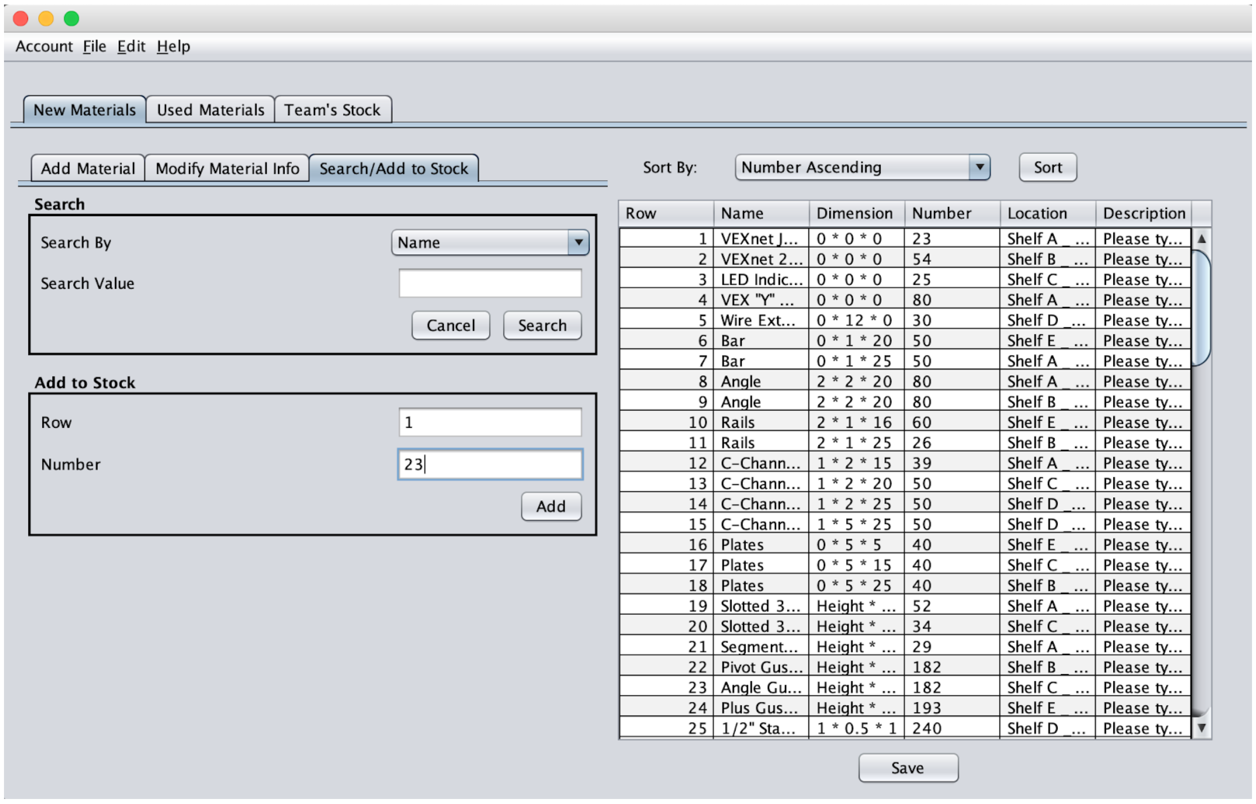
After the show button is pressed, the newMaterial ArrayList will get the material requested by the users. setText() method is used to print the information on the proper text fields.



```
private void modifyButton_NMMouseReleased(java.awt.event.MouseEvent evt) {
    // modify the information of certain material in the arrayList.
    if(!rowTextField_NM.getText().equals(null)){
        JOptionPane.showMessageDialog(null, "Please type in the row number at first", "Error", ERROR_MESSAGE);
        //this will output an warning if the users hasn't choosen the row number
    }

    int row = Integer.parseInt(rowTextField_NM.getText()) - 1;
    newMaterial.get(row).setName(nameTextField_NM1.getText());
    newMaterial.get(row).setDimension(heightTextField_NM1.getText() + "*" +
    lengthTextField_UM1.getText() + "*" + widthTextField_UM1.getText());
    newMaterial.get(row).setNumber(Integer.parseInt(numberTextField_NM1.getText()));
    newMaterial.get(row).setLocation(shelfComboBox_NM1.getSelectedItem() + " _ " + floorComboBox_NM1.getSelectedItem() + "");
    newMaterial.get(row).setDescription(descriptionTextArea_NM1.getText());
    refreshNewMaterialTable(newMaterial);//refresh the display table to the updated version
    //fileSave_NM(newMaterial);//save the arrayList to the local files
}
```

After users modify the information of certain material, they will press the modify button to input the updated information. Set methods are used to resign the value contained in the object and refreshNewMaterial() is used to redisplay the updated lists.

<u>Adding Material to the team's own stock.</u>



Row: 1

Number: 23

Sort By: Number Ascending

Row	Name	Dimension	Number	Location	Description
1	VEXnet J...	0 * 0 * 0	23	Shelf A _ ...	Please ty...
2	VEXnet 2...	0 * 0 * 0	54	Shelf B _ ...	Please ty...
3	LED Indic...	0 * 0 * 0	25	Shelf C _ ...	Please ty...
4	VEX "Y" ...	0 * 0 * 0	80	Shelf A _ ...	Please ty...
5	Wire Ext...	0 * 12 * 0	30	Shelf D _...	Please ty...
6	Bar	0 * 1 * 20	50	Shelf E _ ...	Please ty...
7	Bar	0 * 1 * 25	50	Shelf A _ ...	Please ty...
8	Angle	2 * 2 * 20	80	Shelf A _ ...	Please ty...
9	Angle	2 * 2 * 20	80	Shelf B _ ...	Please ty...
10	Rails	2 * 1 * 16	60	Shelf E _ ...	Please ty...
11	Rails	2 * 1 * 25	26	Shelf B _ ...	Please ty...
12	C-Chann...	1 * 2 * 15	39	Shelf A _ ...	Please ty...
13	C-Chann...	1 * 2 * 20	50	Shelf C _ ...	Please ty...
14	C-Chann...	1 * 2 * 25	50	Shelf D _...	Please ty...
15	C-Chann...	1 * 5 * 25	50	Shelf D _...	Please ty...
16	Plates	0 * 5 * 5	40	Shelf E _ ...	Please ty...
17	Plates	0 * 5 * 15	40	Shelf C _ ...	Please ty...
18	Plates	0 * 5 * 25	40	Shelf B _ ...	Please ty...
19	Slotted 3...	Height * ...	52	Shelf A _ ...	Please ty...
20	Slotted 3...	Height * ...	34	Shelf C _ ...	Please ty...
21	Segment...	Height * ...	29	Shelf A _ ...	Please ty...
22	Pivot Gus...	Height * ...	182	Shelf B _ ...	Please ty...
23	Angle Gu...	Height * ...	182	Shelf C _ ...	Please ty...
24	Plus Gus...	Height * ...	193	Shelf E _ ...	Please ty...
25	1/2" Sta...	1 * 0.5 * 1	240	Shelf D _...	Please ty...

This program allows team members to choose their wanted materials from the new material ArrayList and add them to their own team stocks ArrayList. After inputting the row number and the number needed, the add button will be pressed.

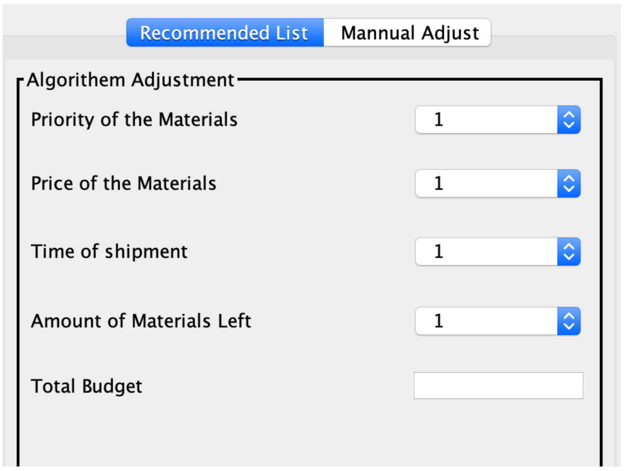
```java
    private void addAddButton_UMMouseReleased(java.awt.event.MouseEvent evt) {
        //add the wanted materials to the team stock.
        int row = Integer.parseInt(rowAddTextField_UM.getText()) - 1;//the reason this is
        String name = usedMaterial.get(row).getName();
        String dimension = usedMaterial.get(row).getDimension();
        int numberLeft = Integer.parseInt(numberAddTextField_UM.getText());
        int totalNumber = numberLeft;
        String newUsed = "Used Material";
        TeamStock add = new TeamStock(name, dimension, numberLeft, totalNumber, newUsed);
        teamStock.add(add);

        usedMaterial.get(row).setNumber(usedMaterial.get(row).getNumber() - totalNumber);
        refreshUsedMaterialTable(usedMaterial);//update the data

        searchComboBox_UM.setSelectedIndex(0);
        numberAddTextField_UM.setText(null);
    }
```

This code will first figure out the address of the wanted material in the ArrayList. Based on the input number, the number of materials left in the newMaterial ArrayList will automatically be decreased. At the same time, the same amount of materials will be added to 8861A's own team stock.

Auto generator for a recommended list of purchase

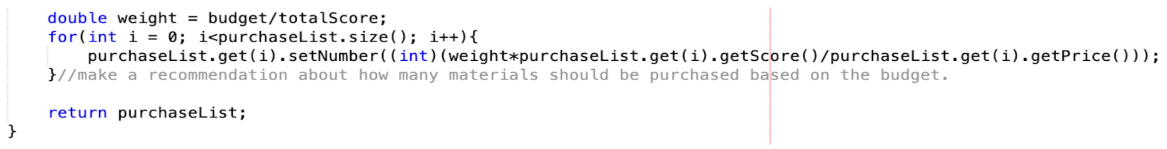
```java
public ArrayList<MaterialPurchase> autoGenerate(ArrayList<MaterialPurchase> materialPurchase, ArrayList<NewMaterial> newMaterial,
int wPriority, int wPrice,  int wTime, int wAmountLeft, int budget){
    //this is an algorithems which will generate a list of recommended materials for purchase based on the weight of each factor.
    ArrayList<MaterialPurchase> purchaseList =  new ArrayList<MaterialPurchase>();
    for(int i = 0; i<materialPurchase.size(); i++){
        //give each factor a score. The adjustment is made by multiplying the weight by a constant
        double sPriority = (double)wPriority/materialPurchase.get(i).getPriority()*3;
        double sPrice = (double)wPrice/materialPurchase.get(i).getPrice()*70;
        double sTime = (double)wTime/materialPurchase.get(i).getTimeOfShipment()*60;
        double sAmountLeft = (double)wAmountLeft/amountLeft(newMaterial, materialPurchase.get(i).getName())*40;
        materialPurchase.get(i).setScore(sPriority + sPrice + sTime + sAmountLeft);
    }

    materialPurchase = sort(materialPurchase);//sort the material purchase list in the descending sequence based on the score.
    for(int i = 0; i<10; i++){
        purchaseList.add(materialPurchase.get(i));//the top 10 will be selected and recommended to the teachers.
    }
    double totalScore= 0.0;
    for(int i = 0; i<purchaseList.size(); i++){
        totalScore += purchaseList.get(i).getScore();
    }
```

To maximize the use of budget, an algorithm is created to automatically generate a list of recommended materials for purchase. This allows teachers to maximize the use of the budget. materialPurchase ArrayList contains various information related to the materials. Through the GUI page shown on the right side, Users can adjust the weight of each information. The algorithms will take the weight into account and calculate a score for each factor. The sum of the score represents the importance of the materials. The top 10 are used to create a list of recommended materials.



Algorithem Adjustment

Priority of the Materials	1
Price of the Materials	1
Time of shipment	1
Amount of Materials Left	1
Total Budget	

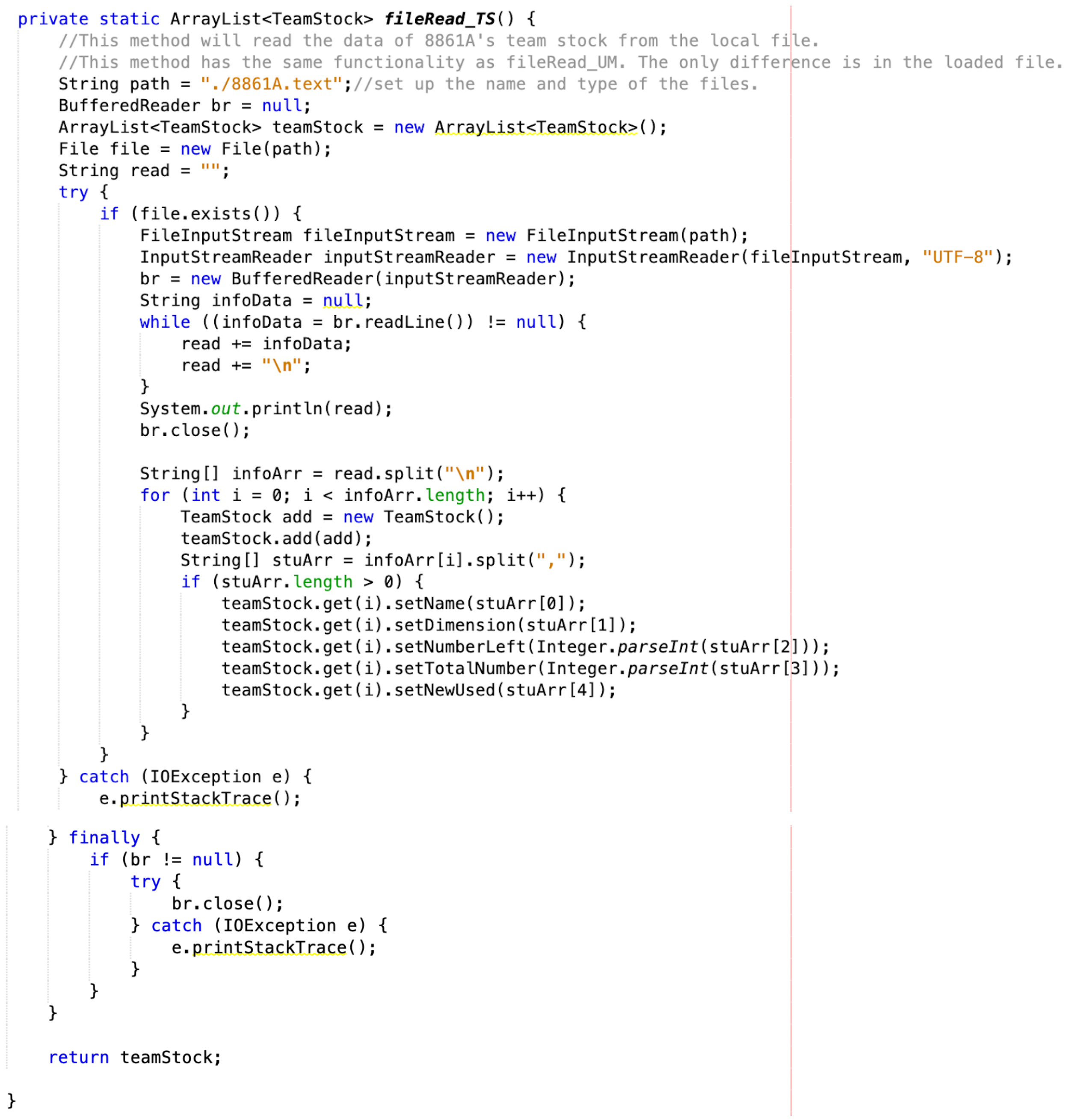


```
    double weight = budget/totalScore;
    for(int i = 0; i<purchaseList.size(); i++){
        purchaseList.get(i).setNumber((int)(weight*purchaseList.get(i).getScore()/purchaseList.get(i).getPrice()));
    }//make a recommendation about how many materials should be purchased based on the budget.

    return purchaseList;
}
```

Algorithms are used to calculate the proper budget for each material based on the total budget. To make teachers make wiser decisions. The recommended amount for each material is calculated by dividing the material’s budget with its price.

Files Management:

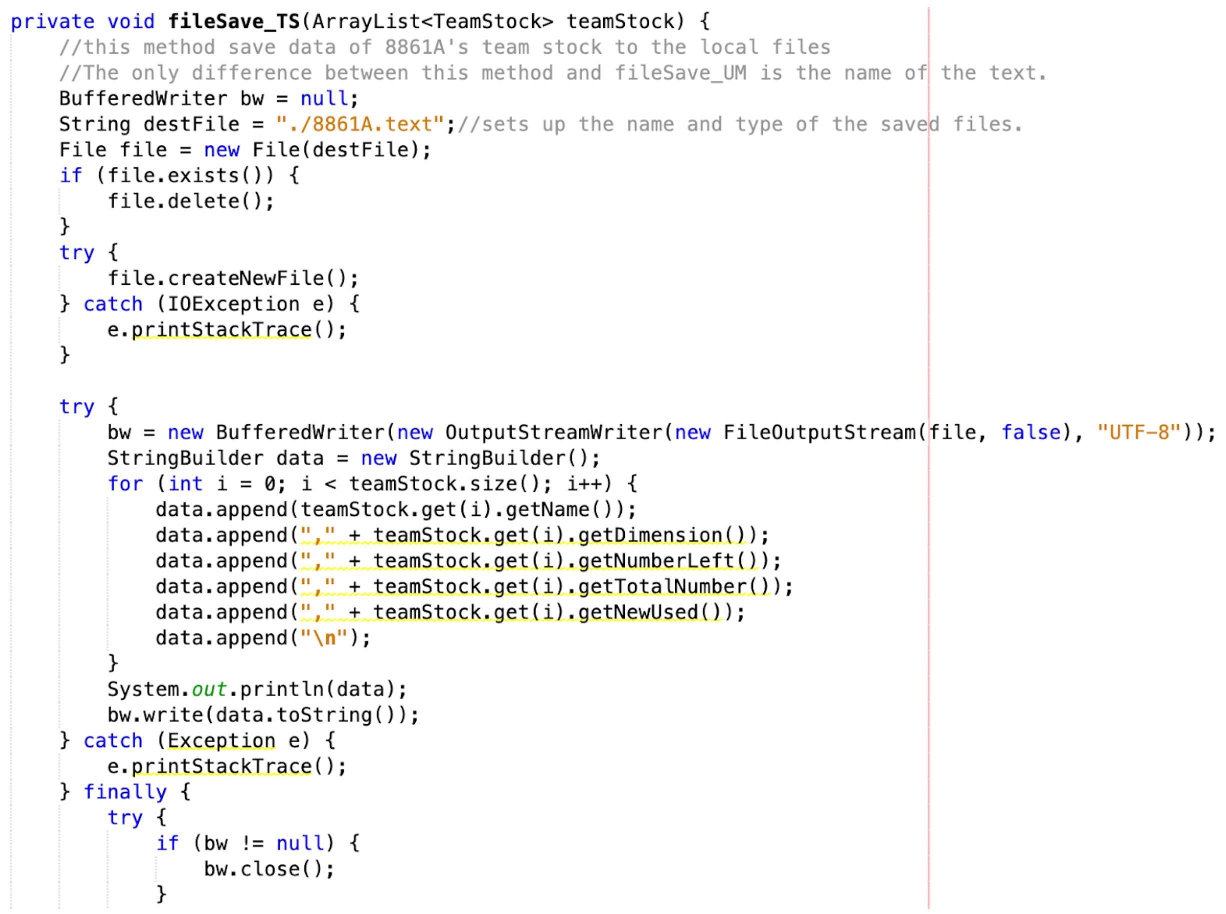
- File Read



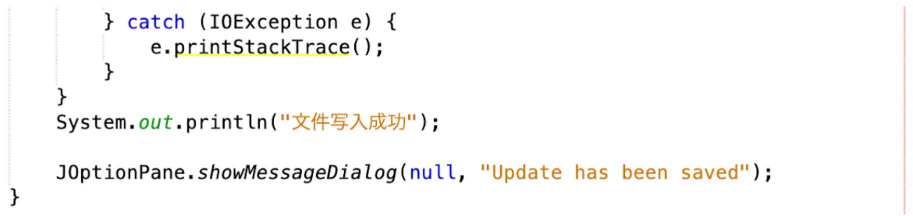
```
private static ArrayList<TeamStock> fileRead_TS() {
    //This method will read the data of 8861A's team stock from the local file.
    //This method has the same functionality as fileRead_UM. The only difference is in the loaded file.
    String path = "./8861A.text";//set up the name and type of the files.
    BufferedReader br = null;
    ArrayList<TeamStock> teamStock = new ArrayList<TeamStock>();
    File file = new File(path);
    String read = "";
    try {
        if (file.exists()) {
            FileInputStream fileInputStream = new FileInputStream(path);
            InputStreamReader inputStreamReader = new InputStreamReader(fileInputStream, "UTF-8");
            br = new BufferedReader(inputStreamReader);
            String infoData = null;
            while ((infoData = br.readLine()) != null) {
                read += infoData;
                read += "\n";
            }
            System.out.println(read);
            br.close();

            String[] infoArr = read.split("\n");
            for (int i = 0; i < infoArr.length; i++) {
                TeamStock add = new TeamStock();
                teamStock.add(add);
                String[] stuArr = infoArr[i].split(",");
                if (stuArr.length > 0) {
                    teamStock.get(i).setName(stuArr[0]);
                    teamStock.get(i).setDimension(stuArr[1]);
                    teamStock.get(i).setNumberLeft(Integer.parseInt(stuArr[2]));
                    teamStock.get(i).setTotalNumber(Integer.parseInt(stuArr[3]));
                    teamStock.get(i).setNewUsed(stuArr[4]);
                }
            }
        }
    } catch (IOException e) {
        e.printStackTrace();

    } finally {
        if (br != null) {
            try {
                br.close();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    return teamStock;

}
```

This is the file read algorithms for the team stock of team 8861A. By using external packages BufferReader and File, I can successfully load the data from the local files to the program. After using the split() method, the data is separated into pieces that are then assigned to designated variables in the teamStock ArrayList. I have made a brave decision that

- File Save



```java
private void fileSave_TS(ArrayList<TeamStock> teamStock) {
    //this method save data of 8861A's team stock to the local files
    //The only difference between this method and fileSave_UM is the name of the text.
    BufferedWriter bw = null;
    String destFile = "./8861A.text";//sets up the name and type of the saved files.
    File file = new File(destFile);
    if (file.exists()) {
        file.delete();
    }
    try {
        file.createNewFile();
    } catch (IOException e) {
        e.printStackTrace();
    }

    try {
        bw = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(file, false), "UTF-8"));
        StringBuilder data = new StringBuilder();
        for (int i = 0; i < teamStock.size(); i++) {
            data.append(teamStock.get(i).getName());
            data.append("," + teamStock.get(i).getDimension());
            data.append("," + teamStock.get(i).getNumberLeft());
            data.append("," + teamStock.get(i).getTotalNumber());
            data.append("," + teamStock.get(i).getNewUsed());
            data.append("\n");
        }
        System.out.println(data);
        bw.write(data.toString());
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            if (bw != null) {
                bw.close();
            }
```



```java
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    System.out.println("文件写入成功");

    JOptionPane.showMessageDialog(null, "Update has been saved");
}
```

This file save method will save 8861A's team stock back to local files. This is achieved by using the append() method from the external package BufferedReader. To make the data readable for the fileRead method, a comma is used to separate each data pice and "\n" is used to represent the end of each line.

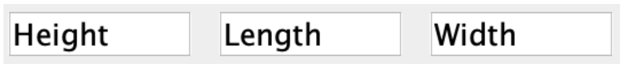
Word Count: 510

User interface/Error handling and Notification

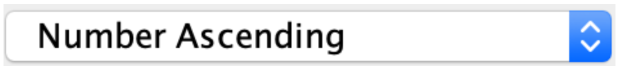
GUI work
Use of GUI: Allowing users to interact with the program in a virtually friendly way which makes users easier to master the functions in the program.

GUI components used

- JTextfields: For users to enter information

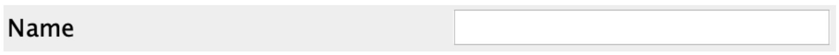


- JCombobox: Providing various options for users to select

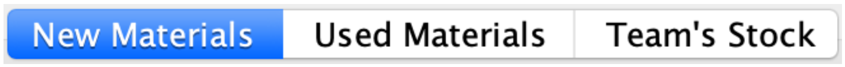


- JButtons: For users to click to conduct certain actions such as saving the updated data.

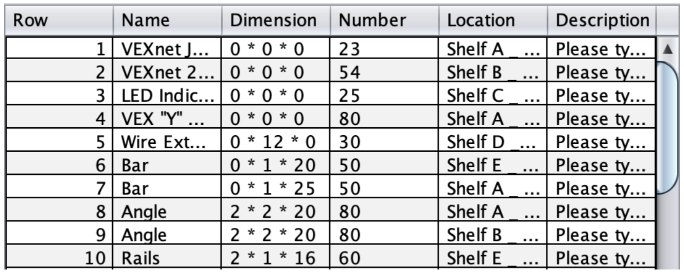
JLabels: Notify which type of information should users put in.



- JTabbedPane: Organizes user interface and make users easier to access different functions located on different pages

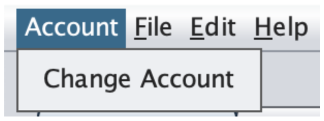


- JTable: Used to display large amounts of data in an organized way.

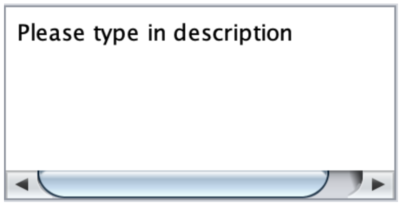


Row	Name	Dimension	Number	Location	Description
1	VEXnet J...	0 * 0 * 0	23	Shelf A _ ...	Please ty...
2	VEXnet 2...	0 * 0 * 0	54	Shelf B _ ...	Please ty...
3	LED Indic...	0 * 0 * 0	25	Shelf C _ ...	Please ty...
4	VEX "Y" ...	0 * 0 * 0	80	Shelf A _ ...	Please ty...
5	Wire Ext...	0 * 12 * 0	30	Shelf D _ ...	Please ty...
6	Bar	0 * 1 * 20	50	Shelf E _ ...	Please ty...
7	Bar	0 * 1 * 25	50	Shelf A _ ...	Please ty...
8	Angle	2 * 2 * 20	80	Shelf A _ ...	Please ty...
9	Angle	2 * 2 * 20	80	Shelf B _ ...	Please ty...
10	Rails	2 * 1 * 16	60	Shelf E _ ...	Please ty...

- JMenuItem: A shortcut for some commonly used functions such as logging out.



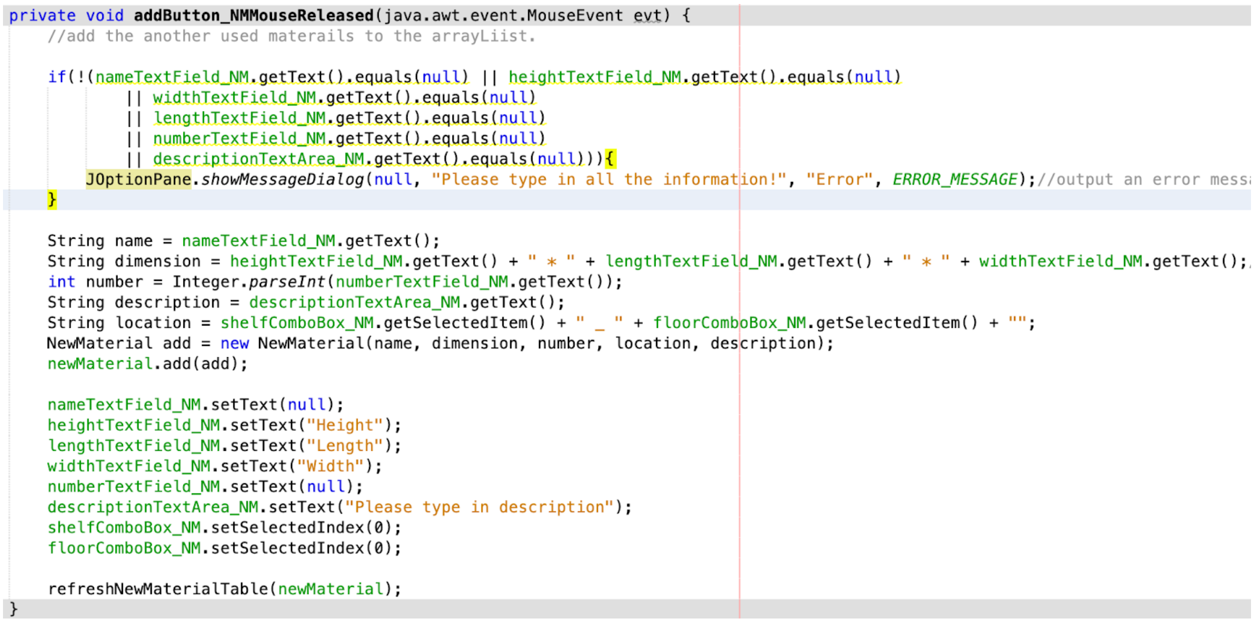
- JTextArea: Allow users to input long description to aspecific material



Word Count: 0

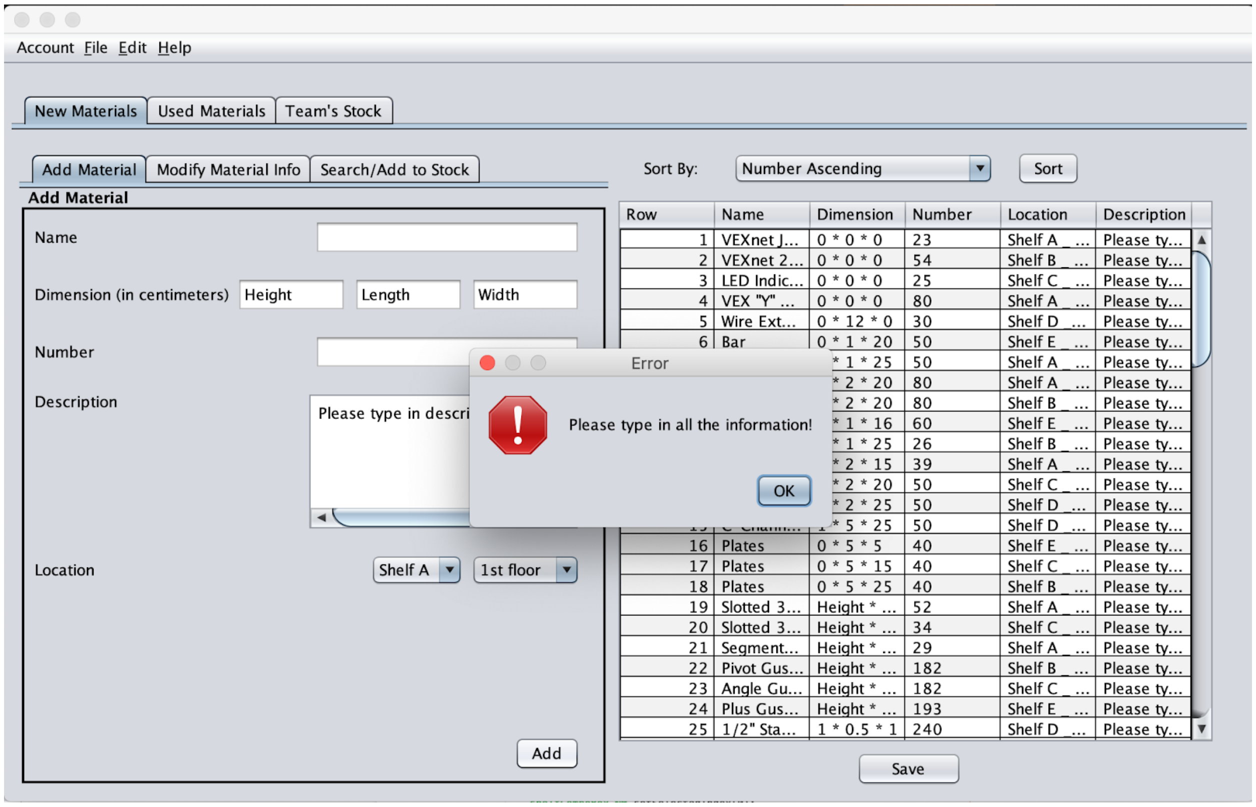
Error handling

It is common for users to make mistakes when they are working with the program. To create a better working experience, an external package JOptionPane is used.



```
private void addButton_NMMouseReleased(java.awt.event.MouseEvent evt) {
    //add the another used materails to the arrayLiist.

    if(!(nameTextField_NM.getText().equals(null) || heightTextField_NM.getText().equals(null)
            || widthTextField_NM.getText().equals(null)
            || lengthTextField_NM.getText().equals(null)
            || numberTextField_NM.getText().equals(null)
            || descriptionTextArea_NM.getText().equals(null))){
        JOptionPane.showMessageDialog(null, "Please type in all the information!", "Error", ERROR_MESSAGE);//output an error mess
    }

    String name = nameTextField_NM.getText();
    String dimension = heightTextField_NM.getText() + " * " + lengthTextField_NM.getText() + " * " + widthTextField_NM.getText();
    int number = Integer.parseInt(numberTextField_NM.getText());
    String description = descriptionTextArea_NM.getText();
    String location = shelfComboBox_NM.getSelectedItem() + " _ " + floorComboBox_NM.getSelectedItem() + "";
    NewMaterial add = new NewMaterial(name, dimension, number, location, description);
    newMaterial.add(add);

    nameTextField_NM.setText(null);
    heightTextField_NM.setText("Height");
    lengthTextField_NM.setText("Length");
    widthTextField_NM.setText("Width");
    numberTextField_NM.setText(null);
    descriptionTextArea_NM.setText("Please type in description");
    shelfComboBox_NM.setSelectedIndex(0);
    floorComboBox_NM.setSelectedIndex(0);

    refreshNewMaterialTable(newMaterial);
}
```

This method allows users to add new material to the list by inputting a series of information. To make sure that users input the correct value. An if-statement is created before the method starts executing. If users fail to fill in some necessary information, a warning sign will jump out and the data won't be added to the table.

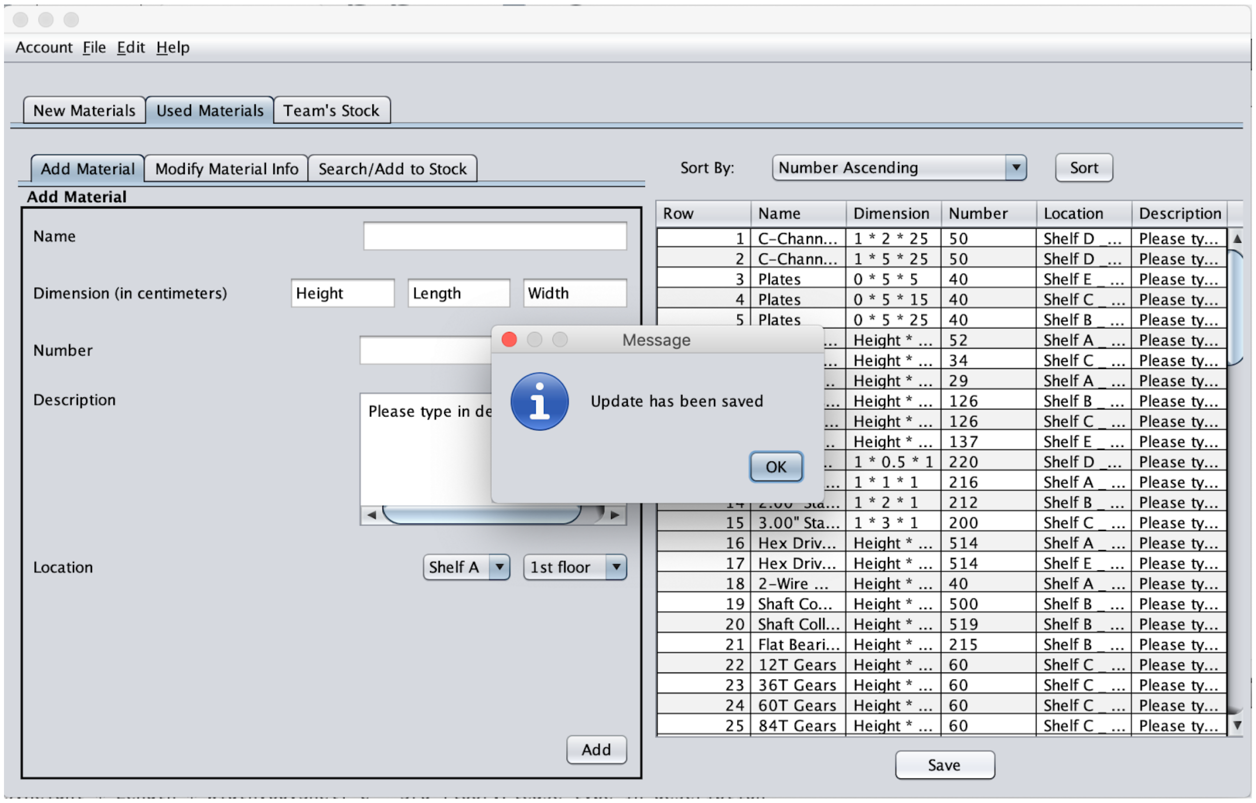


Row	Name	Dimension	Number	Location	Description
1	VEXnet J...	0 * 0 * 0	23	Shelf A _ ...	Please ty...
2	VEXnet 2...	0 * 0 * 0	54	Shelf B _ ...	Please ty...
3	LED Indic...	0 * 0 * 0	25	Shelf C _ ...	Please ty...
4	VEX "Y" ...	0 * 0 * 0	80	Shelf A _ ...	Please ty...
5	Wire Ext...	0 * 12 * 0	30	Shelf D _...	Please ty...
6	Bar	0 * 1 * 20	50	Shelf E _ ...	Please ty...
[illegible]	[illegible]	* 1 * 25	50	Shelf A _ ...	Please ty...
[illegible]	[illegible]	* 2 * 20	80	Shelf A _ ...	Please ty...
[illegible]	[illegible]	* 2 * 20	80	Shelf B _ ...	Please ty...
[illegible]	[illegible]	* 1 * 16	60	Shelf E _ ...	Please ty...
[illegible]	[illegible]	* 1 * 25	26	Shelf B _ ...	Please ty...
[illegible]	[illegible]	* 2 * 15	39	Shelf A _ ...	Please ty...
[illegible]	[illegible]	* 2 * 20	50	Shelf C _ ...	Please ty...
[illegible]	[illegible]	* 2 * 25	50	Shelf D _...	Please ty...
15	C Chann...	1 * 5 * 25	50	Shelf D _...	Please ty...
16	Plates	0 * 5 * 5	40	Shelf E _ ...	Please ty...
17	Plates	0 * 5 * 15	40	Shelf C _ ...	Please ty...
18	Plates	0 * 5 * 25	40	Shelf B _ ...	Please ty...
19	Slotted 3...	Height * ...	52	Shelf A _ ...	Please ty...
20	Slotted 3...	Height * ...	34	Shelf C _ ...	Please ty...
21	Segment...	Height * ...	29	Shelf A _ ...	Please ty...
22	Pivot Gus...	Height * ...	182	Shelf B _ ...	Please ty...
23	Angle Gu...	Height * ...	182	Shelf C _ ...	Please ty...
24	Plus Gus...	Height * ...	193	Shelf E _ ...	Please ty...
25	1/2" Sta...	1 * 0.5 * 1	240	Shelf D _...	Please ty...

<u>Notification</u>

```
JOptionPane.showMessageDialog(null, "Update has been saved");
```

To inform users that a certain action has successfully conducted, a notification will be sent to the users. For example, in fileSave() method, after saving is completed, a window will pop out to represent the completion of the saving.



Add Material

Name

Dimension (in centimeters) Height Length Width

Number

Description Please type in de

Location Shelf A 1st floor

Sort By: Number Ascending

Row	Name	Dimension	Number	Location	Description
1	C-Chann...	1 * 2 * 25	50	Shelf D _...	Please ty...
2	C-Chann...	1 * 5 * 25	50	Shelf D _...	Please ty...
3	Plates	0 * 5 * 5	40	Shelf E _ ...	Please ty...
4	Plates	0 * 5 * 15	40	Shelf C _ ...	Please ty...
5	Plates	0 * 5 * 25	40	Shelf B _ ...	Please ty...
	...	Height * ...	52	Shelf A _ ...	Please ty...
	...	Height * ...	34	Shelf C _ ...	Please ty...
	..	Height * ...	29	Shelf A _ ...	Please ty...
	...	Height * ...	126	Shelf B _ ...	Please ty...
	...	Height * ...	126	Shelf C _ ...	Please ty...
	..	Height * ...	137	Shelf E _ ...	Please ty...
	..	1 * 0.5 * 1	220	Shelf D _...	Please ty...
	...	1 * 1 * 1	216	Shelf A _ ...	Please ty...
14	2.00" Sta...	1 * 2 * 1	212	Shelf B _ ...	Please ty...
15	3.00" Sta...	1 * 3 * 1	200	Shelf C _ ...	Please ty...
16	Hex Driv...	Height * ...	514	Shelf A _ ...	Please ty...
17	Hex Driv...	Height * ...	514	Shelf E _ ...	Please ty...
18	2-Wire ...	Height * ...	40	Shelf A _ ...	Please ty...
19	Shaft Co...	Height * ...	500	Shelf B _ ...	Please ty...
20	Shaft Coll...	Height * ...	519	Shelf B _ ...	Please ty...
21	Flat Beari...	Height * ...	215	Shelf B _ ...	Please ty...
22	12T Gears	Height * ...	60	Shelf C _ ...	Please ty...
23	36T Gears	Height * ...	60	Shelf C _ ...	Please ty...
24	60T Gears	Height * ...	60	Shelf C _ ...	Please ty...
25	84T Gears	Height * ...	60	Shelf C _ ...	Please ty...

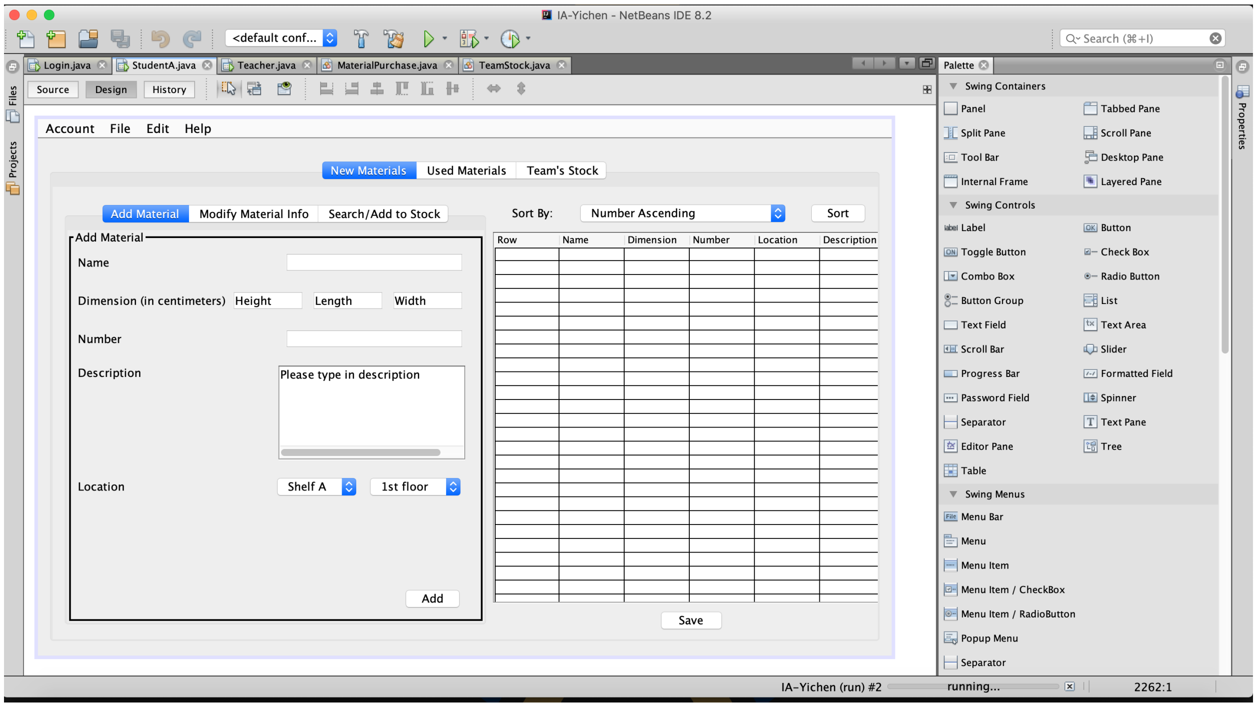
Message

Update has been saved

Word Count: 124

Software Tool Used (NetBeans)

NetBeans, an integrated development environment for java, was used in the development and coding of the application. NetBeans software was chosen because it has developed a user-friendly GUI interface to help me develop a Database. Besides that, it's pre-set template GUI page can also help me get a quick start. What's more, since it is one of the most widely used IDEs for Java, It is convenient for me to seek help through the internet which largely speeds up my process of development



Word Count: 83

Total Word Count: 951