Criterion C: Development

Standard techniques used:

```
public class commonClass {
    public static String[] equipmentColumn = {"equipmentId","category","name","equipmentStatus","currentStatus"};
    public static String[] equipmentHiddenColumn = {"sourcebuying", "timeBought", "lastReview", "itRoom",
                                "specificStoredLocation", "maxAllowDuration", "extensionTime", "timesOfExtension"};
    public static String[] borrowColumn = {"recordId","equipmentId","returnDate","lateOrNot","userId","isStudent",
                                "name","class","classNumber"};
    public static String[] borrowHiddenColumn = {"timesOfExtensionUsed", "feedback", "transactionTime", "dateStart",
                                "dateEnd", "timesBorrowRecord", "gender", "lateRecord", "numberOfBorrowedItems"
                                , "borrowAuthorization"};
    public static String[] addingequipmentColumn = {"category", "equipmentStatus", "itRoom"};
    public static String[] category = {"Book","Microbit","Airblock","Microbit","Microbit1"};
    public static String[] equipmentStatus = {"damaged","to be checked","not opened to borrow","open to borrow"};
    public static String[] itRoom = {"605","607A","607B"};
    public static String[] usermasterColumn = {"userId", "isStudent", "name", "class", "lateRecord", "borrowAuthorization"};
    public static String[] usermasterHiddenColumn = {"numberOfBorrowedItems", "timesBorrowRecord", "classNumber", "gender"};
}
```

1. **Arrays**

- Arrays 2D

```
String data[][] = new String[numberOfRows][columnlength];
```

Figure: 2D Arrays used in creating table

2. **User-defined objects**

```
String userNameValue = jTextFieldUserId.getText();
String borrowId = jTextFieldBorrowId.getText();
```

Figure: Storing data from MySQL to an array

3. **Objects as data records**

```
for(int n=0;n<numberOfRows;n++){
    model.addRow(new Object[]{data[n][0]});
    for(int count = 1; count < columnlength; count++){
      model.setValueAt(data[n][count], n, count);
    }
}
```

Figure: Creating a set of records (table) from different objects

4. **Simple selection (if-else)**

```
if (Integer.parseInt(data[x][y])==0){
    data[x][y] = "free to borrow";
}
else{data[x][y] = "borrowed";}
```

5. **Complex selection (if with multiple conditions)**

```
if (!(jTextFieldFeedback.getText()).equals("") && jCheckBox1.isSelected()){
    PreparedStatement updateQuestionMaster = myCon.prepareStatement("insert into questionmaster (userId, "
            + "equipmentId, questionLocation, feedback, isDamage) values (?, ?, 0, ?, 1);");
    updateQuestionMaster.setString(1, jTextFieldUserId.getText());
    updateQuestionMaster.setString(2, jTextFieldBorrowId.getText());
    updateQuestionMaster.setString(3, jTextFieldFeedback.getText());
    updateQuestionMaster.executeUpdate();
    System.out.println("inserted feedback");
    PreparedStatement changeEquipmentStatusToDamage = myCon.prepareStatement("Update equipmentmaster set "
            + "equipmentStatus = 0 where equipmentId = ?;");
    changeEquipmentStatusToDamage.setString(1, jTextFieldBorrowId.getText());
    changeEquipmentStatusToDamage.executeUpdate();
}
```

6. **Loops**

```
for(int n=0;n<numberOfRows;n++){
  model.addRow(new Object[]{data[n][0],data[n][1],data[n][2],data[n][3],data[n][4],data[n][5]});
  }
```

Figure: for loop used in inserting table

7. Nested Loops

```
for (int x=0; myRs5.next();x++){
    for (int y =0; y < columnlength ; y++){
        data[x][y] = myRs5.getString(equipmentColumn[y]);
    }
}
```

Figure: for loop used in extracting data from MySQL

8. User-defined methods

```
public void updateborrowRecordfromUserMaster(){
    try{
        Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
        Statement myStmt = myCon.createStatement();
        myStmt.executeUpdate("Update borrowRecord set isStudent = (Select isStudent from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set name = (Select name from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set class = (Select class from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set classNumber = (Select classNumber from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set timesBorrowRecord = (Select gender from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set lateRecord = (Select lateRecord from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set gender = (Select gender from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set borrowAuthorization = (Select borrowAuthorization from usermaster where borrowRecord.userId = usermaster.userId);");
        myStmt.executeUpdate("Update borrowRecord set numberOfBorrowedItems = (Select numberOfBorrowedItems from usermaster where borrowRecord.userId = usermaster.userId);");
    }
    catch(Exception exc){
    exc.printStackTrace();
     }
}
```

9. User-defined methods with parameters

```
public void comboBox1and2(String[] equipmentHiddenColumn, String[] equipmentColumn){
    int hiddencolumnlength  = equipmentHiddenColumn.length;
    int columnlength  = equipmentColumn.length;
    jComboBox1.insertItemAt("", 0);
    for (int count = 0; count < hiddencolumnlength; count++){
        jComboBox1.addItem(equipmentHiddenColumn[count]);
    }
    jComboBox2.insertItemAt("", 0);
    for (int count = 1; count < columnlength; count++){ //the reason of being 1 is to let it ignore the equipmentId
        jComboBox2.addItem(equipmentColumn[count]);
    }
}
```

10. Sorting

```
jTable1.setAutoCreateRowSorter(true);
```

11. File i/o

```
try{
Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
Statement myStmt = myCon.createStatement();
ResultSet myRs6 = myStmt.executeQuery("select count(*) from usermaster;");
while(myRs6.next()){
    numberOfRows = Integer.parseInt (new String (myRs6.getString("count(*)")));
    break;
    }
```

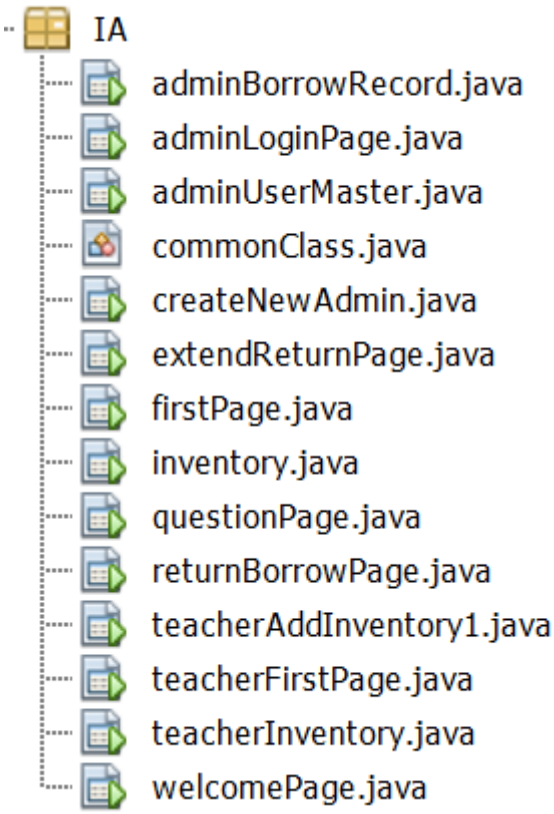
12. Use of additional libraries

```java
import static IA.commonClass.usermasterColumn;
import static IA.commonClass.usermasterHiddenColumn;
import java.awt.Dimension;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Statement;
import javax.swing.JOptionPane;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.table.DefaultTableModel;
```

13. Use of sentinels or flags

```java
for (int x=0; x < numberOfRows ; x++){
    intColumnStatus = Integer.parseInt(data[x][y]);
    if (intColumnStatus==0){
        data[x][y] = "teacher";
    }
    if (intColumnStatus==1){
        data[x][y] = "student";
    }
}
```

All Classes:



Files that started with “teachers” and “admin” on the figure above are referring to the classes used by the admin. Except for commonClass.java, all other classes are used by the users. commonClass.java is a public class that stores arrays that are imported into different files.

Other techniques used:

1. MySQL commands used

- Connection with MySQL Server

```
try{
    Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
    Statement myStmt;
}
catch(Exception exc){
    exc.printStackTrace();
}
```

It is connected to MySQL Server through the client “myuser” with the password “1222”

- Update
  - Update with fixed MySQL statement

```
myStmt.executeUpdate("Update borrowRecord set isStudent = (Select isStudent from "
        + "usermaster where borrowRecord.userId = usermaster.userId);");
```

In this example, it gets the “isStudent” column from “usermaster” and update it to the “isStudent” column in “borrowRecord”.

  - Update with Unfixed MySQL statement

```
PreparedStatement updateisRead = myCon.prepareStatement("Update questionmaster set isRead = 1 where questionNumber = ?;");
updateisRead.setString(1, number);
updateisRead.executeUpdate();
```

This used of Update allows user to put its variable into the MySQL statement, the position of “?” by using setString() or setInt().

  - Update with Unfixed MySQL statement with embedding String directly into it

```
PreparedStatement updateValue = myCon.prepareStatement("Update equipmentmaster set "+ equipmentColumn[y] + " = ? where equipmentId = ? ;");
updateValue.setString(1, c);
updateValue.setInt(2, a1);
updateValue.executeUpdate();
```

This is also using PreparedStatement, but as equipmentColumn[y] is embedded in the PreparedStatement instead of putting it in the setString(), setInt() function. This is used as there are some unknown reason causes setString() not functioning.

- Delete

```
PreparedStatement updateValue = myCon.prepareStatement("Delete from equipmentmaster where equipmentId = ? ;");
updateValue.setString(1, deleteEquipemntId);
updateValue.executeUpdate();
```

- Select

```
ResultSet myRs5 = myStmt.executeQuery("select * from equipmentmaster;");
for (int x=0; myRs5.next();x++){
    for (int y =0; y < columnlength ; y++){
        data[x][y] = myRs5.getString(equipmentColumn[y]);
    }
}
```

The program selects all (*) from “equipmentmaster” by using a for loop and storing it into the array.

- **Why using MySQL**

  During developing my program I chose between using MySQL and Excel, but I finally chose to use MySQL because of the following reasons

  - High Performance
  - A lot of support online connecting with Java
  - Easy to handle and connect
  - I don't need much arithmetic functions

- **Set up MySQL**

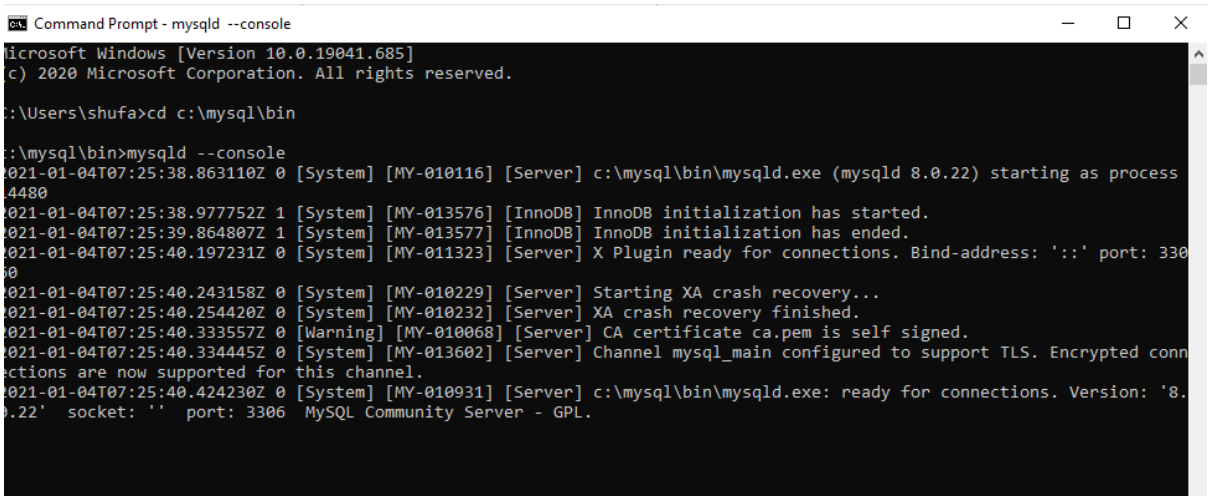


Figure: Turning on the local server

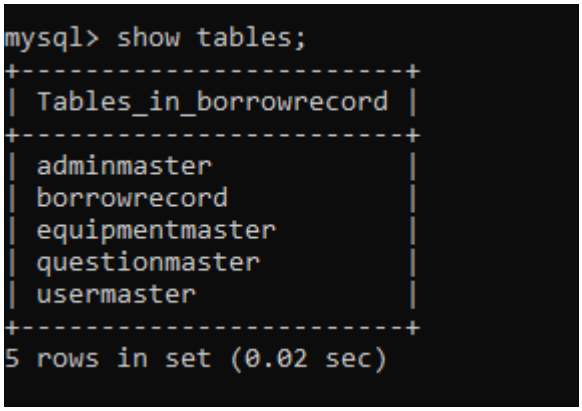


Figure: 5 MySQL tables are created to store information on all of these areas.

- Adminmaster: stores admin.
- Borrowrecord: helps store each of the records created.
- Equipmentmaster: stores all equipment.
- Questionmaster: stores all questions and feedback created by users.
- Usermaster: stores all users, which includes students and teachers.

2. JTable

equipmentId	category	name	equipmentStatus	currentStatus
1	Book	Micro:Bit Basics: A first guide fo...	damaged	free to borrow
2	Book	The Art of LEGO MINDSTORMS...	open to borrow	free to borrow
6	Book	The Lego Mindstorms Ev3 Lab...	open to borrow	free to borrow
7	Microbit	Microbit V1	not opened to borrow	free to borrow
8	Airblock	Airblock V1	open to borrow	free to borrow
11	Microbit	Microbit V1	not opened to borrow	free to borrow
21	Airblock	AirblockV2	damaged	free to borrow
22	Book	Getting Started with Arduino Se...	open to borrow	free to borrow

Figure: Table demo created in the “teacherInventory” (admin equipmentmaster)

JTable is used to list all the information that the admin wants to gather from the database. As there are too many columns in each of the tables in the database, to ensure clear data is presented. A filter function is needed to be added. Thus, to cooperate with the filter function, certain arrangements have to be made.

equipmentId	category	name	equipmentStatus	currentStatus
1	Book	Micro:Bit Basics: A first guide fo...	damaged	free to borrow
2	Book	The Art of LEGO MINDSTORMS...	open to borrow	free to borrow
6	Book	The Lego Mindstorms Ev3 Lab...	open to borrow	free to borrow
7	Microbit	Microbit V1	not opened to borrow	free to borrow
8	Airblock	Airblock V1	open to borrow	free to borrow
11	Microbit	Microbit V1	not opened to borrow	free to borrow
21	Airblock	AirblockV2	damaged	free to borrow
22	Book	Getting Started with Arduino Se...	open to borrow	free to borrow

Add Column

Delete Column

Figure: Initial table created in the “teacherInventory” (admin equipmentmaster)

equipmentId	name	currentStatus	timeBought	itRoom	maxAllowDuration	extensionTime
1	Micro:Bit Basics: A fir...	free to borrow	2018-01-14	605	14 days	7 days
2	The Art of LEGO MIN...	free to borrow	2018-09-24	605	14 days	7 days
6	The Lego Mindstorm...	free to borrow	2018-09-24	605	14 days	7 days
7	Microbit V1	free to borrow	2018-03-24	605	14 days	7 days
8	Airblock V1	free to borrow	2019-01-20	607B	-1 days	7 days
11	Microbit V1	free to borrow		607A	14 days	7 days
21	AirblockV2	free to borrow		607B	14 days	7 days
22	Getting Started with A...	free to borrow		605	14 days	7 days

Add Column

Delete Column

Figure: Filtered table created in the teacherInventory (admin equipmentmaster) where some columns are added and deleted

Comparing the two figures above, column “timebought”, “itRoom”, “maxAllowctionDuration” and “extensionTime” is added. While column “category”, “equipmentStatus” is deleted.

Below will be the procedures in creating a table, I am using the table created in the page teacherInventory as an example.

<u>Steps in creating a table:</u>

1. Import useful arrays from the commonClass
2. Connecting to MySQL, getting essential table data
3. Creating a 2d array in storing the data
4. Using nested loop to store data into 2d array
5. Converting data stored in database to more understandable words
6. Adding Column names from the array
7. Creating table by storing data[][] in it. Add a new row, then set values into other columns
8. Visualize the table, adding ScrollPane in it.

Step 1: Import useful arrays from commonClass

```
import static IA.commonClass.equipmentColumn;
import static IA.commonClass.equipmentHiddenColumn;
```

```
public class commonClass {
    public static String[] equipmentColumn = {"equipmentId","category","name","equipmentStatus","currentStatus"};
    public static String[] equipmentHiddenColumn = {"sourcebuying", "timeBought", "lastReview", "itRoom",
                                "specificStoredLocation", "maxAllowDuration", "extensionTime", "timesOfExtension"};
```

equipmentColumn[] stores all the column names that the admin wants to check at the first moment. equipmentHiddenColumn[] stores the rest of the column names that are not presented at first.

Step 2: Connecting to MySQL, getting the number of rows

Step 3: Creating a 2d array in storing the data

Step 4: Using nested loop to store data into 2d array

```
public void addRowsInTable(){
    try{
    Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
    Statement myStmt = myCon.createStatement();
    ResultSet myRs6 = myStmt.executeQuery("select count(*) from equipmentmaster;");
    while(myRs6.next()){
        numberOfRows = Integer.parseInt (myRs6.getString("count(*)"));
        break;
        }
        columnlength = equipmentColumn.length;
        String data[][] = new String[numberOfRows][columnlength];
    ResultSet myRs5 = myStmt.executeQuery("select * from equipmentmaster;");
    for (int x=0; myRs5.next();x++){
        for (int y =0; y < columnlength ; y++){
            data[x][y] = myRs5.getString(equipmentColumn[y]);
        }
    }
```

In step 2, the line *count(*)* counts the total amount of lines in the “equipmentmaster”, which is saved into “numerOfRows”, and “columnlength” storing the number of columns there is. By using these, it will tell us how large the 2d array created should be to produce the whole table.

```java
int intEquipmentStatus;
for (int y= 0; y< columnlength; y++){
    if ("equipmentStatus".equals(equipmentColumn[y])){
        for (int x=0; x < numberOfRows ; x++){
            intEquipmentStatus = Integer.parseInt(data[x][y]);
            if (intEquipmentStatus==0){
                data[x][y] = "damaged";
            }
            if (intEquipmentStatus==1){
                data[x][y] = "to be checked";
            }
            if (intEquipmentStatus==2){
                data[x][y] = "not opened to borrow";
            }
            if (intEquipmentStatus==3){
                data[x][y] = "open to borrow";
            }
        }
    }
    if ("currentStatus".equals(equipmentColumn[y])){
        for (int x=0;  x < numberOfRows  ; x++){
            if (Integer.parseInt(data[x][y])==0){
                data[x][y] = "free to borrow";
            }
            else{data[x][y] = "borrowed";}
        }
    }
    if ("maxAllowDuration".equals(equipmentColumn[y])){
        for (int x=0; x < numberOfRows ; x++){
            data[x][y] = data[x][y] + " days";
        }
    }
    if ("extensionTime".equals(equipmentColumn[y])){
        for (int x=0; x < numberOfRows ; x++){
            data[x][y] = data[x][y] + " days";
        }
    }
    if ("timesofExtension".equals(equipmentColumn[y])){
        for (int x=0; x < numberOfRows ; x++){
            data[x][y] = data[x][y] + " times";
        }
    }
}
```

Step 5: Converting data stored in database to more

Readable words

Step 6: Adding Column names from the array

```
DefaultTableModel model;
    model = (DefaultTableModel)jTable1.getModel();
    for (int count = 0; count < columnlength; count++){
        model.addColumn(equipmentColumn[count]);
    }
```

Step 7: Creating table by storing data[][] in it. Add a new row, then setValues into other coloumns

```
for(int n=0;n<numberOfRows;n++){
    model.addRow(new Object[]{data[n][0]});
    for(int count = 1; count < columnlength; count++){
      model.setValueAt(data[n][count], n, count);
    }
}
```

In Step 7, the reason for separating addRow() and setValue() is because if I put all my data into addRow at once, it will not allow me to have a flexible change in the number of columns. Hence, I choose to add the key-value first, the "equipemtnId" in this case. Which the key-value no matter in what situation it won't be filtered out. And then input my other values by using setValues() into the row afterward.

Using this method, it would allow me to only changing the array equipmentColumn[], then refresh the table, a new table would be created with different columns can be generated. Further techniques will be explained in later parts.

Step 8: Visualize the table, adding ScrollPane in it.

```
JTable table = new JTable(model);
 table.setPreferredScrollableViewportSize(new Dimension(450,63));
 table.setFillsViewportHeight(true);

 JScrollPane js=new JScrollPane(table);
 js.setVisible(true);
 add(js);
```

3. JList

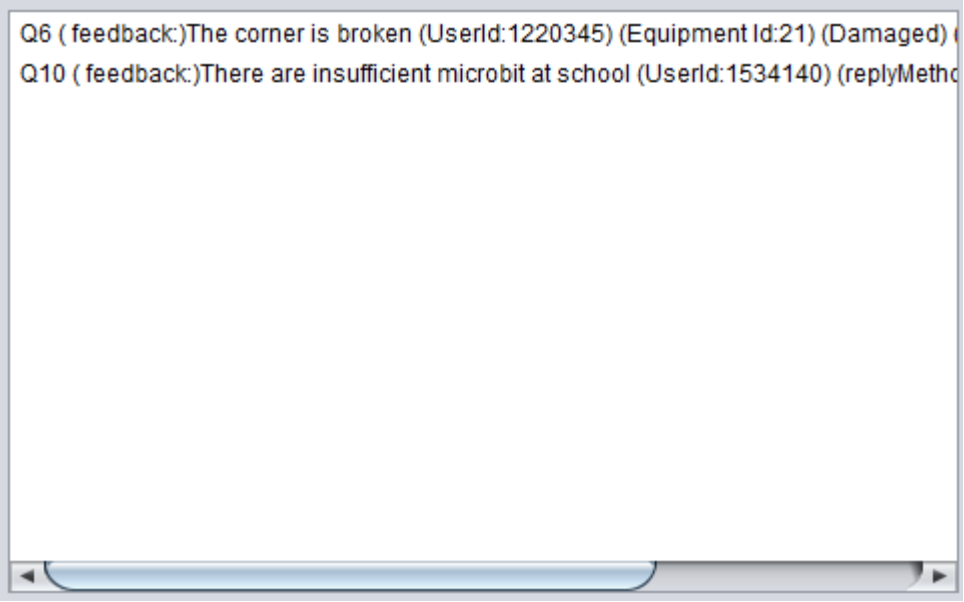


Figure: List demo created in the “teacherInventory” (admin equipmentmaster)

JList is used to list out all the questions asked in a simple way. The benefit of using a list over the table is that it can list out all the information in a very simple form. It’s used here as the “feedback” itself is the most important part of all the information in the “questionmaster”, “e.g. There are insufficient Microbit at school”. However, other information might also be useful for the admin to understand, such as the “UserId”, “replyMethod”, “EquipmentId” and etc. Therefore using a JList and listing all of these information out would be an easy and nice way.

<u>Steps in creating a List:</u>

1. Connecting to MySQL, getting essential table data
2. Creating a 2d array in storing the data
3. Create a DefaultListModel
4. Storing data into the array, then store the array into the list
5. Visualize the list, adding ScrollPane in it

Step 1: Connecting to MySQL, getting the number of rows which fits the condition

Step 2: Creating a 2d array in storing the data

```
try{
    Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
    Statement myStmt = myCon.createStatement();

        int numberOfRows = 0;
        ResultSet myRs6 = myStmt.executeQuery("select count(*) from questionmaster where isRead = 0;");
        while(myRs6.next()){
            numberOfRows = Integer.parseInt (new String (myRs6.getString("count(*)")));
            break;
        }
        String data[][]= new String[numberOfRows][6];
```

In step 1, the condition here is whether isRead = 0. This enables it only shows questions that are not

Step 3: Create a DefaultListModel

```
//https://www.experts-exchange.com/questions/27401815/javax-swing-JList-3-cannot-be-cast-to-javax-swing-DefaultListModel.html
        DefaultListModel dflm = new DefaultListModel();
        jList1.setModel(dflm);
        ListModel lm = jList1.getModel();
        DefaultListModel model = (DefaultListModel)lm;
```

resolved.

Step 4: Storing data into the array, then store the array into the list

Store the data from MySQL into the array

addElement if it suits the following conditions

Store the data from MySQL into the array

addElement if it suits the following conditions

```
for (int x=0; myRs5.next();x++){
        if (myRs5.getString("questionLocation").equals("0")){
            data[x][0] = myRs5.getString("questionnumber");
            data[x][1] = myRs5.getString("Feedback");
            data[x][2] = myRs5.getString("userId");
            data[x][3] = myRs5.getString("equipmentId");
            data[x][5] = "Return Page";
            if(myRs5.getString("isDamage").equals("0")){
                model.addElement("Q" + data[x][0] + " ( feedback:" + ")"+ data[x][1] + " (UserId:" + data[x][2] + ")" +
                        " (Equipment Id:" + data[x][3] + ")" +  " (" + data[x][5] + ")");
            }
            else if(myRs5.getString("isDamage").equals("1")){
                data[x][4] = "Damaged";
                model.addElement("Q" + data[x][0] + " ( feedback:" + ")"+ data[x][1] + " (UserId:" + data[x][2] + ")" +
                        " (Equipment Id:" + data[x][3] + ")" + " (" + data[x][4] + ")" +  " (" + data[x][5] + ")");
            }
        }
        if (myRs5.getString("questionLocation").equals("1")){
            data[x][0] = myRs5.getString("questionnumber");
            data[x][1] = myRs5.getString("Feedback");
            data[x][2] = myRs5.getString("userId");
            data[x][3] = myRs5.getString("replyMethod");
            data[x][4] = "Question Page";
            if((data[x][2])== null){
                System.out.println("is empty");
                model.addElement("Q" + data[x][0] + " ( feedback:" + ")"+data[x][1] + " (replyMethod:" + data[x][3] + ")"
                        +  " (" + data[x][4] + ")");
            }
            else {
                model.addElement("Q" + data[x][0] + " ( feedback:" + ")"+data[x][1] + " (UserId:" + data[x][2] + ")" +
                        " (replyMethod:" + data[x][3] + ")" + " (" + data[x][4] + ")");
            }
        }
}
```

As there is two different question panel for users to ask questions, “user return page” and “user question page”. And each section has different questions asked. Hence, they are organized separately. This is done by using “if statement”. Data from MySQL are then stored into the array and shows into the List.

Step 5: Visualize the list, adding ScrollPane in it

```
JList list = new JList(model);
    JScrollPane js=new JScrollPane(list);
    js.setVisible(true);
    add(js);
}
```

4. Other techniques used

JcomboBox

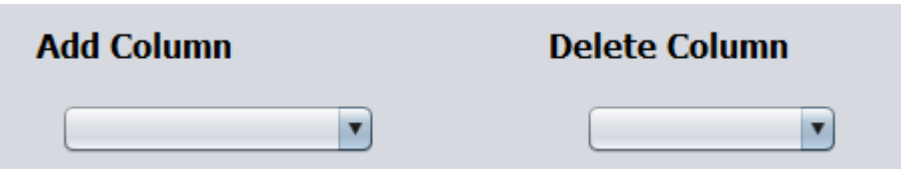


Figure: ComboBox examples

```
jComboBox1.insertItemAt("", 0);
for (int count = 0; count < hiddencolumnlength; count++){
    jComboBox1.addItem(equipmentHiddenColumn[count]);
}
```

This is used when there is selection within a certain and fixed amount of items

JButton

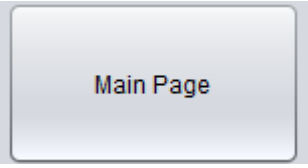


Figure: Button examples

```
private void jButtonReturning1ActionPerformed(java.awt.event.ActionEvent evt) {
  new adminBorrowRecord().setVisible(true);
  dispose();
}
```

```
private void jButtonReadActionPerformed(java.awt.event.ActionEvent evt) {
    //https://www.w3resource.com/java-exercises/basic/java-basic-exercise-69.php
    String selectedValue = jList1.getSelectedValue(); // read the JList
```

It is sometimes used as traveling to different pages, it is also used to ask the system to read after user filled in information in the appropriate boxes or did some selection.

JOptionPane

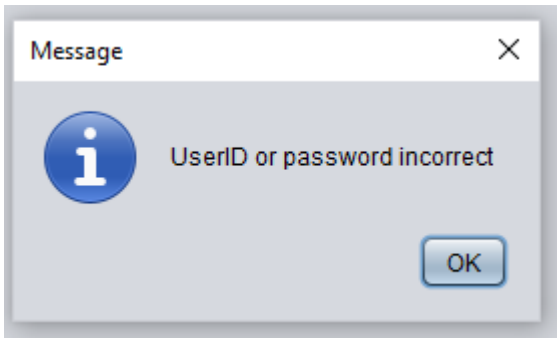


Figure: JOptionPane example.

```
JOptionPane.showMessageDialog(null,"You typed wrong Change States Column Name");
```

```
JOptionPane.showMessageDialog(null, "Succeed in returning");
```

```
JOptionPane.showMessageDialog(null, "You have a total of " + noOflateRecord + " late records. "
        + "If you return late one more time, you would only be be allowed to borrow "
        + "after finding responsible computer sceince teacher to get the autorization of borrowing" );
```

It is used when I want to notify the user the processing in the system is successful or not, whether there are any errors, or tell the user some information after processing.

Getting the current date

```
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
Date date1 = new Date(System.currentTimeMillis());
```

Field	Type	Null	Key	Default	Extra
recordId	int	NO	PRI	NULL	auto_increment
equipmentId	int	YES		NULL	
returnDate	timestamp	YES		NULL	
timesOfExtensionUsed	tinyint	YES		0	
feedback	varchar(200)	YES		NULL	
transactionTime	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
dateStart	date	YES		curdate()	DEFAULT_GENERATED
dateEnd	date	YES		(curdate() + interval 14 day)	DEFAULT_GENERATED

Figure: Current date and current timestamp in MySQL

Curdate() and current_timestamp is used in MySQL. This is used to get the current date or current time.

Calling from commonClass

```
public class commonClass {
    public static String[] equipmentColumn = {"equipmentId","category","name","equipmentStatus","currentStatus"};
    public static String[] equipmentHiddenColumn = {"sourcebuying", "timeBought", "lastReview", "itRoom",
                                "specificStoredLocation", "maxAllowDuration", "extensionTime", "timesOfExtension"};
    public static String[] borrowColumn = {"recordId","equipmentId","returnDate","lateOrNot","userId","isStudent",
                                "name","class","classNumber"};
    public static String[] borrowHiddenColumn = {"timesOfExtensionUsed", "feedback", "transactionTime", "dateStart",
                                "dateEnd", "timesBorrowRecord", "gender", "lateRecord", "numberOfBorrowedItems"
                                , "borrowAuthorization"};
    public static String[] addingequipmentColumn = {"category", "equipmentStatus", "itRoom"};
    public static String[] category = {"Book","Microbit","Airblock","Microbit","Microbitl"};
    public static String[] equipmentStatus = {"damaged","to be checked","not opened to borrow","open to borrow"};
    public static String[] itRoom = {"605","607A","607B"};
    public static String[] usermasterColumn = {"userId", "isStudent", "name", "class", "lateRecord", "borrowAuthorization"};
    public static String[] usermasterHiddenColumn = {"numberOfBorrowedItems", "timesBorrowRecord", "classNumber", "gender"};
}
```

Figure: Stores array from different tables

```
import static IA.commonClass.borrowColumn;
import static IA.commonClass.borrowHiddenColumn;
```

Figure: commonClass imported into adminBorrowRecord

Classes:

User Side:

1. borrow page (Inventory.java)
2. borrow extension page (extendReturnPage.java)
3. main page (firstPage.java)
4. question page (questionPage.java)
5. return page (returnBorrowPage.java)
6. instruction page (welcomePage.java)

Admin Side:

1. check borrow record page (adminBorrowPage.java)
2. login page (adminLoginPage.java)
3. check users page (adminUserMaster.java)
4. create new admin page (createNewAdmin.java)
5. add new equipment page (teacherAddInventory1.java)
6. main page (teacherFirstPage.java)
7. check / change / delete equipment page (teacherInventory.java)

Others:

1. commonClass.java

User Side:

Borrow Page

Major Functions

- Table of equipment
- Borrow function.

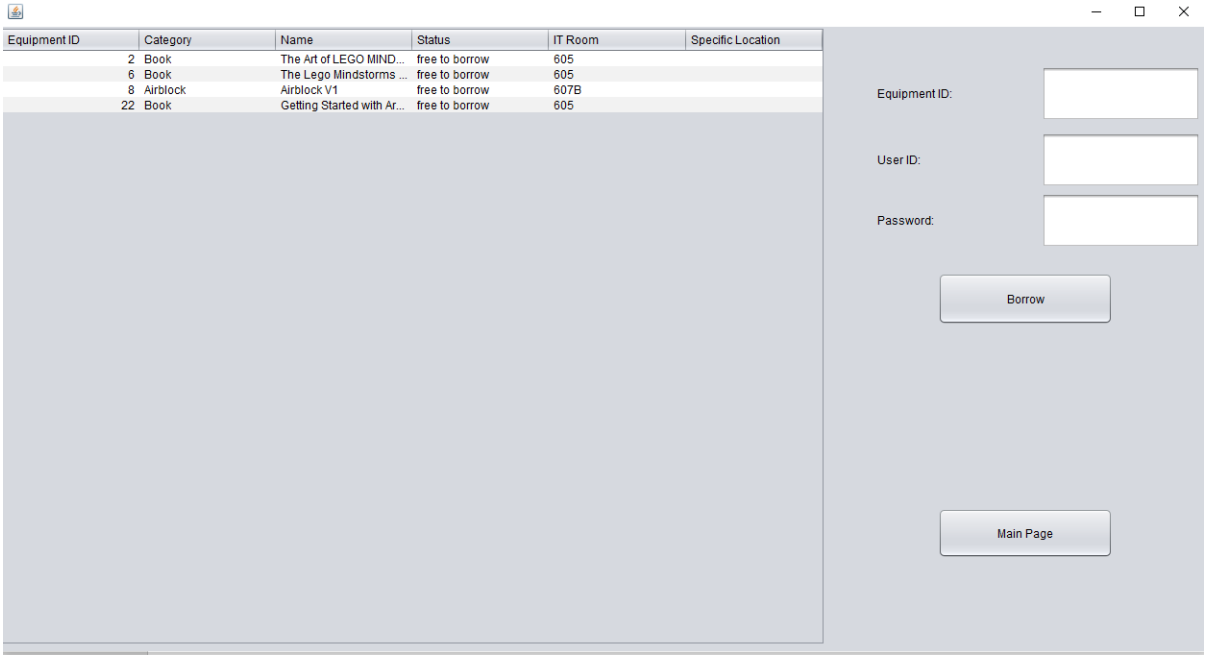


Figure: Table Demo

Table of Equipment

The user side has a table of equipment, which shows all the possible equipment that is available to be borrowed. I used a JTable to demonstrate it. The procedures in creating a JTable and the use of JTable is mentioned in the earlier sections, though what I am describing here has two minor differences:

1. User Borrow Page do not need column customization, hence do not need to reference to common-class
2. It needs to have some extra conditions.

```
ResultSet myRs6 = myStmt.executeQuery("select count(*) from equipmentmaster where equipmentStatus =3;");
ResultSet myRs5 = myStmt.executeQuery("select equipmentId, category, name, currentStatus, "
        + "itRoom, specificStoredLocation from equipmentmaster where equipmentStatus =3;");
```

As Introduce in the above, “equipmentStatus” in “equipmentmaster” stores different states of equipment. Only if an equipment’s “equipmentStatus” = 3, it could only be “open / free to borrow” for users.

Connect to mysql

Get count(*) to set the numberOfRows

Creating a 2d array to store all the values

Overwrite some values to make it more readable

Create the table with printing all the values in the 2d array. Set it visible and scrollable

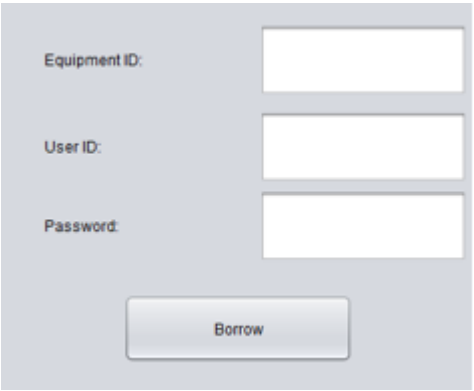
```
public void addRowsInTable(){
  try{
//connect to mysql
Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
int numberOfRows =0;
Statement myStmt = myCon.createStatement();
//find the number of rows
ResultSet myRs6 = myStmt.executeQuery("select count(*) from equipmentmaster where equipmentStatus =3;");
 while(myRs6.next()){
    numberOfRows = Integer.parseInt (myRs6.getString("count(*)"));
    break;
    }
 //creating the 2d array
    String data[][]= new String[numberOfRows][6];
ResultSet myRs5 = myStmt.executeQuery("select equipmentId, category, name, currentStatus, "
        + "itRoom, specificStoredLocation from equipmentmaster where equipmentStatus =3;");
// save the data into array
for (int x = 0;myRs5.next();x++){
    data[x][0]= myRs5.getString("equipmentId");
    data[x][1]= myRs5.getString("category");
    data[x][2]= myRs5.getString("name");
      if (Integer.parseInt(myRs5.getString("currentStatus")) == 0){
            data[x][3]="free to borrow";
          }
      else{
            data[x][3]= "borrowed";
          }
    data[x][4]= myRs5.getString("itRoom");
    data[x][5]= myRs5.getString("specificStoredLocation");
}
    //creating table
    DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
  for(int n=0;n<numberOfRows;n++){
    model.addRow(new Object[]{data[n][0],data[n][1],data[n][2],data[n][3],data[n][4],data[n][5]});
    }
   JTable table = new JTable(model);
    table.setPreferredScrollableViewportSize(new Dimension(450,63));
    table.setFillsViewportHeight(true);

    JScrollPane js=new JScrollPane(table);
    js.setVisible(true);
    add(js);
    }
```

Figure: all the programs in creating table.

```
public inventory() {
      initComponents();
      addRowsInTable();
      jTable1.setAutoCreateRowSorter(true);//Done adding sort function
}
```

Figure: The start of this page

Borrow function



Borrow function allows users to borrow items. After pressing the button, the program will automatically

read the details in “equipmentId”, “UserId” and “password”.

Connect to MySQL. Get values from usermaster and save them to corresponding places

Check if the valued typed in in useId and password is same as the one gets from usermaster

Check if the user doesn’t have the right to borrow and if the user currently borrowed more than 2 items. If yes, print those messages

Check if the equipmentId is typed correctly, continue if correct.

Update the equipment master, save that the equipment is borrowed

Get the maximum allow of time borrowed.

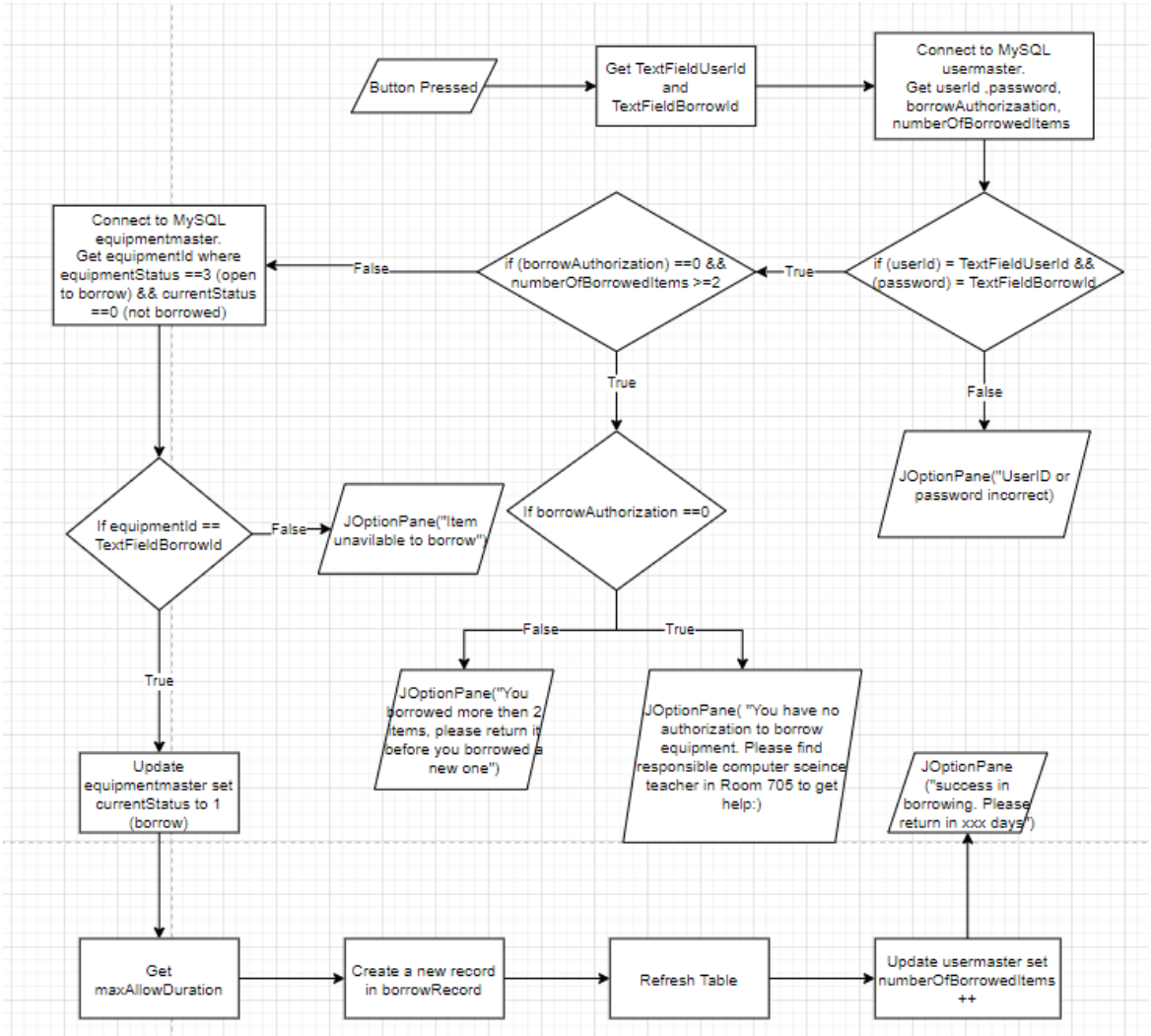
Create a new borrowrecord, print all the details in it

```
private void jButtonBorrowActionPerformed(java.awt.event.ActionEvent evt) {


try{
    String userNameValue = jTextFieldUserId.getText();
    String borrowId = jTextFieldBorrowId.getText();
    Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
    Statement myStmt;
    myStmt = myCon.createStatement();
    ResultSet myRs5 = myStmt.executeQuery("select userId,password, borrowAuthorization, numberOfBorrowedItems from usermaster");
    String userId = null;
    int password = 0;
    int activateKey=0;
    int borrowAuthroization =0;
    int numberOfBorrowedItems = 0;
        while (myRs5.next()){
            userId = myRs5.getString("userId");
            password = myRs5.getInt("password");
            if (userId.equals(userNameValue) && password == Integer.parseInt(new String(jPasswordField.getPassword()))){
                activateKey=1;
                borrowAuthroization = myRs5.getInt("borrowAuthorization");
                numberOfBorrowedItems = myRs5.getInt("numberOfBorrowedItems");
                break;
            }
        }
if (activateKey==1){
    if (borrowAuthroization == 0 || numberOfBorrowedItems >= 2){
        if (borrowAuthroization == 0 )
            JOptionPane.showMessageDialog(null, "You have no authroization to borrow equipment. Please find responsible computer"
                    + " sceince teacher in Room 705 to get help" );
        else if (numberOfBorrowedItems >= 2)
            JOptionPane.showMessageDialog(null, "You borrowed more then 2 items, please return it before you borrowed a new one" );
    }
    else{
    ResultSet myRs6 = myStmt.executeQuery("select equipmentId from equipmentmaster where equipmentStatus = 3 && currentStatus = 0");
    int integerBorrowId = Integer.parseInt(borrowId);
    int activateKey2 =0;
    while (myRs6.next()){
        if (myRs6.getInt("equipmentId")== integerBorrowId){
            activateKey2 = 1;
            break;
        }
    }
    if (activateKey2 == 1){
            PreparedStatement currentStatusToBorrow = myCon.prepareStatement("UPDATE equipmentmaster SET currentStatus = 1 where "
                    + "equipmentID = ?;");
            currentStatusToBorrow.setString(1, borrowId);
            currentStatusToBorrow.executeUpdate();
            PreparedStatement borrowTimeLimit = myCon.prepareStatement("Select maxAllowDuration from equipmentmaster where "
                    + "equipmentID = ?;");
            borrowTimeLimit.setString(1, borrowId);
            ResultSet myRs8 = borrowTimeLimit.executeQuery();
            int maxAllowDuration = 0;
            while (myRs8.next()){
                maxAllowDuration = myRs8.getInt("maxAllowDuration");
                break;
            }
            PreparedStatement saveToBorrowRecord = myCon.prepareStatement("insert into borrowRecord( equipmentId, userId, dateEnd)values"
                    + " ( ? , ?, curdate()+interval ? day);");
            saveToBorrowRecord.setInt(1, integerBorrowId);
            saveToBorrowRecord.setString(2, userNameValue);
            saveToBorrowRecord.setInt (3, maxAllowDuration);
            saveToBorrowRecord.execute();
```

Add table and print table	
Update the user's number of borrowed items by increasing 1.	
Error messages.	

```
                    DefaultTableModel tableModel = (DefaultTableModel) jTable1.getModel();
                    try{
                        tableModel.getDataVector().removeAllElements();
                        tableModel.fireTableDataChanged();
                    }
                     catch(Exception E){
                        JOptionPane.showMessageDialog(null,E.getMessage());
                    }
                    addRowsInTable();
                    PreparedStatement updateTimesBorrowRecord = myCon.prepareStatement("Update usermaster set timesBorrowRecord = "
                            + "timesBorrowRecord+1 where userId = ?;");
                    updateTimesBorrowRecord.setString(1, userId);
                    updateTimesBorrowRecord.executeUpdate();
                    PreparedStatement updateNumberOfBorrowedItems = myCon.prepareStatement("Update usermaster set numberOfBorrowedItems = "
                            + "numberOfBorrowedItems+1 where userId = ?;");
                    updateNumberOfBorrowedItems.setString(1, userId);
                    updateNumberOfBorrowedItems.executeUpdate();
                    JOptionPane.showMessageDialog(null,"success in borrowing. Please return it in " + maxAllowDuration + " days");
                }
                else{JOptionPane.showMessageDialog(null,"Item unavailable to be borrowed");};
            }
            }
            else{
                JOptionPane.showMessageDialog(null,"UserID or password incorrect");
            }

        }


    catch(Exception exc){
        exc.printStackTrace();
    }

        }
```

This program for the button is a little bit difficult to understand, but it simply did what is shown in the flow chart below.



The reason for using a lot of "activateKey" is because I found that I can't run nested while loop when using "ResultSet" selecting data from the MySQL table. This might not be working as MySQL can only execute commands sequentially, which two different commends, different "resultset" can't be executed at the same

time. Hence, I used “activateKey” as the way in saving, whether the “resultset” successfully obtains the result or not from the while loop, and continue by using if (activateKey = 1) {} else {} operation. Though the consequence would be difficult to read and understand.

Different possible errors are also thought thoroughly in the program and by using JOptionPane provide pop-up boxes to tell users what’s detected wrong.

Return Page

Major functions:

- Return Equipment
- Provide Feedback about the equipment

This part is for the user to return the equipment that is borrowed, and provide feedback to the admin if needed.

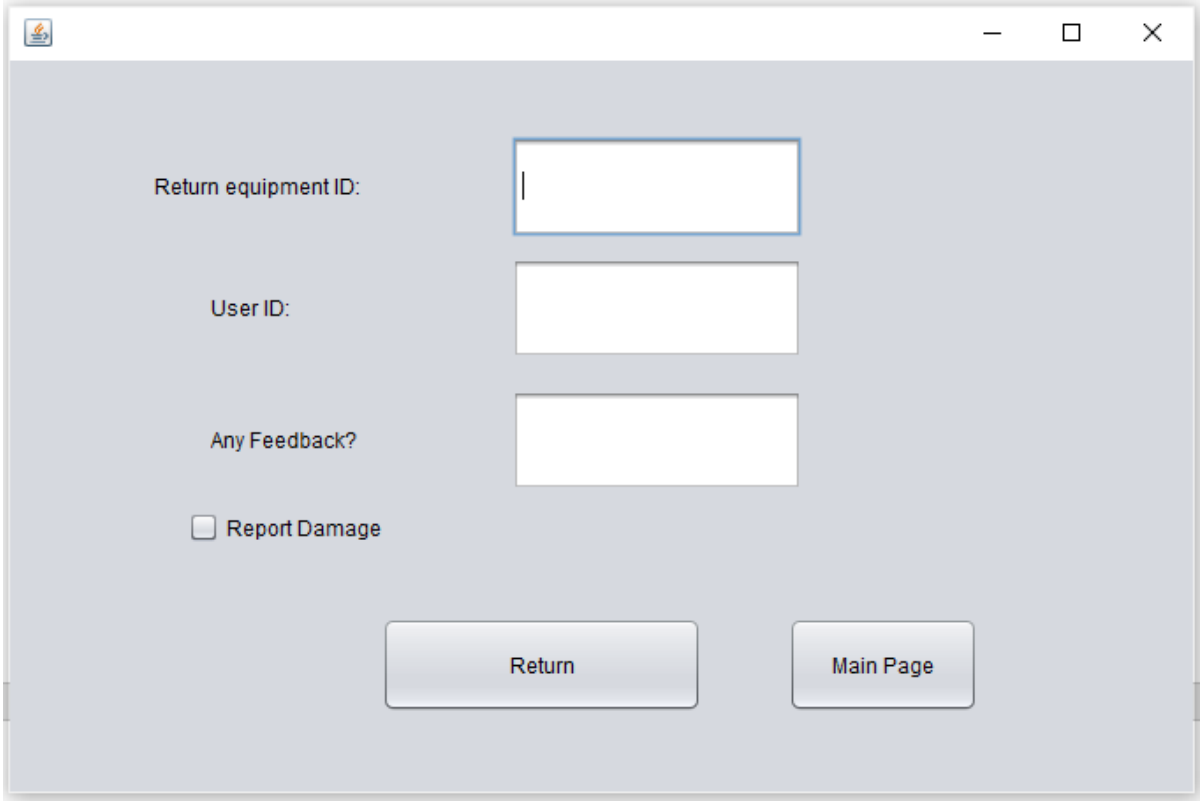


Figure: Return Page Demo

Execute after the button “return” is pressed

Connect to MySQL

Get the data from borrowRecord.

Update the return time

Update equipmentmaster sets the equipment is returned.

Update usermaster set the number of borrowed item --

Sends the feedback if the user write some feedback or tick the box

```
private void jButtonBorrowActionPerformed(java.awt.event.ActionEvent evt) {

    try{
        String userNameValue = jTextFieldUserId.getText();
        String borrowId = jTextFieldBorrowId.getText();
        Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
        Statement myStmt;
        myStmt = myCon.createStatement();

        int activateKey = 0;
        String dateEnd = null;
        int recordId = 0;
        int userId = 0;
        ResultSet myRs5= myStmt.executeQuery("select recordId ,equipmentId, userId, dateEnd from borrowRecord where "
                + "returnDate is NULL;");
        while (myRs5.next()){
            if (myRs5.getInt("equipmentId")== Integer.parseInt(borrowId)
                && myRs5.getInt("userId") == Integer.parseInt(userNameValue)){
                activateKey = 1;
                userId = myRs5.getInt("userId");
                dateEnd = myRs5.getString("dateEnd");
                recordId = myRs5.getInt("recordId");
                break;
            }
        }
        if (activateKey ==1){
            PreparedStatement updateBorrowRecord = myCon.prepareStatement ("Update borrowRecord set returnDate = "
                    + "current_timestamp() where recordId = ?;");
            updateBorrowRecord.setInt(1, recordId);
            int executeUpdate = updateBorrowRecord.executeUpdate();
            ResultSet myRs6 = myStmt.executeQuery("select equipmentId from equipmentmaster where currentStatus = 1");
            while (myRs6.next()){
                PreparedStatement updateCurrentStatus = myCon.prepareStatement("UPDATE equipmentmaster SET currentStatus "
                        + "= 0 where equipmentID = ?;");
                updateCurrentStatus.setString(1, borrowId);
                int executeUpdate2 = updateCurrentStatus.executeUpdate();
                JOptionPane.showMessageDialog(null, "Succeed in returning");
                PreparedStatement updateNumberOfBorrowedItems = myCon.prepareStatement("UPDATE usermaster SET "
                        + "numberOfBorrowedItems = numberOfBorrowedItems -1 where userId = ?;");
                updateNumberOfBorrowedItems.setInt(1, userId);
                updateNumberOfBorrowedItems.executeUpdate();
                System.out.println(jCheckBox1.isSelected());
                if (!(jTextFieldFeedback.getText()).equals("") && jCheckBox1.isSelected()){
                    PreparedStatement updateQuestionMaster = myCon.prepareStatement("insert into questionmaster (userId, "
                            + "equipmentId, questionLocation, feedback, isDamage) values (?, ?, 0, ?, 1);");
                    updateQuestionMaster.setString(1, jTextFieldUserId.getText());
                    updateQuestionMaster.setString(2, jTextFieldBorrowId.getText());
                    updateQuestionMaster.setString(3, jTextFieldFeedback.getText());
                    updateQuestionMaster.executeUpdate();
                    System.out.println("inserted feedback");
                    PreparedStatement changeEquipmentStatusToDamage = myCon.prepareStatement("Update equipmentmaster set "
                            + "equipmentStatus = 0 where equipmentId = ?;");
                    changeEquipmentStatusToDamage.setString(1, jTextFieldBorrowId.getText());
                    changeEquipmentStatusToDamage.executeUpdate();
                }
                else if (!(jTextFieldFeedback.getText()).equals("")){
                    PreparedStatement updateQuestionMaster = myCon.prepareStatement("insert into questionmaster (userId, "
                            + "equipmentId, questionLocation, feedback) values (?, ?, 0, ?);");
                    updateQuestionMaster.setString(1, jTextFieldUserId.getText());
                    updateQuestionMaster.setString(2, jTextFieldBorrowId.getText());
                    updateQuestionMaster.setString(3, jTextFieldFeedback.getText());
                    updateQuestionMaster.executeUpdate();
                    System.out.println("inserted feedback");
                }
                else if (jCheckBox1.isSelected()){
                    PreparedStatement updateQuestionMaster = myCon.prepareStatement("insert into questionmaster (userId, "
                            + "equipmentId, questionLocation, isDamaged) values (?, ?, 0, 1);");
                    updateQuestionMaster.setString(1, jTextFieldUserId.getText());
                    updateQuestionMaster.setString(2, jTextFieldBorrowId.getText());
                    updateQuestionMaster.executeUpdate();
```

Sends the feedback if the user write some feedback or tick the box

```
else if (!(jTextFieldFeedback.getText()).equals("")){
    PreparedStatement updateQuestionMaster = myCon.prepareStatement("insert into questionmaster (userId, "
            + "equipmentId, questionLocation, feedback) values (?, ?, 0, ?);");
    updateQuestionMaster.setString(1, jTextFieldUserId.getText());
    updateQuestionMaster.setString(2, jTextFieldBorrowId.getText());
    updateQuestionMaster.setString(3, jTextFieldFeedback.getText());
    updateQuestionMaster.executeUpdate();
    System.out.println("inserted feedback");
}
else if (jCheckBox1.isSelected()){
    PreparedStatement updateQuestionMaster = myCon.prepareStatement("insert into questionmaster (userId, "
            + "equipmentId, questionLocation, isDamaged) values (?, ?, 0, 1);");
    updateQuestionMaster.setString(1, jTextFieldUserId.getText());
    updateQuestionMaster.setString(2, jTextFieldBorrowId.getText());
    updateQuestionMaster.executeUpdate();
    System.out.println("inserted feedback");
    PreparedStatement changeEquipmentStatusToDamage = myCon.prepareStatement("Update equipmentmaster set "
            + "equipmentStatus = 0 where equipmentId = ?;");
    changeEquipmentStatusToDamage.setString(1, jTextFieldBorrowId.getText());
    changeEquipmentStatusToDamage.executeUpdate();
}
break;
}
```

Use the date timer to check whether it is late. If it is late save it to the borrowrecord the equipment is returned late, the usermaster lateRecord+1
Use optionplane print a notification

Tell the user how much late records (if have). Tell he have and what is the consequence or what should be done.

```
// Check if the item is returned late
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
Date date1 = new Date(System.currentTimeMillis());
System.out.println(formatter.format(date1));
if(formatter.format(date1).compareTo(dateEnd) > 0){
    System.out.println("late");
    PreparedStatement updateLateOrNot = myCon.prepareStatement("UPDATE borrowRecord SET lateOrNot = 1 where recordId = ?;");
    updateLateOrNot.setInt(1, recordId);
    updateLateOrNot.executeUpdate();
    //find the number of dates late     cite from https://stackoverflow.com/questions/20165564/calculating-days-between-two-dates-with-java
    java.util.Date date2 = formatter.parse(dateEnd);
    long difference = Math.abs(date1.getTime() - date2.getTime());
    long differenceDates = difference / (24 * 60 * 60 * 1000);
    String dayDifference = Long.toString(differenceDates);
    JOptionPane.showMessageDialog(null, "You returned " + dayDifference + " days late");
    // mark bad record?
    PreparedStatement updateLateRecord =myCon.prepareStatement("Update userMaster set lateRecord = lateRecord + 1 where userId = ?;");
    updateLateRecord.setInt(1, Integer.parseInt(userNameValue));
    updateLateRecord.executeUpdate();
    PreparedStatement getLateRecord = myCon.prepareStatement("Select lateRecord from userMaster where userId = ?;");
    getLateRecord.setInt(1, Integer.parseInt(userNameValue));
    ResultSet myRs7 = getLateRecord.executeQuery();
    int noOflateRecord =0;
    while (myRs7.next()){
        noOflateRecord = myRs7.getInt("lateRecord");
        break;
    }
    if (noOflateRecord%2 == 0){
        PreparedStatement bye =myCon.prepareStatement("Update userMaster set borrowAuthorization = 0 where userId = ?;");
        bye.setInt(1, Integer.parseInt(userNameValue));
        bye.executeUpdate();
        JOptionPane.showMessageDialog(null, "You have a total of " + noOflateRecord +" late records. You won't have "
                + "authorization in borrowing any more items. Please find responsible computer science teacher in Room 705 to get "
                + "the authoriztion of borrowing");
    }
    else{
        JOptionPane.showMessageDialog(null, "You have a total of " + noOflateRecord + " late records. "
                + "If you return late one more time, you would only be be allowed to borrow "
                + "after finding responsible computer sceince teacher to get the autorization of borrowing" );
    }
}

else{
    JOptionPane.showMessageDialog(null, "ReturnId cannot match with UserID");

}
```

```
catch(Exception exc){
    exc.printStackTrace();
}
}
```

Similar techniques are used here. Returning function updates 4 tables of MySQL, “usermaster”, “equipmentmaster”, “questionmaster” and “borrowRecord”. Below listed which component respective component in each table that is updated in this return page.

- Usermaster: Update the numberOfItemsBorrowed. Update LateRecord and borrowAuthorization if certain conditions are satisfied.
- Equipmentmaster: Update the currentStatus to 0 (free to borrow). If the checkbox “Report Damage” is selected, it also update the equipmentStatus to 0 (damaged).
- Questionmaster: Insert a new row if there are feedbacks or the question box is ticked.
- BorrowRecord: Update returnDate. And if it is returned late, Update lateOrNot to 1 (late).

```
// Check if the item is returned late
SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd");
Date date1 = new Date(System.currentTimeMillis());
```

It also checks whether it is late or not. This is done by using System.currentTimeMills().

```
java.util.Date date2 = formatter.parse(dateEnd);
long difference = Math.abs(date1.getTime() - date2.getTime());
long differenceDates = difference / (24 * 60 * 60 * 1000);
String dayDifference = Long.toString(differenceDates);
```

Finding the number of dates late is done by using a calculation method referenced online to calculate.

Admin Side

Check / change / delete equipment page

Major Functions

- Table of equipment
- Filter function
- Delete equipment
- Change status of equipment

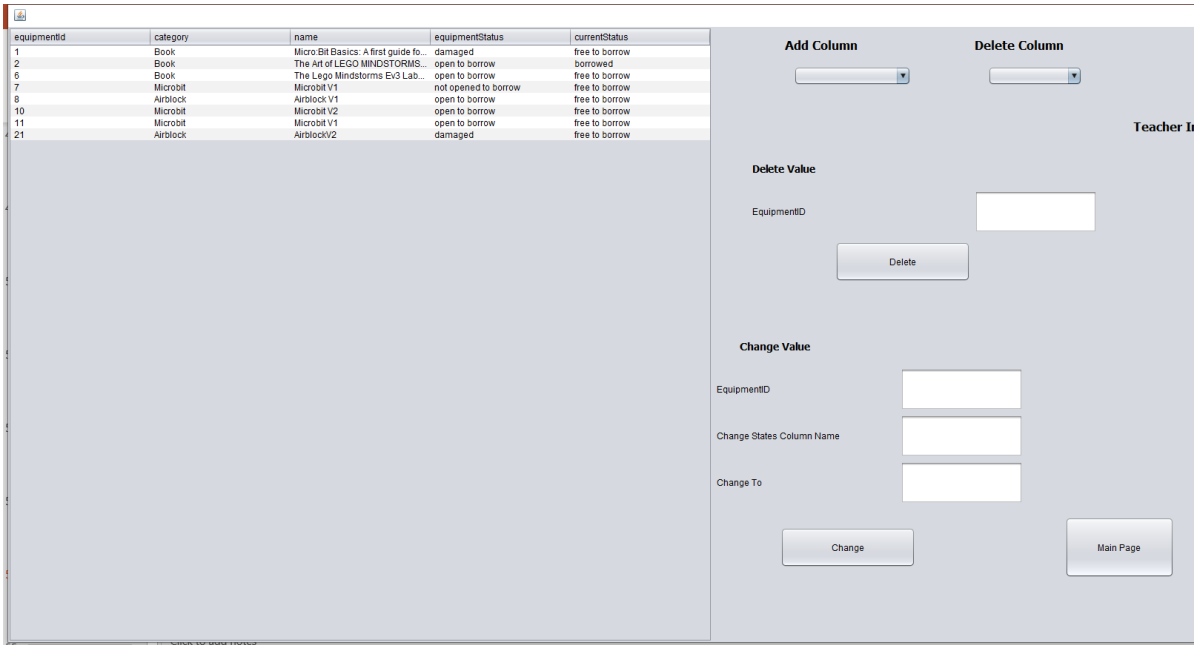


Table of equipment

Two arrays are imported from the commonClass, they are equipmentColumn[] and equipmentHiddenColumn[].
All combination of values in equipmentColumn[] and equipmentHiddenColumn[] have stores all the columns in “equipmentmaster”. equipmentColumn[] stores all column that the admin wants to see. equipmentColumn[] stores all the column names that the admin wants to check at the first moment. equipmentHiddenColumn[] stores the rest of the column names that are not presented at first.
As suggested in the JTable part, it is described how different steps are building a table.

Filter Function

The filter function is done by pressing the combo box at the top right-hand side either “add column” or “delete column. It then will refresh and update the table.
As mentioned, equipmentColumn[] stores those column that the admin wants to see while equipmenHiddenColum[] saves the rest. Thus following steps are done.

- Showing initial function in the Combo box

```
public void comboBox1and2(String[] equipmentHiddenColumn, String[] equipmentColumn){
    int hiddencolumnlength  = equipmentHiddenColumn.length;
    int columnlength  = equipmentColumn.length;
    jComboBox1.insertItemAt("", 0);
    for (int count = 0; count < hiddencolumnlength; count++){
        jComboBox1.addItem(equipmentHiddenColumn[count]);
    }
    jComboBox2.insertItemAt("", 0);
    for (int count = 1; count < columnlength; count++){ //the reason of being 1 is to let it ignore the equipmentId
        jComboBox2.addItem(equipmentColumn[count]);
    }
}
```

1. Inserting equipmenthiddenColumn into jCombox1 (the combo box that is used to “add column”). This is planned like that as when the “add column” combo box is pressed, there will be a column that is not showing presently in the table added.

2. Inserting equipmentColumn into jCombox2 (the combo box that is used to “delete column”). This is planned like that as when the “delete column” combo box is pressed, there will be a column which is showing presently in the table being removed.

insertItemAt(“ ”,0) is used to certain that there isn’t any initial component being selected.

Get the selected value from combo box, do action if the selected value has content

Extend the array length, then input the selectedValue into the array

Take the selectedValue away and shift some other contents in the hiddenColumn to the pevious index.

decrease the array length by one

Re-render the whole page

```
private void jComboBox1ActionPerformed(java.awt.event.ActionEvent evt) {
    String selectedValue = jComboBox1.getSelectedItem().toString();
    if (selectedValue != ""){
    equipmentColumn = java.util.Arrays.copyOf(equipmentColumn, equipmentColumn.length+1);
    equipmentColumn[equipmentColumn.length-1] = selectedValue;

    for (int count=0; count < equipmentHiddenColumn.length; count++){
        if (equipmentHiddenColumn[count] == selectedValue){
            for(int x = count; x < equipmentHiddenColumn.length-1 ; x++){
                equipmentHiddenColumn[x]=equipmentHiddenColumn[x+1];
            }
            break;
        }
    }
    equipmentHiddenColumn = java.util.Arrays.copyOf(equipmentHiddenColumn, equipmentHiddenColumn.length-1);
    new teacherInventory().setVisible(true);
    dispose();
    }
}
```

- If ComboBox is pressed

1. Each time a button is pressed. It needs to reduce the length of one column and increase another. Allowing the selected value transferred from “adding column” combo box transfer to “delete column” combo box. Lastly, re-render the page, a new table would generate.

```java
private void jComboBox2ActionPerformed(java.awt.event.ActionEvent evt) {
    String selectedValue = jComboBox2.getSelectedItem().toString();
    if (selectedValue != ""){
    equipmentHiddenColumn = java.util.Arrays.copyOf(equipmentHiddenColumn, equipmentHiddenColumn.length+1);
    equipmentHiddenColumn[equipmentHiddenColumn.length-1] = selectedValue;

    for (int count=0; count < equipmentColumn.length; count++){
        if (equipmentColumn[count] == selectedValue){
            for(int x = count; x < equipmentColumn.length-1 ; x++){
                equipmentColumn[x]=equipmentColumn[x+1];
            }
            break;
        }
    }
    equipmentColumn = java.util.Arrays.copyOf(equipmentColumn, equipmentColumn.length-1);
    new teacherInventory().setVisible(true);
    dispose();
    }
}
```

Same structure is done here.

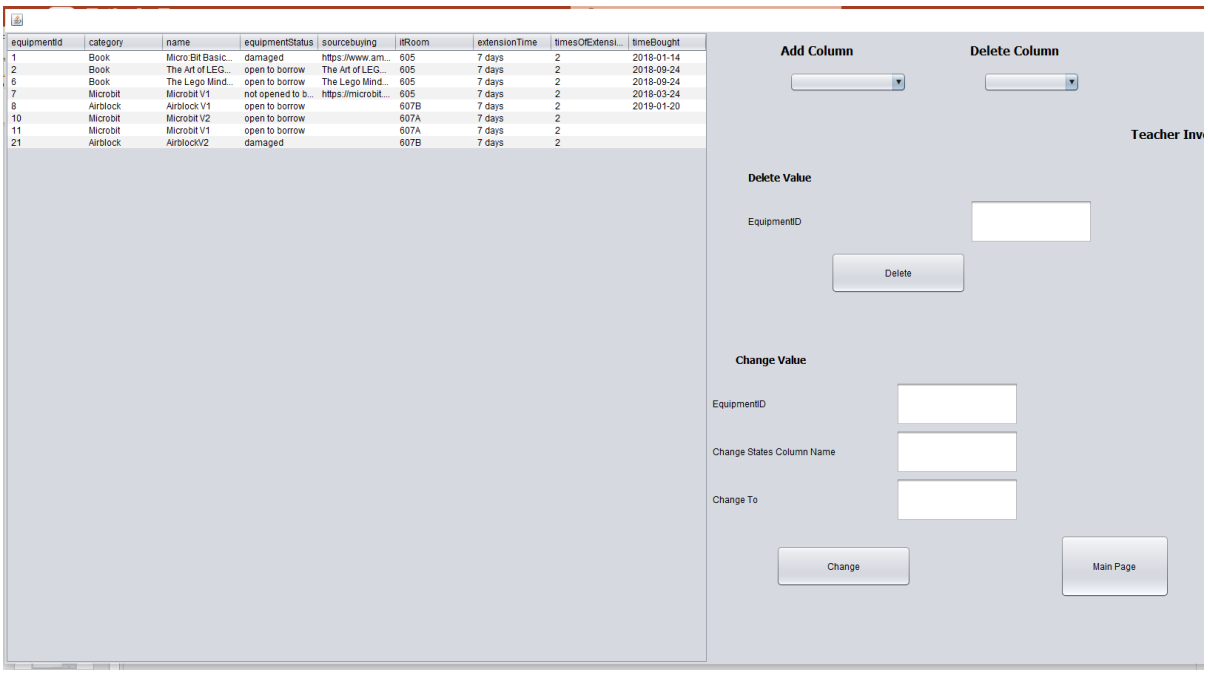


Figure: Demo after adding and deleting a few columns, could be compared with the figure above.
Added column: “sourcebuying”, “itRoom”, “extensionTime”, “timeOfExtension”, “timeBought”
Deleted column: “currentStatus”

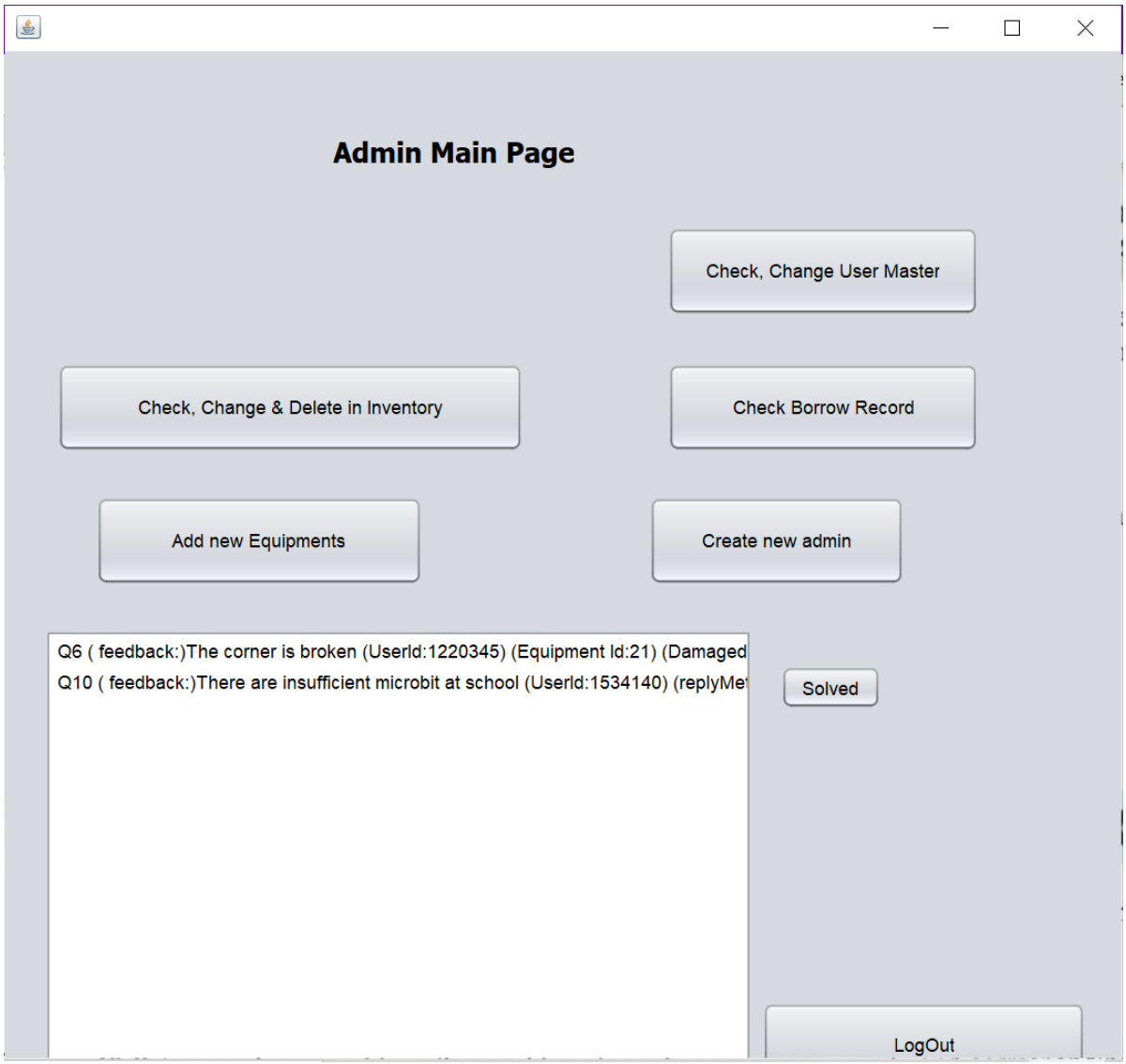


Figure: List demo

Main Page: Resolve Button

Step 1: Get the selectedValue

Step 2: Cut the selectedValue into 1/6

Step 3: Convert that part all into numbers

Step 4: Connect and update

Step 5: Refresh

```java
private void jButtonReadActionPerformed(java.awt.event.ActionEvent evt) {
    //https://www.w3resource.com/java-exercises/basic/java-basic-exercise-69.php
    String selectedValue = jList1.getSelectedValue(); // read the JList
    System.out.println("SelectedValue: " + selectedValue);
    String newSelectedValue =  selectedValue.substring(0, selectedValue.length()/6);
    System.out.println("cutted into 1/6" + newSelectedValue);
    //https://stackoverflow.com/questions/4030928/extract-digits-from-a-string-in-java
    final StringBuilder sb = new StringBuilder(
        newSelectedValue.length() /* also inspired by seh's comment */);
for(int i = 0; i < newSelectedValue.length(); i++){
    final char c = newSelectedValue.charAt(i);
    if(c > 47 && c < 58){
        sb.append(c);
    }
}
    String number = sb.toString();
    System.out.println("The number part of it" + number);
    try{
        Connection myCon = DriverManager.getConnection("jdbc:mysql://localhost:3306/borrowrecord","myuser","1222");
        Statement myStmt = myCon.createStatement();
        PreparedStatement updateisRead = myCon.prepareStatement("Update questionmaster set isRead = 1 where questionNumber = ?;");
        updateisRead.setString(1, number);
        updateisRead.executeUpdate();
        new teacherFirstPage().setVisible(true);
        dispose();
    }
    catch(Exception exc){
```

The resolve function is created due to the reason that when the teacher finishes resolving or reading the question, he may not want to see this question showing up in his question box. After pressing the button, the question would be resolved and no longer will shown on the admin interface. Though there is a difficulty, as all the elements in the list are all cropped in a String thus it is hard to extract some unique key from it such

as the “questionId” to distinguish which question does the admin wants to remove. Hence the following procedure is used.

1. When the button is pressed, it selects the “JList1.SelectedValue()”.
2. It cuts the length of the selected value into 1/6 and save it to newSelectedValue.
3. Using the method referenced on the internet, it helps me to select only the number parts of this newSelectedValue.
4. This will help me to extract the questionId only.

This method is mostly available unless the student typed a super long description which would interrupt the result.

Total: 1278 word.