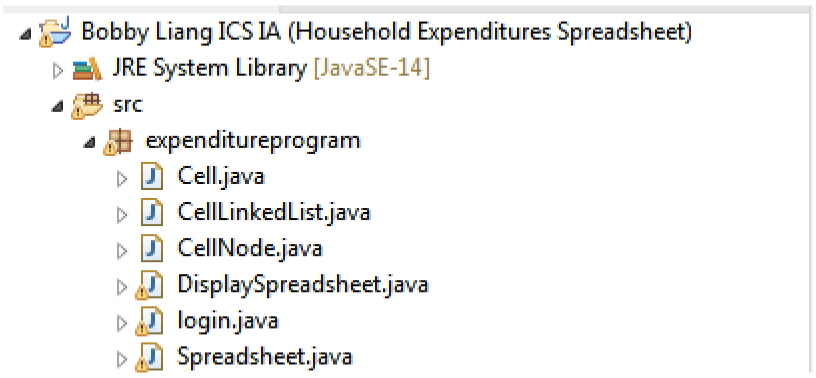
Bobby Liang

IA Implementation

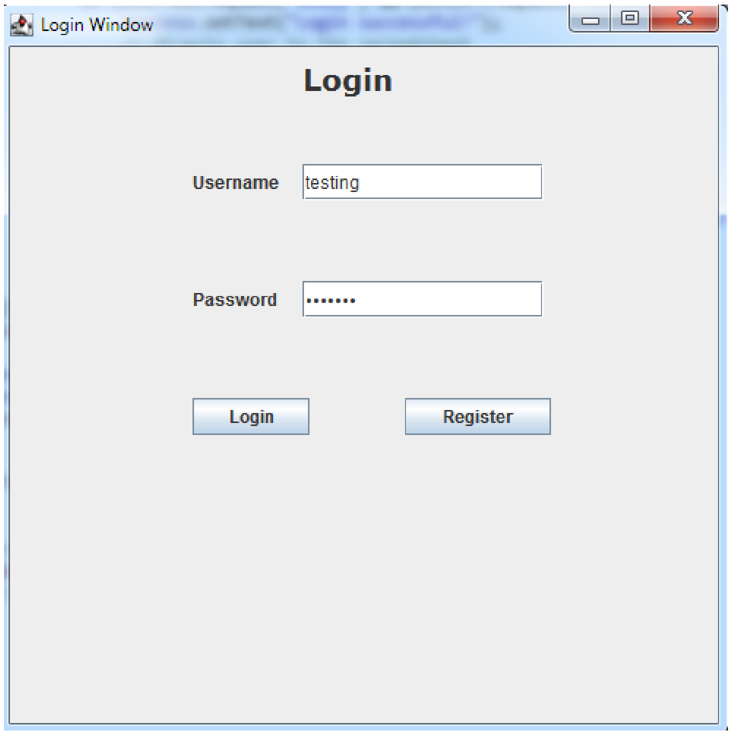
The product is a Java program, written in the Eclipse IDE. The program's functionality is to show an organized spreadsheet that can be modified and printed by the client. All inputted data should be saved in a text file, along with the login credentials.

Files (as of January 8)



```
Bobby Liang ICS IA (Household Expenditures Spreadsheet)
  JRE System Library [JavaSE-14]
  src
    expenditureprogram
      Cell.java
      CellLinkedList.java
      CellNode.java
      DisplaySpreadsheet.java
      login.java
      Spreadsheet.java
```

Login screen:

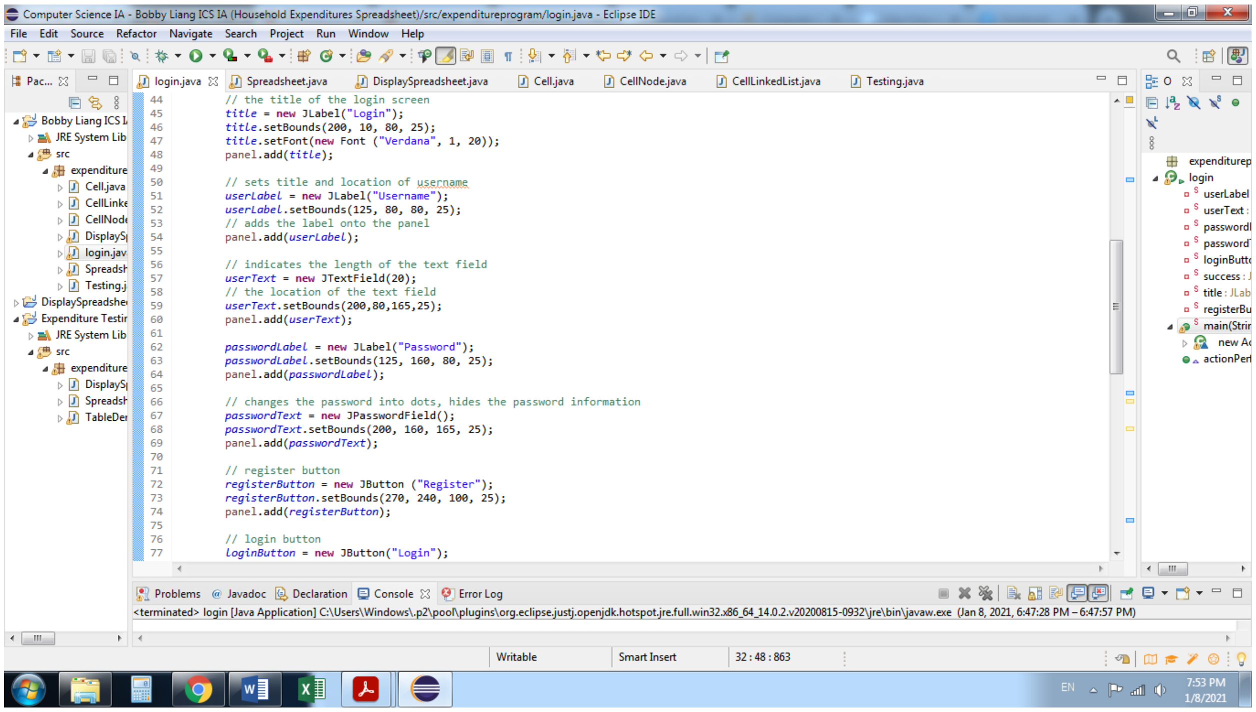


Username: testing

Password: •••••••

For the program's interface, I imported and used Java Labels, TextFields, Buttons, and PasswordField.

Bobby Liang



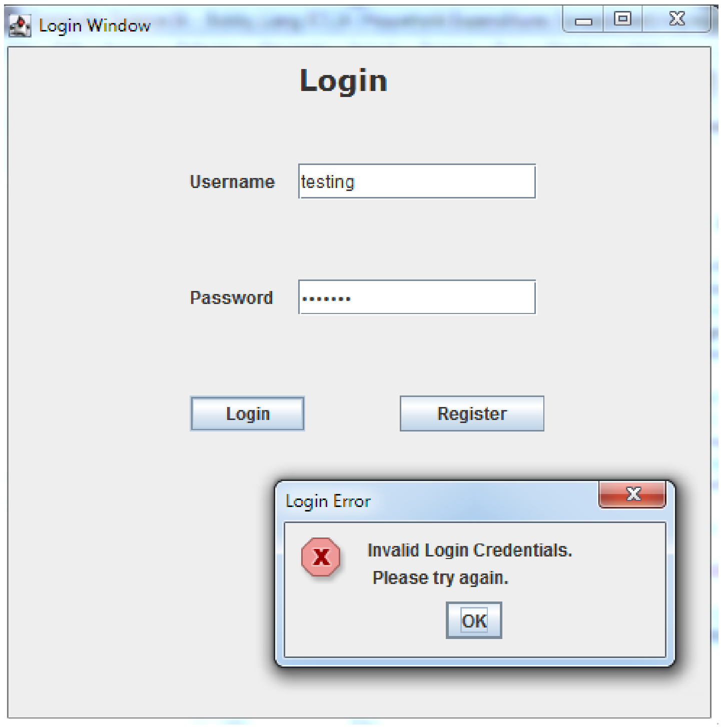
Computer Science IA - Bobby Liang ICS IA (Household Expenditures Spreadsheet)/src/expenditureprogram/login.java - Eclipse IDE

```
        // the title of the login screen
        title = new JLabel("Login");
        title.setBounds(200, 10, 80, 25);
        title.setFont(new Font ("Verdana", 1, 20));
        panel.add(title);

        // sets title and location of username
        userLabel = new JLabel("Username");
        userLabel.setBounds(125, 80, 80, 25);
        // adds the label onto the panel
        panel.add(userLabel);

        // indicates the length of the text field
        userText = new JTextField(20);
        // the location of the text field
        userText.setBounds(200,80,165,25);
        panel.add(userText);

        passwordLabel = new JLabel("Password");
        passwordLabel.setBounds(125, 160, 80, 25);
        panel.add(passwordLabel);

        // changes the password into dots, hides the password information
        passwordText = new JPasswordField();
        passwordText.setBounds(200, 160, 165, 25);
        panel.add(passwordText);

        // register button
        registerButton = new JButton ("Register");
        registerButton.setBounds(270, 240, 100, 25);
        panel.add(registerButton);

        // login button
        loginButton = new JButton("Login");
```

```
<terminated> login [Java Application] C:\Users\Windows\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_14.0.2.v20200815-0932\jre\bin\javaw.exe (Jan 8, 2021, 6:47:28 PM – 6:47:57 PM)
```

The password will show up as dots to prevent others from viewing sensitive information.

If the password and the login credentials do not match up, an error message will appear and access to the spreadsheet is denied.



Login Window

Login

Username: testing

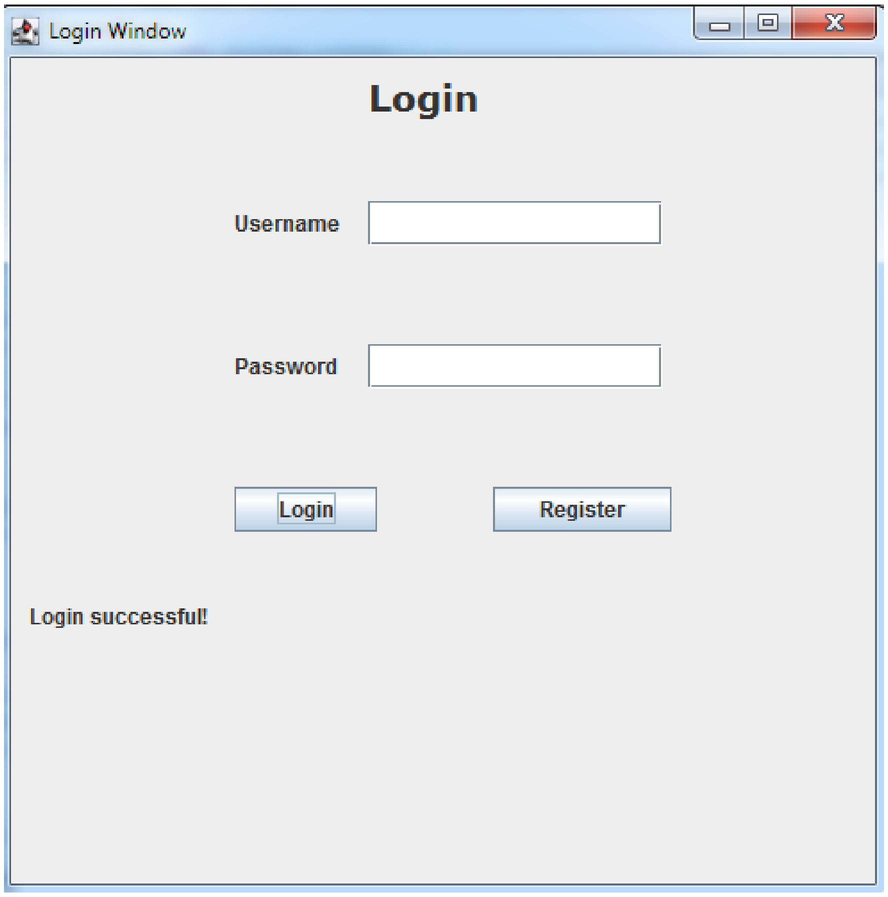
Password: •••••••

Login Error

Invalid Login Credentials.
Please try again.

Bobby Liang

Once the correct username and password combination are entered, the login screen will display a message that the login was successful, and the user has access to the spreadsheet that is opened in a new JFrame.

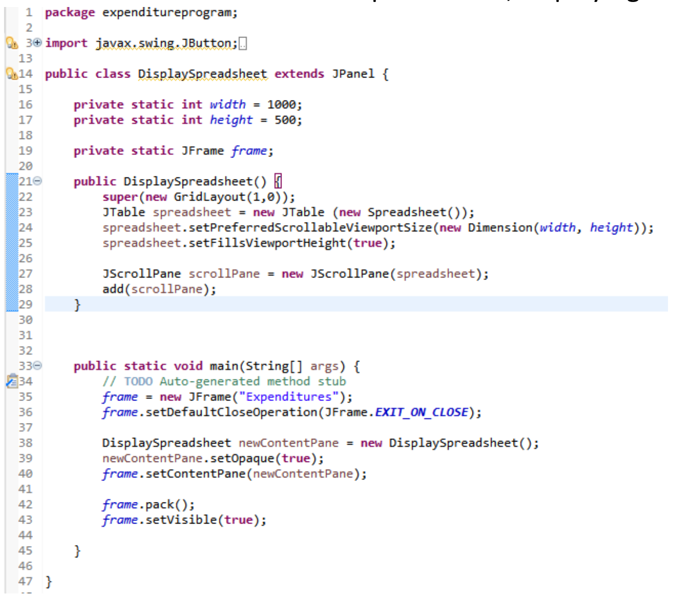


Login successful!

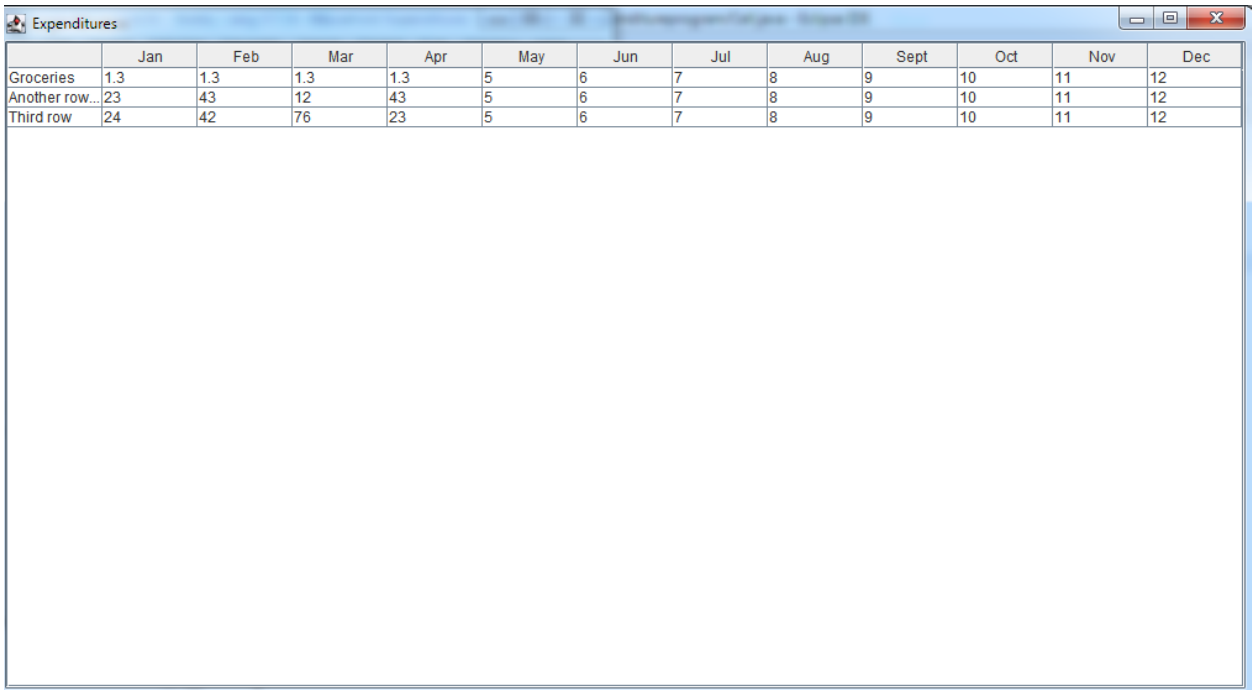
Spreadsheet: (incomplete class)

Bobby Liang

A JTable was used to create a spreadsheet, displaying the monthly expenses of the user.



```java
package expenditureprogram;

import javax.swing.JButton;

public class DisplaySpreadsheet extends JPanel {

    private static int width = 1000;
    private static int height = 500;

    private static JFrame frame;

    public DisplaySpreadsheet() {
        super(new GridLayout(1,0));
        JTable spreadsheet = new JTable (new Spreadsheet());
        spreadsheet.setPreferredScrollableViewportSize(new Dimension(width, height));
        spreadsheet.setFillsViewportHeight(true);

        JScrollPane scrollPane = new JScrollPane(spreadsheet);
        add(scrollPane);
    }


    public static void main(String[] args) {
        // TODO Auto-generated method stub
        frame = new JFrame("Expenditures");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        DisplaySpreadsheet newContentPane = new DisplaySpreadsheet();
        newContentPane.setOpaque(true);
        frame.setContentPane(newContentPane);

        frame.pack();
        frame.setVisible(true);

    }

}
```



	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sept	Oct	Nov	Dec
Groceries	1.3	1.3	1.3	1.3	5	6	7	8	9	10	11	12
Another row...	23	43	12	43	5	6	7	8	9	10	11	12
Third row	24	42	76	23	5	6	7	8	9	10	11	12

Ideally, the spreadsheet includes the months of the year and user specified rows regarding their monthly and yearly expenses.

The user should be able to add a row for their expenses.

The values of each row and each column would be summed up to provide a monthly total and yearly total of expenses.

There will be a button to return to the main menu (not yet implemented), and a button to edit any cell.

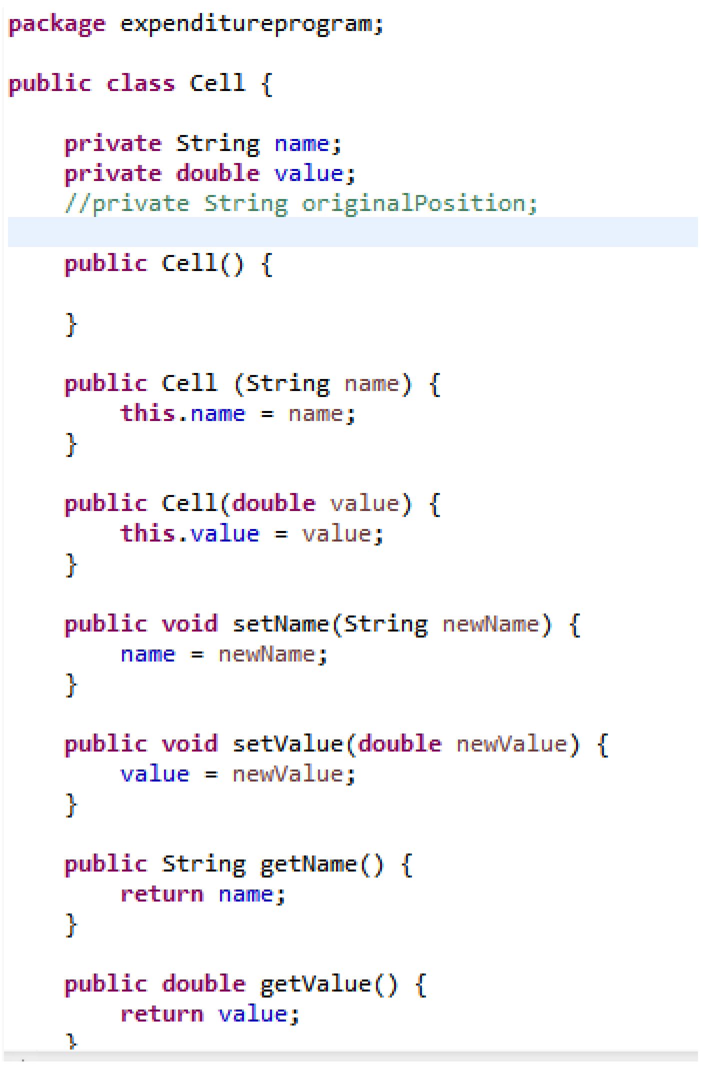
To modify and update the cells, a custom LinkedList will be created to store and modify these values.

Bobby Liang

Custom LinkedList:

The linked list consists of three classes; the cell, the cell node, and the cell linked list.

Cell:

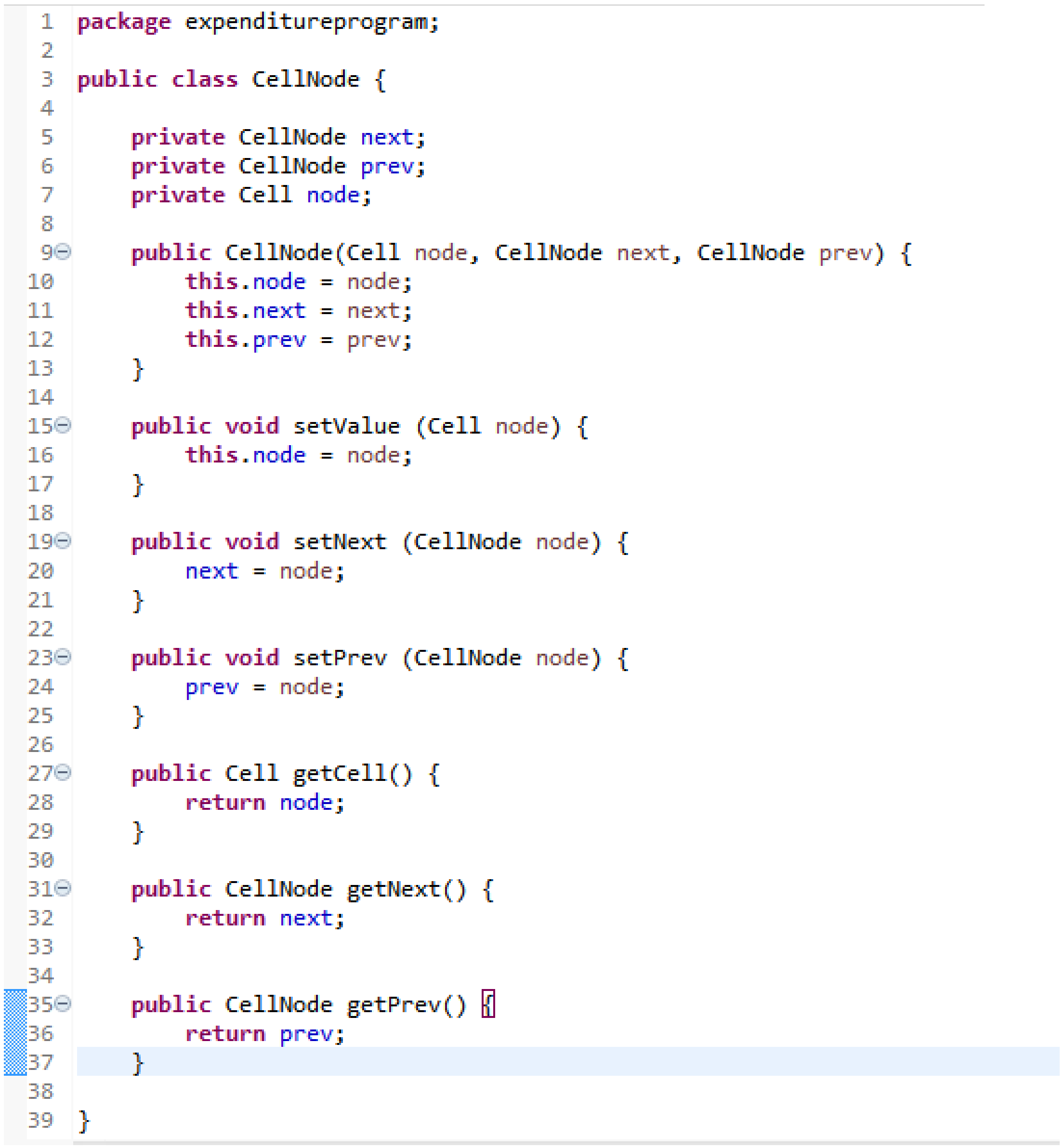


```java
package expenditureprogram;

public class Cell {

    private String name;
    private double value;
    //private String originalPosition;

    public Cell() {

    }

    public Cell (String name) {
        this.name = name;
    }

    public Cell(double value) {
        this.value = value;
    }

    public void setName(String newName) {
        name = newName;
    }

    public void setValue(double newValue) {
        value = newValue;
    }

    public String getName() {
        return name;
    }

    public double getValue() {
        return value;
    }
```

The cell class contains the content of one cell in the spreadsheet. The constructor can take in a String, given that it is the first element in the row, for the name of the expense. A cell can also be a double value for the user to input the amount of money spent during a month.

Bobby Liang

Cell Node:

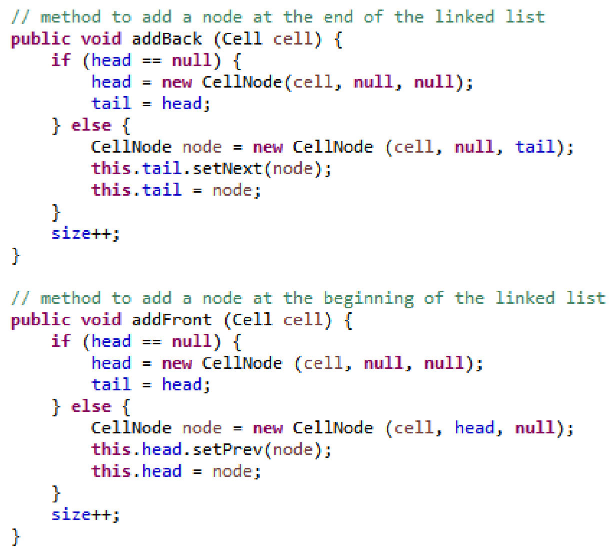


```java
package expenditureprogram;

public class CellNode {

    private CellNode next;
    private CellNode prev;
    private Cell node;

    public CellNode(Cell node, CellNode next, CellNode prev) {
        this.node = node;
        this.next = next;
        this.prev = prev;
    }

    public void setValue (Cell node) {
        this.node = node;
    }

    public void setNext (CellNode node) {
        next = node;
    }

    public void setPrev (CellNode node) {
        prev = node;
    }

    public Cell getCell() {
        return node;
    }

    public CellNode getNext() {
        return next;
    }

    public CellNode getPrev() {
        return prev;
    }

}
```

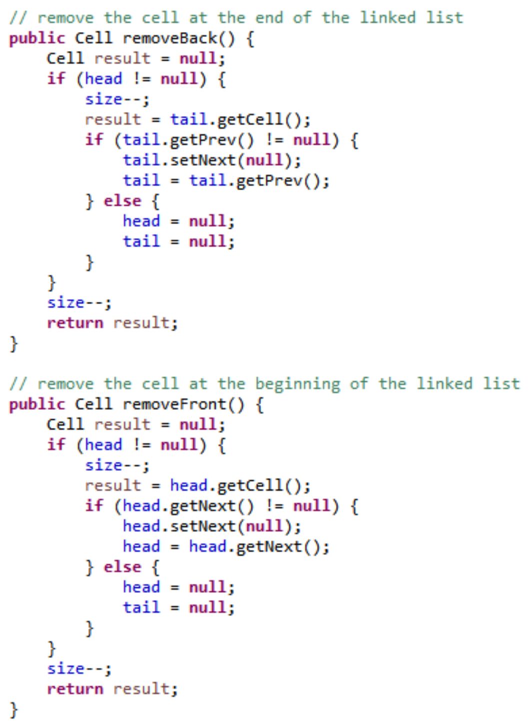
The CellNode class is a node of a singly linked list of Cells. The constructor creates a node with a Cell, the previous CellNode and the next CellNode. Accessor and modifier methods are used to modify and return any Cells and CellNodes.

CellLinkedList:

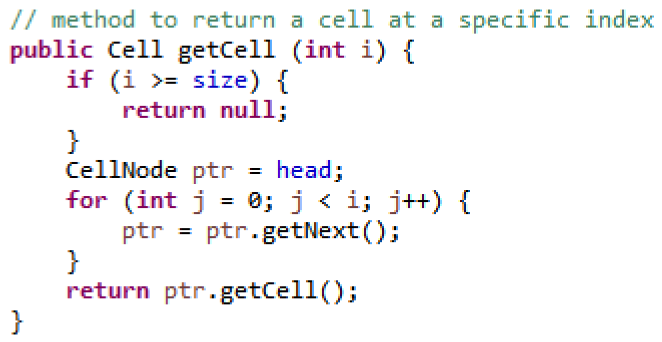
Bobby Liang



```java
// method to add a node at the end of the linked list
public void addBack (Cell cell) {
    if (head == null) {
        head = new CellNode(cell, null, null);
        tail = head;
    } else {
        CellNode node = new CellNode (cell, null, tail);
        this.tail.setNext(node);
        this.tail = node;
    }
    size++;
}

// method to add a node at the beginning of the linked list
public void addFront (Cell cell) {
    if (head == null) {
        head = new CellNode (cell, null, null);
        tail = head;
    } else {
        CellNode node = new CellNode (cell, head, null);
        this.head.setPrev(node);
        this.head = node;
    }
    size++;
}
```



```java
// remove the cell at the end of the linked list
public Cell removeBack() {
    Cell result = null;
    if (head != null) {
        size--;
        result = tail.getCell();
        if (tail.getPrev() != null) {
            tail.setNext(null);
            tail = tail.getPrev();
        } else {
            head = null;
            tail = null;
        }
    }
    size--;
    return result;
}

// remove the cell at the beginning of the linked list
public Cell removeFront() {
    Cell result = null;
    if (head != null) {
        size--;
        result = head.getCell();
        if (head.getNext() != null) {
            head.setNext(null);
            head = head.getNext();
        } else {
            head = null;
            tail = null;
        }
    }
    size--;
    return result;
}
```



```java
// method to return a cell at a specific index
public Cell getCell (int i) {
    if (i >= size) {
        return null;
    }
    CellNode ptr = head;
    for (int j = 0; j < i; j++) {
        ptr = ptr.getNext();
    }
    return ptr.getCell();
}
```

The CellLinkedList class takes in a Cell and creates CellNodes to organize the user's input. The modifier methods of addFront and addBack allows the user to specify where they want to add their expenses in the program. The getCell method in this class takes in a specific index (e.g. the user wants to access a cell at a specific row and column) and returns the cell. This is useful when modifying a specific cell.

Additional classes and methods that have yet to be created:

- Main menu class, should store all spreadsheets and allow users to choose which one they want to edit/create
- Find the sum of all rows and columns
- Implementation of CellLinkedList into the DisplaySpreadsheet class
- Storing username and password from text fields into a text file
- Updating the spreadsheet
- Implement new panel when user enters register
- Sorting the row names alphabetically