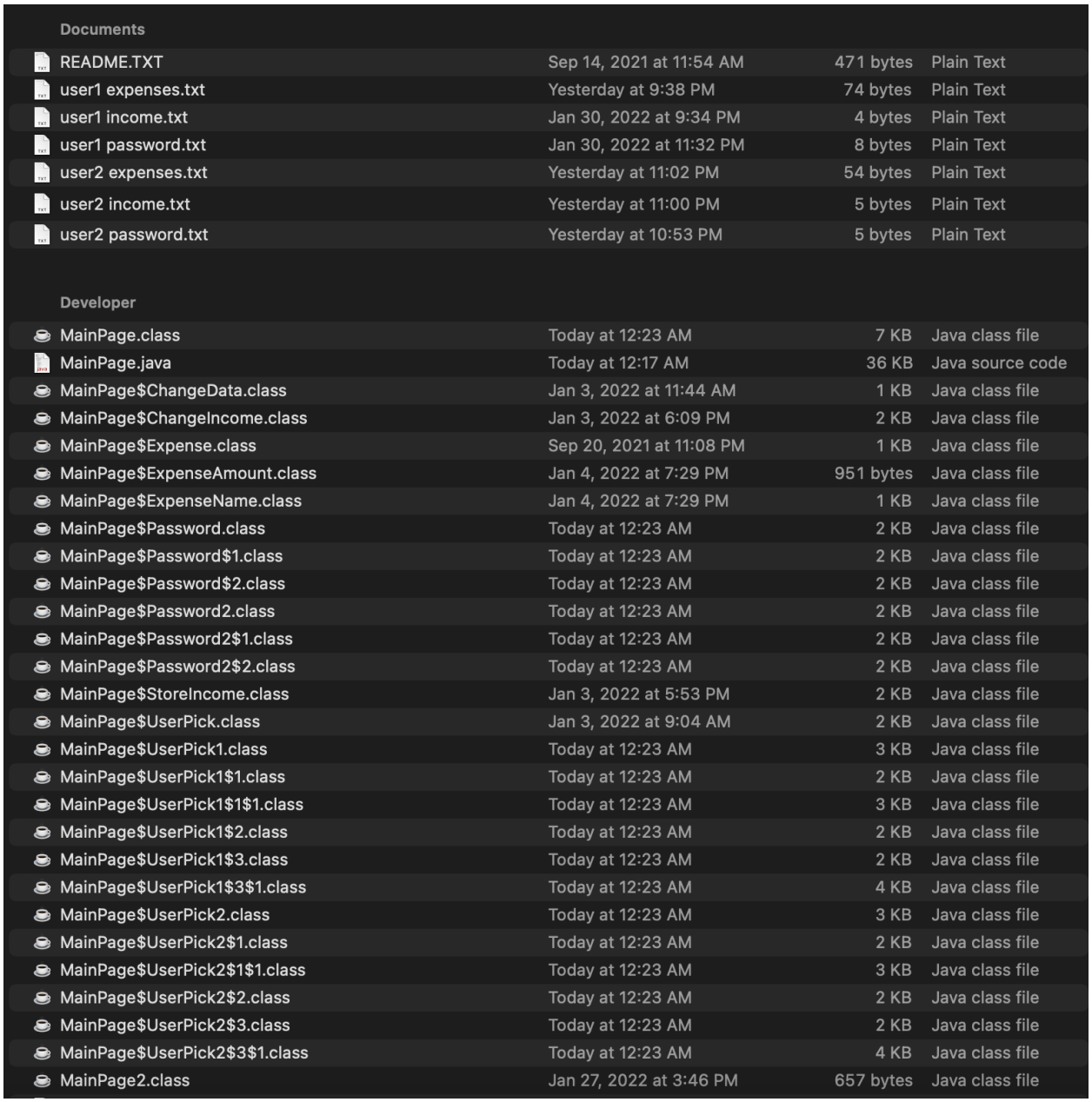
All Classes:



Documents

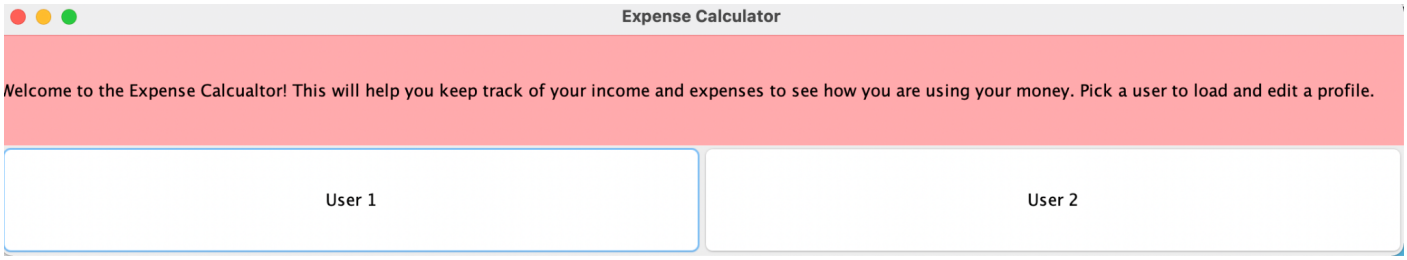
README.TXT	Sep 14, 2021 at 11:54 AM	471 bytes	Plain Text
user1 expenses.txt	Yesterday at 9:38 PM	74 bytes	Plain Text
user1 income.txt	Jan 30, 2022 at 9:34 PM	4 bytes	Plain Text
user1 password.txt	Jan 30, 2022 at 11:32 PM	8 bytes	Plain Text
user2 expenses.txt	Yesterday at 11:02 PM	54 bytes	Plain Text
user2 income.txt	Yesterday at 11:00 PM	5 bytes	Plain Text
user2 password.txt	Yesterday at 10:53 PM	5 bytes	Plain Text

Developer

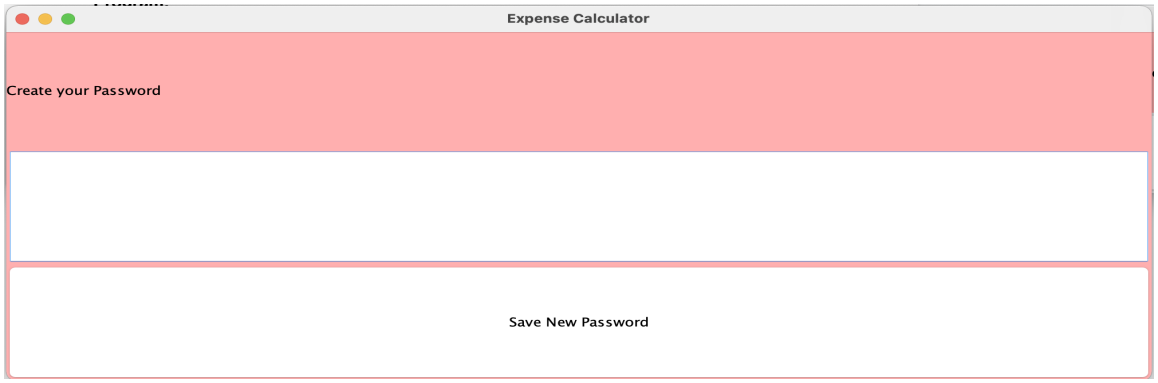
MainPage.class	Today at 12:23 AM	7 KB	Java class file
MainPage.java	Today at 12:17 AM	36 KB	Java source code
MainPage$ChangeData.class	Jan 3, 2022 at 11:44 AM	1 KB	Java class file
MainPage$ChangeIncome.class	Jan 3, 2022 at 6:09 PM	2 KB	Java class file
MainPage$Expense.class	Sep 20, 2021 at 11:08 PM	1 KB	Java class file
MainPage$ExpenseAmount.class	Jan 4, 2022 at 7:29 PM	951 bytes	Java class file
MainPage$ExpenseName.class	Jan 4, 2022 at 7:29 PM	1 KB	Java class file
MainPage$Password.class	Today at 12:23 AM	2 KB	Java class file
MainPage$Password$1.class	Today at 12:23 AM	2 KB	Java class file
MainPage$Password$2.class	Today at 12:23 AM	2 KB	Java class file
MainPage$Password2.class	Today at 12:23 AM	2 KB	Java class file
MainPage$Password2$1.class	Today at 12:23 AM	2 KB	Java class file
MainPage$Password2$2.class	Today at 12:23 AM	2 KB	Java class file
MainPage$StoreIncome.class	Jan 3, 2022 at 5:53 PM	2 KB	Java class file
MainPage$UserPick.class	Jan 3, 2022 at 9:04 AM	2 KB	Java class file
MainPage$UserPick1.class	Today at 12:23 AM	3 KB	Java class file
MainPage$UserPick1$1.class	Today at 12:23 AM	2 KB	Java class file
MainPage$UserPick1$1$1.class	Today at 12:23 AM	3 KB	Java class file
MainPage$UserPick1$2.class	Today at 12:23 AM	2 KB	Java class file
MainPage$UserPick1$3.class	Today at 12:23 AM	2 KB	Java class file
MainPage$UserPick1$3$1.class	Today at 12:23 AM	4 KB	Java class file
MainPage$UserPick2.class	Today at 12:23 AM	3 KB	Java class file
MainPage$UserPick2$1.class	Today at 12:23 AM	2 KB	Java class file
MainPage$UserPick2$1$1.class	Today at 12:23 AM	3 KB	Java class file
MainPage$UserPick2$2.class	Today at 12:23 AM	2 KB	Java class file
MainPage$UserPick2$3.class	Today at 12:23 AM	2 KB	Java class file
MainPage$UserPick2$3$1.class	Today at 12:23 AM	4 KB	Java class file
MainPage2.class	Jan 27, 2022 at 3:46 PM	657 bytes	Java class file

- The program is run from the Main Page class
- All classes extend JFrame.
- The text files record the different statistics of the user, like their income, expenses, and the name of those expenses. That way, the user doesn't have to input the same numbers every time they use the program.

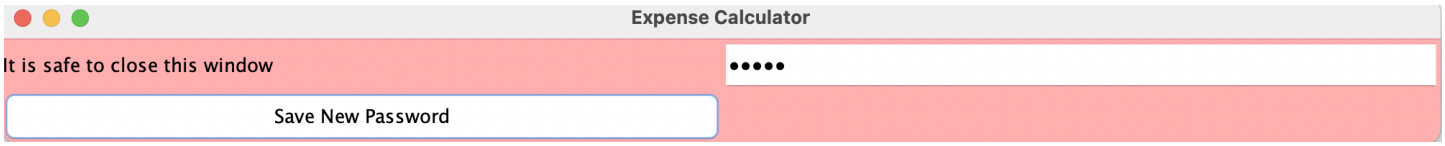
Program:



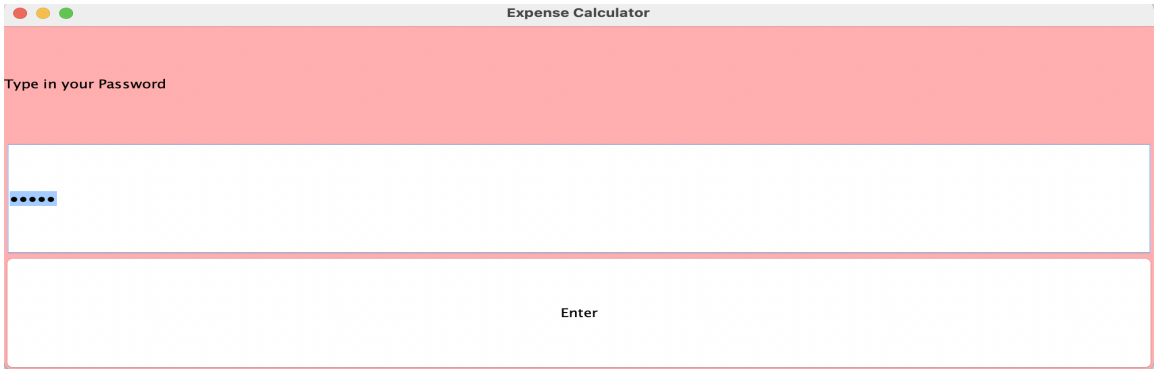
- This is the first page that greets the user upon running the program.
- The text above shows instructions for the user.
- The two buttons, User 1 and User 2, will load up the stats for whichever profile they pick in a new window.



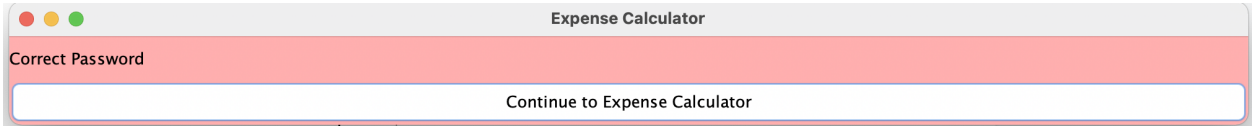
- After a user is selected, and if a password has not been made for said user yet, a window will show that asks for a password to be set.
- The user can enter a password and click ‘Save New Password’



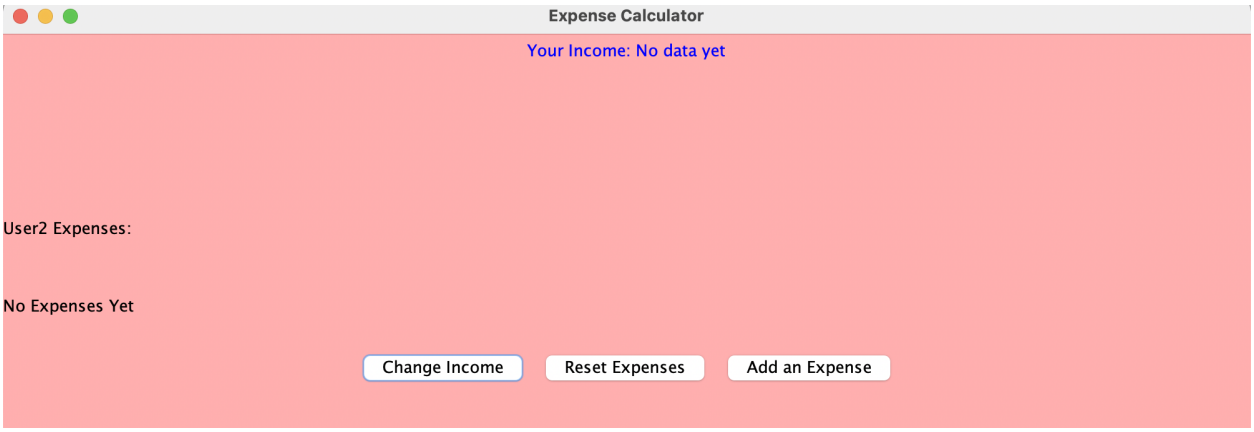
- The window is then updated, with “It is safe to close this window.” The client will then pick the user again, and this time they will be able to enter the password they just set to access the calculator.



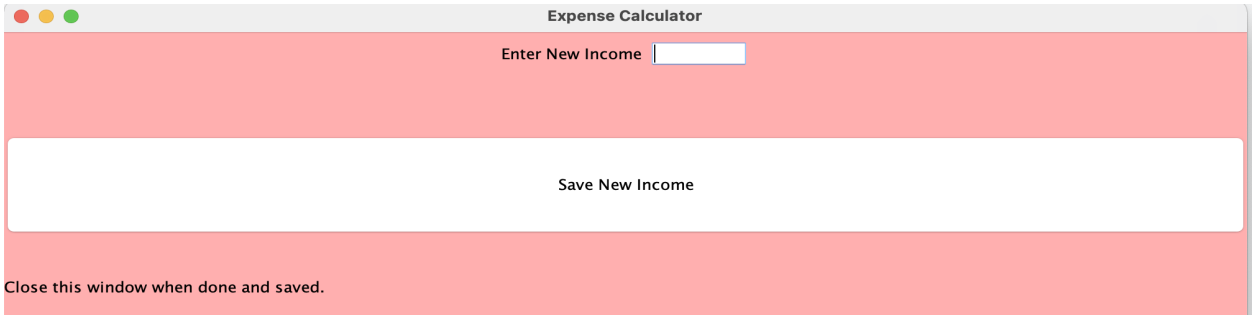
- If the user enters the same password again, then a new window with confirmation will appear.



- Clicking on “Continue to Expense Calculator” will open the Expense Calculator with the user’s information.

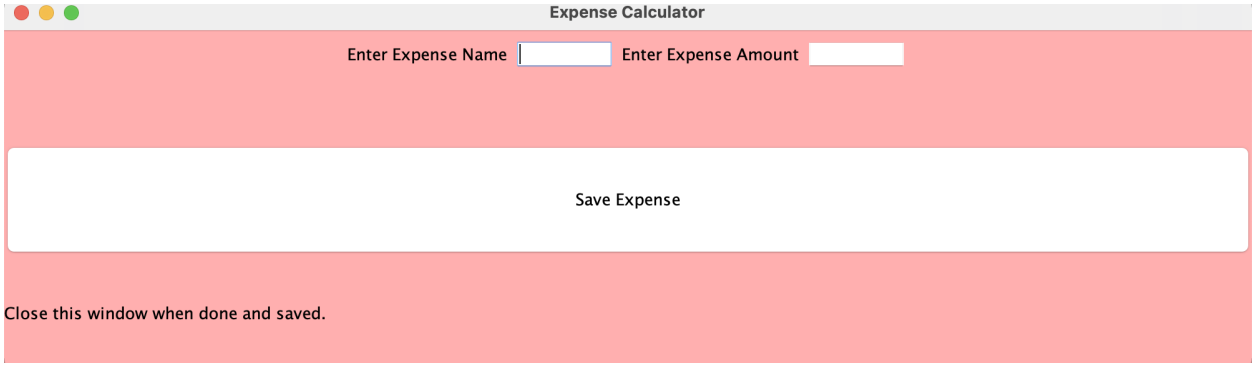


- This shows a user’s page with no data inputted yet, which means that the textfile for this user’s income and their expenses hasn’t been created yet.
- The “Change Income” button will create a new text file for this user’s income if it doesn't exist and enter a new income number. If the text file does exist, then it gets deleted and a new file is created for the new income number.
- The “Reset Expenses” button just deletes the existing expense text file.
- The “Add an Expense” button will create a new text file to store expenses if it doesn’t exist, and enter the expense inputted by the user. If the text file does exist then it will only add the expense inputted by the user with the existing data.

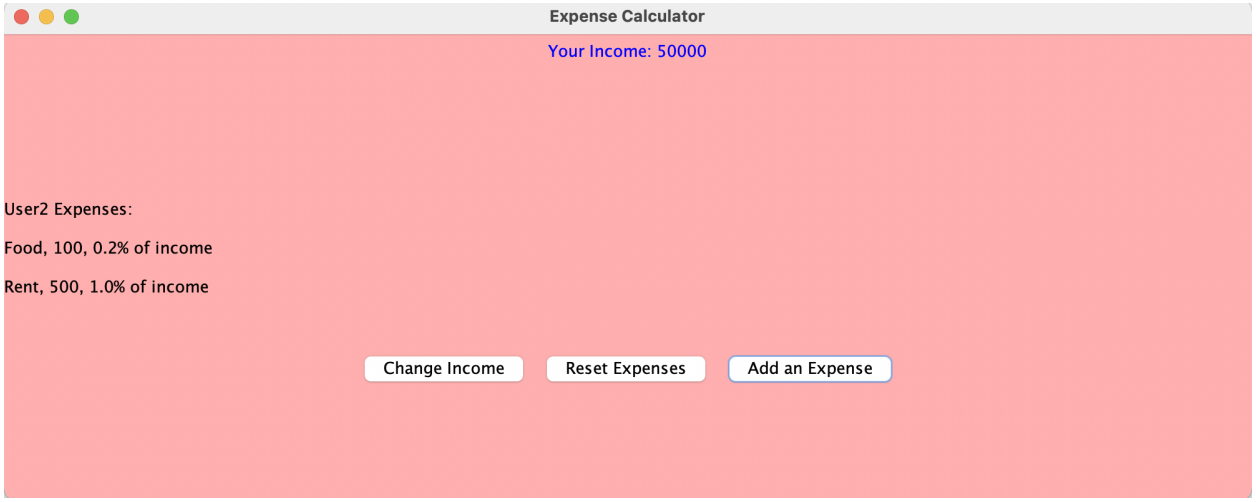


- The button “Save New Income” enters the number from the text field into the file “user1 income.txt”. The button also revalidates the previous window with income and expenses so that the stats are updated without needing to exit the program.

- The bottom text tells the user to close this window, which won't exit the program. The previous window with the data of the user will be already updated with the new income.

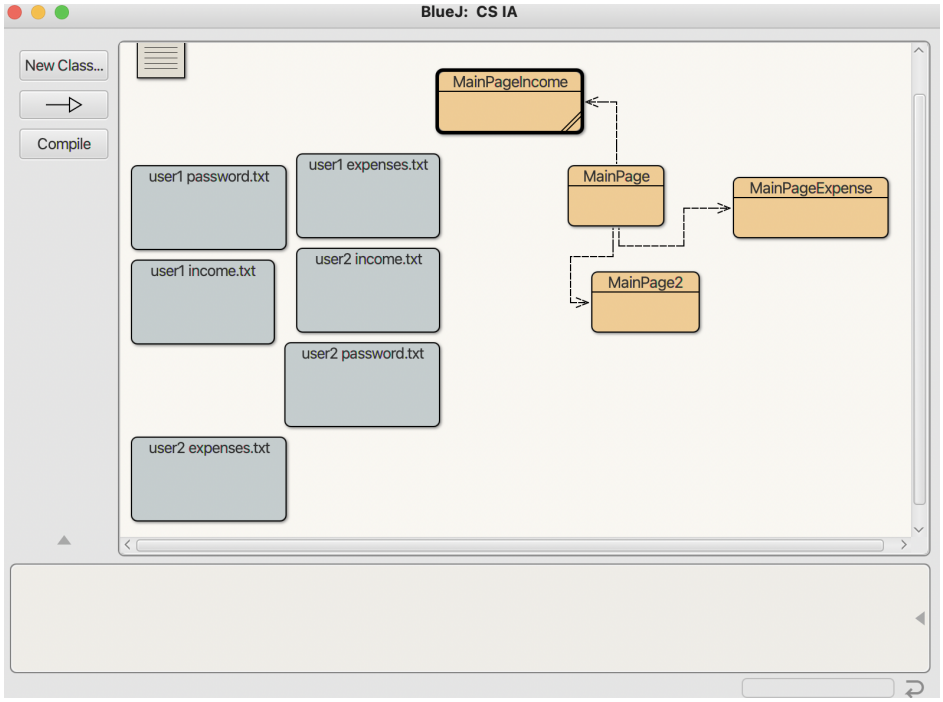


- This is the window that appears when "Add an Expense" is clicked.
- The button "Save Expense" will add the expense name paired with the amount into the user's expenses text file.
    - It then adds the new expense to the main page, without needing to stop running the program
- Again, closing the window doesn't exit the program.



- This user has entered $50,000 as their income and entered two expenses of "Food" and "Rent". It shows the name, dollar amount of the expense, and the percentage of the income that is spent on the expense. The expenses are shown in a list form in order for easy understanding and clarity.
    - According to Criterion A, this program lets users clearly see their expenses and income, as well as the percentage of income they are spending to better control how they spend their money.

Code:



- The class MainPage, which holds the program, also access the classes MainPage2, MainPageIncome, and MainPage Expense in order to create windows with different parameters.
- These classes only hold a constructor and nothing else.

```
import java.awt.*;
import javax.swing.*;
/**
 * Write a description of class MainPage2 here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class MainPage2 extends JFrame
{
    public MainPage2(){
        super("Expense Calculator");
        Container c = getContentPane();
        c.setBackground(Color.PINK);
        c.setLayout(new GridLayout(3,1));


    }
}
```

- This picture shows all the code of MainPage2, which is very simple. The other two auxiliary classes are also very similar, with minor tweaks in the parameters.

```
import java.awt.*;
import javax.swing.*;
import java.util.*;
import java.awt.event.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.io.*;
import java.io.IOException;
import java.io.File;
import java.util.Scanner;
import java.io.FileWriter;
import java.io.FileReader;
```

- These are all imported java packages for the program functions

```
private static JTextField textfield_income = new JTextField(6);
private static JTextField textfield_expenseamount = new JTextField(6);
private static JTextField textfield_expensename = new JTextField(6);
private static JPasswordField textfield_password = new JPasswordField(6);

private static JLabel Intro = new JLabel();
private static JLabel IncomeOutput = new JLabel();
private static JLabel EnterIncome = new JLabel();
private static JLabel EnterExpenseN = new JLabel();
private static JLabel EnterExpenseA = new JLabel();
private static JLabel Close = new JLabel();
private static JLabel text = new JLabel();
private static JLabel correct = new JLabel();
private static JLabel incorrect = new JLabel();
```

- These text fields and JLabels are used throughout the program, most of their uses are described in the title

```
public MainPage(){
    super("Expense Calculator");
    Container c = getContentPane();
    c.setBackground(Color.PINK);
    c.setLayout(new GridLayout(2,1));
}
```

- The method “MainPage” is the constructor that creates the window and sets parameters like the color and the layout.

```
public static void Calculator(){
    MainPage window = new MainPage();
    window.setBounds(300,300,1100,200);
    window.setDefaultCloseOperation(EXIT_ON_CLOSE);

    Intro.setText("Welcome to the Expense Calcualtor! This will help you keep track of your income and expenses to see how you are using your money. Pick a user to load and e
    window.add(Intro);

    JPanel users = new JPanel();
    users.setLayout(new GridLayout (1,2));

    JButton button_obj3 = new JButton ("User 1");
    button_obj3.addActionListener(new Password());
    users.add(button_obj3);

    JButton button_obj4 = new JButton ("User 2");
    button_obj4.addActionListener(new Password2());
    users.add(button_obj4);

    window.add(users);

    window.setVisible(true);
}
```

- This is the main method that runs the program. It creates the first window for the user, that asks the user to pick “User 1” or “User 1” and lets them know what to expect from the program.
- The buttons activate the classes Password and Password2, respectively. This is the security feature of this program, and is identical for both users.

```
static String something="";
static String user1Password = "user1 password.txt";
private static class Password implements ActionListener{
    public void actionPerformed(ActionEvent a){
        MainPageIncome window5 = new MainPageIncome();
        window5.setBounds(300,300,1000,300);
```

- The class Password is used for user1, and Password2 is for user2.
- The constructor for the window uses the class MainPageIncome.

```
try{
    File myObj = new File(user1Password);
    if(!myObj.exists()){
        myObj.createNewFile();
        System.out.println("File successfully created.");
        text.setText("Create your Password");
        window5.add(text);
        window5.add(textfield_password);
        JButton button_obj4 = new JButton("Save New Password");

        window5.add(button_obj4);
        button_obj4.addActionListener(new ActionListener(){
                @Override
                public void actionPerformed(ActionEvent j){
                    try{
                        FileWriter myWriter = new FileWriter(user1Password, true);
                        myWriter.append(textfield_password.getText());
                        myWriter.close();
                        text.setText("It is safe to close this window");
                    }catch(IOException e){
                        System.out.println("An error occurred.");
                    }
                }
            });
    }
```

- If a password text file has not been made yet, it will create one and ask the user to set their password to be stored in that text file.

```
else{
    Scanner myReader = new Scanner(myObj);

    while(myReader.hasNextLine()){
        something = myReader.nextLine();
    }
    myReader.close();
    text.setText("Type in your Password");
    window5.add(text);
    window5.add(textfield_password);

    JButton button_obj4 = new JButton("Enter");
    window5.add(button_obj4);
    button_obj4.addActionListener(new ActionListener(){
            @Override
            public void actionPerformed(ActionEvent x){

                if (textfield_password.getText().equals(something)){
                    window5.setVisible(false);
                    MainPage window6 = new MainPage();
                    window6.setBounds(300,300,1000,100);
                    correct.setText("Correct Password");
                    window6.add(correct);
                    JButton button_obj5 = new JButton("Continue to Expense Calculator");
                    window6.add(button_obj5);

                    button_obj5.addActionListener(new UserPick1());
                    window6.setVisible(true);
                }
                else{
                    incorrect.setText("Password is Incorrect");
                    window5.add(incorrect);
                    window5.revalidate();
                    window5.repaint();
                }
            }

        });
}
```

- If a password has already been created, a window will appear to ask the user to enter the said password. If the password is correct another window will appear with a button to direct to the calculator. If it is incorrect the user has to try again.

```
private static class UserPick1 implements ActionListener{
    public void actionPerformed(ActionEvent h){
        MainPage2 window2 = new MainPage2();
        window2.setBounds(300,300,1000,400);
        window2.setDefaultCloseOperation(EXIT_ON_CLOSE);

        JPanel Income = new JPanel();
        Income.setLayout(new FlowLayout());
        Income.setBackground(Color.PINK);
        IncomeOutput.setText("User1 Income: " + user1_income());
        IncomeOutput.setForeground(Color.BLUE);
        Income.add(IncomeOutput);
        window2.add(Income);
```

- This is the code for the main calculator page for user1, which is in a separate class to use the ActionListener function. This is opened from the password class before and it creates a new window.

- MainPage2 has the parameters of a (3,1) GridLayout. The top element is a JPanel Income, which displays the user's income if it has been saved.

```
JPanel Expense = new JPanel();
ArrayList<String> listOfExpenses = user1_expenses();
Expense.setLayout(new GridLayout(listOfExpenses.size()+2,1));
Expense.setBackground(Color.PINK);
text.setText("User1 Expenses: ");
Expense.add(text);

if (listOfExpenses.size() == 0){
    JLabel ExpenseOutput = new JLabel();
    ExpenseOutput.setText("No Expenses Yet");
    Expense.add(ExpenseOutput);
}
else {
    for(int i = 0; i<listOfExpenses.size(); i++){

        JLabel ExpenseOutput = new JLabel();
        String holder = listOfExpenses.get(i);
        ExpenseOutput.setText(holder);
        Expense.add(ExpenseOutput);

    }
}
window2.add(Expense);
```

- This is the code to get the expenses from the expenses text file, and then output them in a list.
- This is put in a JPanel Expense, which comes after the JPanel Income.
- It takes the arraylist that is returned from the method user1_expenses(); and iterates through them, adding it to JLabel ExpenseOutput and adding that to JLabel Expense.
- Finally, the entire list in JLabel Expense is added to the main page.

```
static String user2Income = "user2 income.txt";
public static String user2_income(){
    try{
        File myObj = new File(user2Income);
        if(!myObj.exists()){
            myObj.createNewFile();
            System.out.println("File successfully created.");
        }
        else{
            Scanner myReader = new Scanner(myObj);
            while(myReader.hasNextLine()){
                String something = myReader.nextLine();
                return something;
            }
            myReader.close();
        }
    }catch(IOException e){
        System.out.println("An error occurred.");
    }
    return "No data yet";
}
```

- This is the code to fetch the income stat, which was called from the stat page.
- If there is no saved income, it will return “No data yet” as the text file is empty.
- If the file already exists then it will read what is there and return it.
- Following the success criteria in Criterion A, the data is saved from the last time the program was run, or a new file is created so that the program can be used on any computer. The saved data in the text files can also be sent to a different computer, if it is sent together in a folder with the program.

```java
static String user1Expense = "user1 expenses.txt";
public static ArrayList user1_expenses(){
    ArrayList<String> listOfExpenses = new ArrayList<String>();
    try{
        File myObj = new File(user1Expense);
        if(!myObj.exists()){
            myObj.createNewFile();
            System.out.println("File successfully created.");
        }
        else{
            FileReader fr = new FileReader("user1 expenses.txt");
            String s = new String();
            char ch;
            while (fr.ready()) {
                ch = (char)fr.read();
                if (ch == '|') {
                    listOfExpenses.add(s.toString());
                    s = new String();
                }
                else {
                    s += ch;
                }
            }
            if (s.length() > 0) {
                listOfExpenses.add(s.toString());
            }
            String[] array= listOfExpenses.toArray(new String[0]);

            return listOfExpenses;
        }
    }catch(IOException e){
        System.out.println("An error occurred .");
    }
    return listOfExpenses;
}
```

- This is the code for the method user1_expenses(), which takes the expenses saved in the text file and returns them in the form of an ArrayList.
- It uses a delimiter of “|”, which means that every time it encounters that character it will separate the string and that section then gets added to the ArrayList listOfExpenses.

```
JPanel Buttons = new JPanel();
Buttons.setBackground(Color.PINK);
Buttons.setLayout(new FlowLayout());
JButton button_obj3 = new JButton("Change Income");
Buttons.add(button_obj3);
button_obj3.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent i){
            MainPageIncome window3 = new MainPageIncome();
            window3.setBounds(300,300,1000,300);

            JPanel Enter = new JPanel();
            Enter.setBackground(Color.PINK);
            Enter.setLayout(new FlowLayout());

            EnterIncome.setText("Enter New Income");
            Enter.add(EnterIncome);

            File myObj = new File(user1Income);
            myObj.delete();
            Enter.add(textfield_income);
            user1_income();

            window3.add(Enter);
```

- This code is located inside the main stat page code, “UserPick1”, and comes after the two stats shown before. I am using inline action listeners so that variables can be shared from the main stat page.
- A new JPanel, Buttons, is created that is added after the JPanel Expense. The buttons “Change Income”, “Reset Expenses”, and “Add an Expense” are all in this JPanel.
- A textfield allows the user to input the new income.
- The code deletes the existing income text field and creates a new one.

```
JButton button_obj4 = new JButton("Save New Income");
window3.add(button_obj4);
button_obj4.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent j){
            try{
                FileWriter myWriter = new FileWriter(user1Income);
                myWriter.append(textfield_income.getText());
                myWriter.close();
            } catch (IOException f) {
                System.out.println("Failed to Save");
            }

            Expense.removeAll();
            ArrayList<String> listOfExpenses = user1_expenses();
            Expense.setLayout(new GridLayout(listOfExpenses.size()+2,1));
            text.setText("User1 Expenses: ");
            Expense.add(text);
            if (listOfExpenses.size() == 0){
                JLabel ExpenseOutput = new JLabel();
                ExpenseOutput.setText("No Expenses Yet");
                Expense.add(ExpenseOutput);
            }
            else {
                for(int i = 0; i<listOfExpenses.size(); i++){

                    JLabel ExpenseOutput = new JLabel();
                    String holder = listOfExpenses.get(i);
                    ExpenseOutput.setText(holder);
                    Expense.add(ExpenseOutput);

                }
            }
            IncomeOutput.setText("Your Income: " +user1_income());

            window2.revalidate();
            window2.repaint();

        }
```

- The button “Save New Income” comes right after the code above and uses another inline action listener inside the above action listener.
- This button appends the text from “textfield_income” and adds it to the newly created text file.
- It also uses the expense retrieval code from earlier to output updated expenses and updates accordingly on the main page.

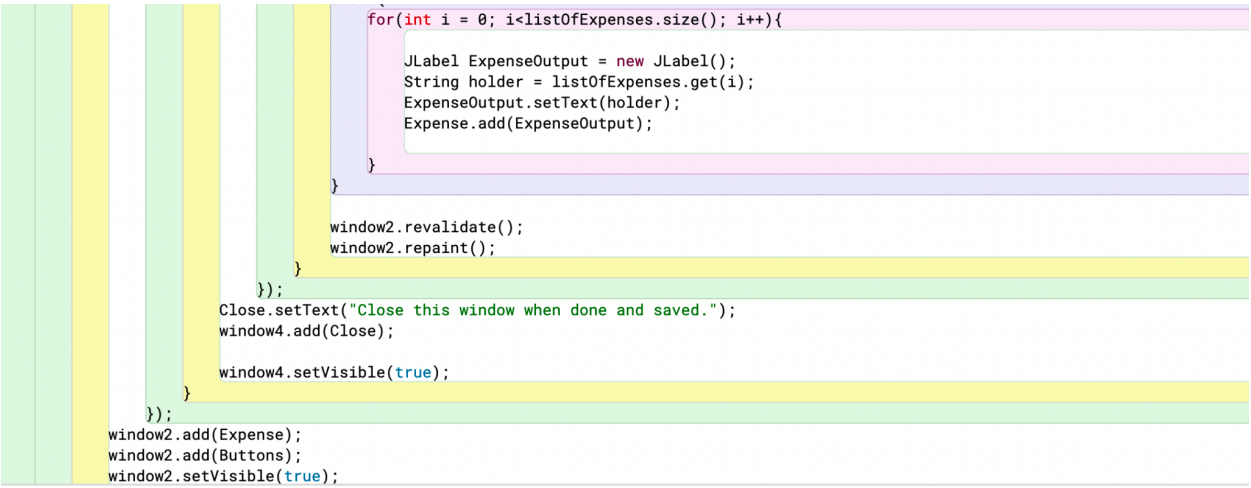
```
JButton button_obj4 = new JButton("Reset Expenses");
Buttons.add(button_obj4);
button_obj4.addActionListener(new ActionListener(){
        @Override
        public void actionPerformed(ActionEvent j){
            File myObj = new File(user1Expense);
            myObj.delete();
            Expense.removeAll();
            ArrayList<String> listOfExpenses = user1_expenses();
            Expense.setLayout(new GridLayout(listOfExpenses.size()+2,1));
            text.setText("User1 Expenses: ");
            Expense.add(text);
            if (listOfExpenses.size() == 0){
                JLabel ExpenseOutput = new JLabel();
                ExpenseOutput.setText("No Expenses Yet");
                Expense.add(ExpenseOutput);
            }
            else {
                for(int i = 0; i<listOfExpenses.size(); i++){

                    JLabel ExpenseOutput = new JLabel();
                    String holder = listOfExpenses.get(i);
                    ExpenseOutput.setText(holder);
                    Expense.add(ExpenseOutput);

                }
            }
            window2.revalidate();
            window2.repaint();
        }
    });
```

- This button “Reset Expenses” is located on the main stat page. The inline action listener deletes the existing expenses text field and updates the main stat page accordingly.

```
JButton button_obj5 = new JButton("Add an Expense");
Buttons.add(button_obj5);
button_obj5.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent j){
        MainPageExpense window4 = new MainPageExpense();
        window4.setBounds(300,300,1000,300);

        JPanel Enter = new JPanel();
        Enter.setBackground(Color.PINK);
        Enter.setLayout(new FlowLayout());

        EnterExpenseN.setText("Enter Expense Name");

        Enter.add(EnterExpenseN);
        Enter.add(textfield_expensename);

        EnterExpenseA.setText("Enter Expense Amount");
        Enter.add(EnterExpenseA);
        Enter.add(textfield_expenseamount);

        window4.add(Enter);
```

- This button “Add an Expense” is also an inline action listener from the main stat page.
- This creates a new window that has the user enter the expense name and amount.

```
JButton button_obj4 = new JButton("Save Expense");
window4.add(button_obj4);
button_obj4.addActionListener(new ActionListener(){
    @Override
    public void actionPerformed(ActionEvent j){
        try{
            File myObj = new File(user1Income);
            File myObj2 = new File(user1Expense);
            Scanner myReader = new Scanner(myObj);
            String income = "";
            String expenseA = "";
            expenseA=(textfield_expenseamount.getText());
            while(myReader.hasNextLine()){
                income = myReader.nextLine();

            }
            myReader.close();
            double i = Integer.parseInt(income);
            double e = Integer.parseInt(expenseA);
            double p = e/i*100;
            FileWriter myWriter = new FileWriter(user1Expense, true);
            myWriter.append(textfield_expensename.getText() + ", " +expenseA + ", " + String.valueOf(p)+"% of income |");
            myWriter.close();
        } catch (IOException e) {
            System.out.println("Expense could not be added");
        }

        Expense.removeAll();
        ArrayList<String> listOfExpenses = user1_expenses();
        Expense.setLayout(new GridLayout(listOfExpenses.size()+2,1));
        text.setText("User1 Expenses: ");
        Expense.add(text);
        if (listOfExpenses.size() == 0){
            JLabel ExpenseOutput = new JLabel();
            ExpenseOutput.setText("No Expenses Yet");
            Expense.add(ExpenseOutput);
        }
        else {
            for(int i = 0; i<listOfExpenses.size(); i++){
```



```
                    for(int i = 0; i<listOfExpenses.size(); i++){

                        JLabel ExpenseOutput = new JLabel();
                        String holder = listOfExpenses.get(i);
                        ExpenseOutput.setText(holder);
                        Expense.add(ExpenseOutput);

                    }
                }

                window2.revalidate();
                window2.repaint();
            }
        });
        Close.setText("Close this window when done and saved.");
        window4.add(Close);

        window4.setVisible(true);
    }
});
window2.add(Expense);
window2.add(Buttons);
window2.setVisible(true);
```

- This code is another inline action listener called “Save Expense” inside the “Add an Expense” action listener
- The button “Save Expense” takes the data from the income and expenses textfield and passes them into integers to calculate the percent income of each expense.
- It also adds a “|” character after every expense to separate them as a delimiter.
- All the data from the expenses text fields and the percent then get added to the expenses text file.
- It also updates the main stat page with the new added expense.