Vending Machine Project

By Yazad Rowhani
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>GUI Design</td>
<td>3</td>
</tr>
<tr>
<td>UML</td>
<td>5</td>
</tr>
<tr>
<td>Development of Code</td>
<td>6</td>
</tr>
<tr>
<td>Program Operation</td>
<td>8</td>
</tr>
<tr>
<td>Test Plan</td>
<td>16</td>
</tr>
<tr>
<td>White Box Testing</td>
<td>17</td>
</tr>
<tr>
<td>Bugs &amp; Weaknesses</td>
<td>20</td>
</tr>
<tr>
<td>References</td>
<td>22</td>
</tr>
<tr>
<td>List of Code</td>
<td>23</td>
</tr>
</tbody>
</table>
Program Design (GUI) & Widget Names

Design: `VendingMachine.java`

- **JButton**: bannerButton -> ImageIcon: bannerImage
- **selectLabel**: JLabel "Please select your items:"
  - productImage1: ImageIcon
  - product1: JButton
  - product1des
  - productPrice
  - productImage2: ImageIcon
  - product2: JButton
  - product2des
  - productPrice
  - productImage3: ImageIcon
  - product3: JButton
  - product3des
  - productPrice
  - productImage4: ImageIcon
  - product4: JButton
  - product4des
  - productPrice
  - productImage5: ImageIcon
  - product5: JButton
  - product5des
  - productPrice
  - productImage6: ImageIcon
  - product6: JButton
  - product6des
  - productPrice
  - productImage7: ImageIcon
  - product7: JButton
  - product7des
  - productPrice
  - productImage8: ImageIcon
  - product8: JButton
  - product8des
  - productPrice
  - productImage9: ImageIcon
  - product9: JButton
  - product9des
  - productPrice
  - productImage10: ImageIcon
  - product10: JButton
  - product10des
  - productPrice

- **moneyInputLabel**: JLabel "Please input your money:"
- **textField**: JTextField: selectedItems

- **selectedItemTotalLabel**: JLabel

- **moneyInputLabel**: JLabel "Please input your money:"
- **textField**: JTextField: moneyInput

- **JButton**: enterCoins
- **JButton**: returnCoins
- **JTextArea**: outputMessage
- **JButton**: takeChange

- **JLabel**: selectLabel
- **JLabel**: selectedLabel
- **JLabel**: selectedItemTotal

- **JButton**: fivePenceButton
- **JButton**: fiftyPenceButton
- **JButton**: onePoundButton
- **JButton**: tenPenceButton
- **JButton**: twoPoundButton
- **JButton**: twentyPenceButton

- **JButton**: currentDate
- **JButton**: currentTime

- **JButton**: h1 (button used as horizontal rule)
Design:  **VendingMachine.java – (JFrame : changeFrame)**

Note: Displayed when user has to collect change. Initiated when clicking the “takeChange” button shown on previous design. Also note, default close button (X) has been disabled on this frame to eliminate user error.
**UML**

**VendingMachine**

<table>
<thead>
<tr>
<th>Variable:</th>
<th>selectLabel : JLabel</th>
<th>productImage1 : ImageIcon</th>
<th>product1 : JButton</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>productImage1des : JLabel</td>
<td>product1price : JLabel</td>
<td>Product1price : JLabel</td>
</tr>
<tr>
<td></td>
<td>productImage2 : ImageIcon</td>
<td>product2 : JButton</td>
<td>Product2price : JLabel</td>
</tr>
<tr>
<td></td>
<td>productImage3 : ImageIcon</td>
<td>product3 : JButton</td>
<td>Product13price : JLabel</td>
</tr>
<tr>
<td></td>
<td>product3des : JLabel</td>
<td>Product3price : JLabel</td>
<td>Product13price : JLabel</td>
</tr>
<tr>
<td></td>
<td>Product1price : JLabel</td>
<td>product4 : JButton</td>
<td>Product4price : JLabel</td>
</tr>
<tr>
<td></td>
<td>productImage7 : ImageIcon</td>
<td>product7 : JButton</td>
<td>Product7price : JLabel</td>
</tr>
<tr>
<td></td>
<td>product7des : JLabel</td>
<td>Product7price : JLabel</td>
<td>Product7price : JLabel</td>
</tr>
<tr>
<td></td>
<td>productImage8 : ImageIcon</td>
<td>product8 : JButton</td>
<td>Product8price : JLabel</td>
</tr>
<tr>
<td></td>
<td>product8des : JLabel</td>
<td>Product8price : JLabel</td>
<td>Product8price : JLabel</td>
</tr>
<tr>
<td></td>
<td>productImage11 : ImageIcon</td>
<td>selectedItems : JTextArea</td>
<td>selectedItemsTotal : JTextField</td>
</tr>
<tr>
<td></td>
<td>productImage12 : ImageIcon</td>
<td>selectedItems : JTextArea</td>
<td>selectedItemsTotal : JTextField</td>
</tr>
<tr>
<td></td>
<td>productImage13 : ImageIcon</td>
<td>moneyInput : JTextField</td>
<td>outputMessage : JTextArea</td>
</tr>
</tbody>
</table>

- `addToStringSelection : String`
- `totalInserted : double`
- `poundsFormat : DecimalFormat`
- `numFormat : DecimalFormat`
- `twoFormat : DecimalFormat`
- `decFormat : DecimalFormat`
- `productTotal : double`
- `totalInserted : double`
- `errorGiveChange : double`
- `product1stockNum : double`
- `product2stockNum : double`
- `totalFW : String`
- `inputFW : String`
- `changePH : double`
- `errorInputShort : double`

Method: `-closeChangeFrame(), -changeFrameInit(), -VendingMachine(), +FileHandler, +VendingMachineDBHandler, +DataHandler`
Code Development

The first step of creating this program was to design the interface, as seen in the design section of this report, which includes the UML. From this, I was able to begin coding my GUI, which involved a lot of planning on paper, as I used the null layout, and precise pixel positioning of all elements to get the perfect look I desired. Although this took some time, I felt it was worth it to achieve what I exactly wanted, as the interface is a key part of the program and how people interact with it.

Once I had played around with the layout and objects in the GUI, I moved onto coding how the program actually works. This entire section takes place in the action performed section of the code, where all the event handling goes.

My first task was to get the product selection to work, which was easy as it just meant when the user clicked the product, it displayed the product in the selected items section. The harder part was figuring out a method to do the calculations.

At first, I immediately thought of using the ‘getText’ method, where the string from the text field which displays the cost is taken and then parsed into a double. However, as my text fields contained the £ symbol, this threw out several errors. So instead, I used a double to calculate all the numbers in the background, then just used the ‘setText’ method with a pound formatting, in the format of “£#.##”. This worked as I anticipated. I could have used the getText and parsing method if I wanted, and just put the “£” symbol outside of the textfield, but I found this method to be sloppy, and not as efficient as using the double methods, and then just either “-=” or “+=” the numbers to devise the calculations.

Using this method was simple, for example, for a simple order, the user would select a product, which when clicked would initiate “productTotal += 0.5;” which would add “50 pence” to the productTotal double value. Then, the following would be initiated “selectedItemTotal.setText(poundsFormat.format(productTotal));” which sets the double value to the selected items total textfield, and adds a format to it (£#.##). This method is effective, as the user can select several products, and the “+=” would always update the value giving a running total for that session.

I used this same method for the stock of my products and coins, for example, if a user selected product1, it would run “product1stockNum -= 1”, and this number would then be applied to the stock text field next to the product. For coins inserted, the stock would increase, and then this same number would decrease when the user is taken to the change frame to select their change. This system works well, and was fairly simple to implement.

The change frame is initiated when the user has change to select. Here, the user can select exactly how they would like their change. I had some trouble with this to begin with. I got the frame to open when the “Take Change” button is pressed (which is only enabled when the user has change to take), but getting it to close was a problem. This was because I had disabled the default close button (the X in the corner), to eliminate user error, for example someone closing the window before taking their change. This meant I had to enable the frame to close via a button (which only enabled itself when all the change was taken). At first, I attempted to use the “dispose()” function, but this
created several errors and seemed very complicated to get right. After many attempts to correct this, I decided to use a hide method, where the frame is never really closed, just not made visible at the right times (“changeFrame.setVisible(false);”). Although this method seems sloppy, it makes it a lot easier to keep the coin stock number levels relevant, than using the dispose function, which I couldn’t get to function as intended anyway.

At this point, I had a fully functional vending machine with a nice interface. The next step was to develop the file writer and database, so that all the data such as date and time of purchase, products selected and coins input was stored in both a file writer and an access database. For both of these, I had to create separate classes, for the filewriter – FileHandler.java, and for the database – VendingMachineDBHandler.java.

I found the file handler to be the easier of the two, as the database had some complex requirements to get it working, such as importing the JDBC / ODBC drivers. Also, the syntax for getting the field of the table was very hard to get right, as there were single and double quotes which made it very difficult to get exactly right and took some tries.

Once I had a well made vending machine, it was a matter of touching it up. I added images to almost all my buttons, such as close icons, enter and return coin icons, to make it more user friendly and look more appealing. I added ‘extra’ messages for some realism, for example, when a user orders a can of drink, the message appears “Open with caution, may be shaken”.

I had used almost everything learnt from the entire year to put this program together, with a few exceptions. I used many examples from programs already made in lectures and tutorials, and help from fellow students on the same course. I also researched many points on how to implement additional frames, and the date/time feature, all of which can be found in the referencing section (refer to contents). Overall, I enjoyed creating this program, and learnt a lot in doing this coursework independently.
Program Operation

Here is the main interface for the Vending Machine. As you can see, it is laid out well and appears easy to use for novice users. The user can select any of the 10 products shown, by clicking on the image of the product.

Here, 3 products have been selected (1x Mars Bar, 1x Walkers Ready Salted Crisps and 1x Coca-Cola (330ml)), and they will appear in the selected products section.

The total amount that the user has to input is shown just below this.
The user must then move onto the money input section, where he/she must select which coins to insert by clicking on the images. The information below shows how much the user has input, and this amount must match the amount of the total to be inserted, in the previous section to the left.

In this example, the exact amount has been inserted, and as you can see in the stock increases, the user has input 1x £1 Coin, 1x 50p Coin, 1x 20p Coin and 1x 10p Coin.

At any time, the user can click the “Cancel / Clear order” button, which will reset the vending machine to its original state before the order.

Once the user has input the correct amount, or more than required, or any input (even none) for that matter, they must click the Enter Coins button. A message will appear based on the input, in the message output text area (take note of the colour co-ordination, red for bad outcomes (e.g. not enough input), and green for good outcomes (money accepted, etc.)

Here are some of the possible outcomes:

If no money has been input:

---

End Please input your money using the Input Money Section.
You need to input £1.80!
If correct amount has been input:

![Image of a device with buttons to enter or return coins, a display saying £1.00 Accepted, and a message Thank you for your purchase.]

If not enough money has been input (In this case, only £1):

![Image of a device showing a list of items selected: Mars Bar (£0.50), Walker’s Ready Salted Crisps (£0.55), Coca-Cola (330ml) (£0.75). The total cost is £1.80. The user has inputted £1.00. The message says You still have £0.80 to pay.]

If too much money is input, the amount is accepted, the change is calculated, and take note that the ‘take change’ button now displays:

![Image of a device showing a list of items selected: Walkers Sensations (£1.10), Coca-Cola (£0.75), Dr Pepper (£0.75), Fruit Pastels (£0.60), Wine Gums (£0.65). The total cost is £5.45. The user has inputted £6.00. The message says £0.50 Accepted! Thank you for your purchase! Please take your £0.50 change. Press the ‘Take Change’ button below.]
Also note, all other buttons have been disabled, as this limits user error, adds focus, and looks professional. It is also relatively simple to do.

Once the user clicks the ‘take change’ button, a new window appears (the secondary JFrame – changeFrame (see design)) which allows the user to select their exact change:

Following from the previous example (where £2.50 was inserted in a £1.80 total), here is the changeFrame that would appear:

![Change Frame Example]

Note the “change remaining” states correctly that the user needs to take £0.70, and also that the stock levels of the money input (1x £2 Coin and 1x 50p Coin) is displayed. Now, when the user takes 70 pence, in whatever form, the stock of the money is deducted.

In this case, the user has selected 3x 20 pence coins:

Take note of the stock updates.
Note, the change remaining updates dynamically, stating that there is only ten pence remaining to take.

If the user were to click the “finish” button now, the following would be displayed in the output message:

![Change Remaining: £0.10](image)

Now, when the user inputs the final 10 pence, and presses the finished button, the following is displayed:

![Please Take Your Change](image)

Take note, the “close this window” button has appeared once the correct amount of change has been taken. This is because the default close button on the window has been disabled, so that users don’t close the window by accident.

Once the user clicks this close button, they are returned to the main frame, and all the buttons are re-enabled.
Other points to note when using the program:

Product Stock Depletion:

Each product has its own stock level. Here, I will demonstrate what happens when a product stock is depleted:

The product button is disabled while the product is out of stock, and a warning message appears.

No product selected:

If the user does not select a product at all, the following message will be displayed in the output message text area:
Coin stock depleted:

The coin stock which is depleted is disabled, and a warning message appears.

File writer:

The program will write any order into the file: “VendingMachineData.txt”, as shown in the example:

Selected products include 1x Mars Bar, and 1x Coca-Cola, which totals at £1.25. Total amount inserted is £2.

Shown here is the “VendingMachineData.txt” file, with the correct data, including time and date of purchase.
Database:

In this example, the user has selected Mars Bar and Coca Cola:

Now if we go directly to the database (VendingMachineDB.mdb) we can see they have been correctly written to the database.
<table>
<thead>
<tr>
<th>Test No.</th>
<th>Test Case</th>
<th>Expected Output</th>
<th>Actual Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select Mars Bar, Input £1</td>
<td>Accepted, 50 Pence Change</td>
<td>Accepted, 50 Pence Change</td>
</tr>
<tr>
<td>2</td>
<td>Select Bounty Bar, Input £2</td>
<td>Accepted, 1.50 Pence Change</td>
<td>Accepted, 1.50 Pence Change</td>
</tr>
<tr>
<td>3</td>
<td>Select Mars Bar x10</td>
<td>Out of Stock warning, £5.00 charge</td>
<td>Out of Stock warning, £5.00 charge</td>
</tr>
<tr>
<td>4</td>
<td>Insert £1, press ‘Return Coins’</td>
<td>£1 returned</td>
<td>£1 returned</td>
</tr>
<tr>
<td>5</td>
<td>Press ‘Return Coins’</td>
<td>Error – No money input</td>
<td>Error – No money input</td>
</tr>
<tr>
<td>6</td>
<td>Insert £1, press “Enter Coins”</td>
<td>Error – No product selected</td>
<td>Error – No product selected</td>
</tr>
<tr>
<td>7</td>
<td>Taking £3 change, take all 5 pence coins</td>
<td>Stock warning – No 5 pence coins left</td>
<td>Stock warning – No 5 pence coins left</td>
</tr>
<tr>
<td>8</td>
<td>Select Mars Bar, insert £1, press “Cancel Order”</td>
<td>Order cleared, coins returned</td>
<td>Order cleared, coins returned, stock not reset</td>
</tr>
<tr>
<td>9</td>
<td>50 pence change to take, select £1 at change frame</td>
<td>Error – cannot take that much</td>
<td>Error – cannot take that much</td>
</tr>
<tr>
<td>10</td>
<td>50 pence change to take, select 20 pence, press “finished”</td>
<td>Error – still 30 pence to take</td>
<td>Error – still 30 pence to take</td>
</tr>
<tr>
<td>11</td>
<td>Select Fruit Pastels, press “Enter Coins”</td>
<td>Error – Need to input 60p</td>
<td>Error – Need to input 60p</td>
</tr>
<tr>
<td>12</td>
<td>Select Fruit Pastels, input 50 pence, press “Enter Coins”</td>
<td>Error – Need to input 50p</td>
<td>Error – Need to input 50p</td>
</tr>
<tr>
<td>13</td>
<td>Select Mars Bar, press “Return Coins”</td>
<td>Error – No money Input</td>
<td>Error – No money Input</td>
</tr>
<tr>
<td>14</td>
<td>Select Bounty Bar, insert £1, press “Enter Coins”</td>
<td>File writer to display Bounty Bar, 50 pence, £1 inserted &amp; Date / Time</td>
<td>File writer to display Bounty Bar, 50 pence, £1 inserted &amp; Date / Time</td>
</tr>
<tr>
<td>15</td>
<td>Select Mars Bar, insert 50 pence, press “Enter Coins”</td>
<td>Database to display Mars Bar, 50 pence inserted, and Date / Time</td>
<td>Database to display Mars Bar, 50 pence inserted, and Date / Time</td>
</tr>
<tr>
<td>16</td>
<td>Select Coca-Cola, insert £1, press “Enter Coins”, press “Finished” at change frame</td>
<td>Error – Please take change</td>
<td>Error – Please take change</td>
</tr>
<tr>
<td>17</td>
<td>Select Coca-Cola, insert £1, press “Enter Coins”, press X (close) at change frame</td>
<td>No Action</td>
<td>No Action</td>
</tr>
<tr>
<td>18</td>
<td>Insert £1, select Mars Bar x3, Press Enter Coins</td>
<td>Error – 50p needs to be inserted</td>
<td>Error – 50p needs to be inserted</td>
</tr>
<tr>
<td>19</td>
<td>Insert £1.50, Mars Bar x3, Enter Coins</td>
<td>Accepted, No change.</td>
<td>Accepted, No change.</td>
</tr>
<tr>
<td>20</td>
<td>Take 50p change, return to main frame</td>
<td>Stock update on main frame (-50p x1)</td>
<td>Stock not updated.</td>
</tr>
</tbody>
</table>
White Box Testing

White box testing of user buying product:

VendingMachine

Select Product & Press “Enter Coins”

If Coins Entered

If Exact Amount

Error – Please Input Money

Coins Accepted

Coins Accepted – Calculate Change

Note:

O = Option
T = True
F = False
White box testing of user taking 20 pence change:

VendingMachine : ChangeFrame

10 pence Coin Selected

If Pressed Finished Button

Select 10 pence, press “Finished” button.

Error – Please take a further 10 pence

Coins Accepted, close button enabled

Cannot close frame
White box testing of user returning coins:

VendingMachine

Mars Bar selected and £1 inserted

If Enter Coins pressed
Order accepted, 50p returned

If Return Coins pressed
£1 returned

Order reset
Bugs and Weaknesses

Bugs

1. Product stock bug:

Each product starts with 10 stock in my vending machine. If a user was to buy, for example, 10 mars bars, the stock warning would appear and the button for the mars bar would be disabled, as shown:

But now, if the user was to insert £6, go to the change frame, take their change, and return to the main frame, the mars bar button would be reenabled, and they would be able to go into negative stock numbers. I attempted to resolve this, by implementing an if statement when the close change frame button was clicked, that would check the stock levels of products, and if it was 0, would ensure the button remained disabled. This did not work, and I was unable to resolve this bug.

2. Time and Date:

I thought it would be nice to implement a time and date feature in my Vending Machine, which would display the current date and time. I wanted them displayed in two separate text fields at the top right of the main frame. I used examples from Java Workbook 1, using code such as ‘int hours = now.getHours();    int minutes = now.getMinutes();’ etc..

However, this only displayed the date and time when the program was run, and didn’t update every second as it should have. I researched many methods online on how to get this to work as intended, and tried using the ‘Timer’ function, using threads, but nothing worked. In the end, I didn’t have enough time to perfect this, but decided to leave it in anyway.

3. Coin Stock Not Updating when switching Frames:

The stock level of coins, which increases when the user inserts coins, and decreases as they take change, does not communicate well between the two frames. This can create a problem, as the stock levels will not make sense when the program has been operating for a long period of time, and is a fairly problematic bug.
Weaknesses

User not able to select FW or DB

The user has no option whether or not to select the file writer method or the database. In its current state, the program simply writes to both, and based on which ever the user has (including both), he/she can use that. It would be better if the user had a choice, because if they didn’t have a database program installed, it would throw out errors.

I did attempt to implement this method, but found it difficult to do so. My plan was to initiate a message dialog input as soon as the program was loaded, and then the user would input either the word File or DB, depending on which, would correctly select the desired writer method.
References

- **Introduction to Programming - Part 1** – Used for date and time code mainly, and other general ideas including Don Cowell’s car park program
  e.g. productStock += 1

- **Introduction to Programming - Part 2** – Used for file writer and database examples

  Help and ideas on multiple JFrames. Code not directly used.

- [http://www.java-tips.org/java-se-tips/java.util/how-to-get-current-date-time.html](http://www.java-tips.org/java-se-tips/java.util/how-to-get-current-date-time.html)
  Resource to obtain current date & time in Java. Again, not used, just gave ideas.

- **InOutButton.java & InOutButton2.java** – Examples from Kate Finney from Teachmat, used in code (see comments in code for usage).
List of Code:

VendingMachine.java (Main)

```java
import java.text.DecimalFormat;
import java.util.Date;
import static javax.swing.JOptionPane.*;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.sql.*;

public class VendingMachine extends JFrame
    implements ActionListener
{
    //Here I will declare all my objects to be used, with their properties too.
    //datahandler
    static DataHandler dh;
    //date and time
    Date now = new Date();
    //date:
    int year = now.getYear() + 1900;
    int month = now.getMonth() + 1;
    int day = now.getDate();
    //time:
    int hours = now.getHours();
    int minutes = now.getMinutes();
    int seconds = now.getSeconds();

    //changeFrame
    JFrame changeFrame = new JFrame("Please Take Your Change");

    //Banner Image and button:
    ImageIcon bannerImage = new ImageIcon("images/bannerIcon.jpg");
    JButton bannerButton = new JButton("", bannerImage);

    JLabel selectLabel = new JLabel("Please select your items:");

    //All objects for Product 1:
    ImageIcon productImage1 = new ImageIcon("images/productImage1.jpg");
    JButton product1 = new JButton("", productImage1);
    JLabel product1des = new JLabel("Mars Bar", JLabel.CENTER);
    JLabel product1price = new JLabel("£0.50", JLabel.CENTER);

    //All objects for Product 2:
    ImageIcon productImage2 = new ImageIcon("images/productImage2.gif");
    JButton product2 = new JButton("", productImage2);
    JLabel product2des = new JLabel("Bounty Bar Original", JLabel.CENTER);
    JLabel product2price = new JLabel("£0.50", JLabel.CENTER);

    //All objects for Product 3:
    ImageIcon productImage3 = new ImageIcon("images/productImage3.jpg");
    JButton product3 = new JButton("", productImage3);
    JLabel product3des = new JLabel("Twix Double Bar", JLabel.CENTER);
```
JLabel product3price = new JLabel("£0.65", JLabel.CENTER);

//All objects for Product 4:
ImageIcon productImage4 = new ImageIcon("images/productImage4.jpg");
JButton product4 = new JButton("", productImage4);
JLabel product4des = new JLabel("Walker's Ready Salted", JLabel.CENTER);
JLabel product4des2 = new JLabel("Crisps", JLabel.CENTER);
JLabel product4price = new JLabel("£0.55", JLabel.CENTER);

//All objects for product 5:
ImageIcon productImage5 = new ImageIcon("images/productImage5.jpg");
JButton product5 = new JButton("", productImage5);
JLabel product5des = new JLabel("Monster Munch -", JLabel.CENTER);
JLabel product5des2 = new JLabel("Pickled Onion", JLabel.CENTER);
JLabel product5price = new JLabel("£0.70", JLabel.CENTER);

//All objects for product 6:
ImageIcon productImage6 = new ImageIcon("images/productImage6.jpg");
JButton product6 = new JButton("", productImage6);
JLabel product6des = new JLabel("Walkers Sensations -", JLabel.CENTER);
JLabel product6des2 = new JLabel("Thai Sweet Chilli", JLabel.CENTER);
JLabel product6price = new JLabel("£1.10", JLabel.CENTER);

//All objects for product 7:
ImageIcon productImage7 = new ImageIcon("images/productImage7.jpg");
JButton product7 = new JButton("", productImage7);
JLabel product7des = new JLabel("Coca-Cola (330ml)", JLabel.CENTER);
JLabel product7price = new JLabel("£0.75", JLabel.CENTER);

//All objects for product 8:
ImageIcon productImage8 = new ImageIcon("images/productImage8.jpg");
JButton product8 = new JButton("", productImage8);
JLabel product8des = new JLabel("Dr Pepper (330ml)", JLabel.CENTER);
JLabel product8price = new JLabel("£0.75", JLabel.CENTER);

//All objects for product 9:
ImageIcon productImage9 = new ImageIcon("images/productImage9.jpg");
JButton product9 = new JButton("", productImage9);
JLabel product9des = new JLabel("Fruit Pastels", JLabel.CENTER);
JLabel product9price = new JLabel("£0.60", JLabel.CENTER);

//All objects for product 10:
ImageIcon productImage10 = new ImageIcon("images/productImage10.jpg");
JButton product10 = new JButton("", productImage10);
JLabel product10des = new JLabel("Wine Gums", JLabel.CENTER);
JLabel product10price = new JLabel("£0.65", JLabel.CENTER);

//Horizontal/Vertical Rules as buttons for simplicity and layout
JButton ruleH = new JButton("");
JButton ruleV = new JButton("");
JButton ruleV2 = new JButton("");

//selected items section
JLabel selectedLabel = new JLabel("You have selected the following items:.");
JTextArea selectedItems = new JTextArea(8,250);
//Total price:
JLabel selectedItemTotalLabel = new JLabel("Total - Please insert: ");
JTextField selectedItemTotal = new JTextField("£0.00", 30);

//Money section:

//Label:
JLabel moneyInputLabel = new JLabel("Please input your money: ");

//Money buttons with imageIcons:

//5 pence
ImageIcon fivePenceImage = new ImageIcon("images/fivePenceImage.jpg");
JButton fivePenceButton = new JButton("", fivePenceImage);

//10 pence
ImageIcon tenPenceImage = new ImageIcon("images/tenPenceImage.jpg");
JButton tenPenceButton = new JButton("", tenPenceImage);

//20 pence
ImageIcon twentyPenceImage = new ImageIcon("images/twentyPenceImage.jpg");
JButton twentyPenceButton = new JButton("", twentyPenceImage);

//50 pence
ImageIcon fiftyPenceImage = new ImageIcon("images/fiftyPenceImage.jpg");
JButton fiftyPenceButton = new JButton("", fiftyPenceImage);

//1 pound
ImageIcon onePoundImage = new ImageIcon("images/onePoundImage.jpg");
JButton onePoundButton = new JButton("", onePoundImage);

//2 pound
ImageIcon twoPoundImage = new ImageIcon("images/twoPoundImage.jpg");
JButton twoPoundButton = new JButton("", twoPoundImage);

//You have input:
JLabel moneyInputLabel2 = new JLabel("You have input: ");
JTextField moneyInput = new JTextField("£0.00", 30);

//Enter / return coins
ImageIcon enterCoinsImage = new ImageIcon("images/entercoins.jpg");
JButton enterCoins = new JButton("Enter Coins", enterCoinsImage);
ImageIcon returnCoinsImage = new ImageIcon("images/returncoins.jpg");
JButton returnCoins = new JButton("Return Coins", returnCoinsImage);

//Output message window (Thank you for purchase / £2.10 Returned / etc)
JTextArea outputMessage = new JTextArea(2,30);

//cancel order objects
JLabel cancelOrderLabel = new JLabel("Cancel / Clear Order: ");
ImageIcon cancelOrderImage = new ImageIcon("images/cancelOrderImage.jpg");
JButton cancelOrder = new JButton("", cancelOrderImage);

//product stock levels all set at 10 stock per item
JTextField product1stock = new JTextField("10", 2);
JTextField product2stock = new JTextField("10", 2);
JTextField product3stock = new JTextField("10", 2);
JTextField product4stock = new JTextField("10", 2);
JTextField product5stock = new JTextField("10", 2);
JTextField product6stock = new JTextField("10", 2);
JTextField product7stock = new JTextField("10", 2);
JTextField product8stock = new JTextField("10", 2);
JTextField product9stock = new JTextField("10", 2);
JTextField product10stock = new JTextField("10", 2);

//coin stock
JTextField fivepStock = new JTextField("20", 2);
JTextField tenpStock = new JTextField("20", 2);
JTextField twentypStock = new JTextField("20", 2);
JTextField fiftypStock = new JTextField("20", 2);
JTextField onepStock = new JTextField("20", 2);
JTextField twopStock = new JTextField("20", 2);

//take change button
JButton takeChange = new JButton("Take Change");

//date and time textfields & labels and calls
JLabel dateLabel = new JLabel("Date:");
JLabel timeLabel = new JLabel("Time:");
JTextField currentDate = new JTextField("", 10);
JTextField currentTime = new JTextField("", 10);

//declarations for changeFrame:
ImageIcon cfLabelicon = new ImageIcon("images/CFLabel1icon.jpg");
JLabel cfLabelTakeChange = new JLabel("", cfLabelicon, JLabel.CENTER);

//Money buttons with imageicons for changeFrame (imageicons used same as previous)
//5 pence
JButton CFfivePenceButton = new JButton("", fivePenceImage);
//10 pence
JButton CFtenPenceButton = new JButton("", tenPenceImage);
//20 pence
JButton CفتwentyPenceButton = new JButton("", twentyPenceImage);
//50 pence
JButton CFfiftyPenceButton = new JButton("", fiftyPenceImage);
//1 pound
JButton CFonePoundButton = new JButton("", onePoundImage);
//2 pound
JButton CFtwoPoundButton = new JButton("", twoPoundImage);

//coin stock CF
JTextField CFfivepStock = new JTextField("20", 2);
JTextField CFtenpStock = new JTextField("20", 2);
JTextField CفتwentypStock = new JTextField("20", 2);
JTextField CFfiftypStock = new JTextField("20", 2);
JTextField CFonepStock = new JTextField("20", 2);
JTextField CFtwopStock = new JTextField("20", 2);

JLabel CFpleaseTakeChange = new JLabel("Please select how you would like your change:");
JLabel CFchangeRemainingLabel = new JLabel("Change Remaining: £");
public static void main(String[] args) {
    VendingMachine jf = new VendingMachine();
    dh = new FileHandler();//data handler for file writer
}
public VendingMachine()
{
    setLayout(null); // Null layout for absolute positioning
    setSize(830, 705); // Dimensions 830x705
    setTitle("Super Awesome Vending Machine"); // Title
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // banner and initial label
    bannerButton.setBounds(210, 10, 400, 30);
    add(bannerButton);
    selectLabel.setBounds(10, 50, 200, 15);
    add(selectLabel);

    // date and time objects (labels and textfields)
    currentDate.setBounds(740, 5, 80, 20);
    currentDate.setEditable(false);
    add(currentDate);
    dateLabel.setBounds(705, 7, 30, 15);
    add(dateLabel);
    currentTime.setBounds(740, 28, 80, 20);
    currentTime.setEditable(false);
    add(currentTime);
    timeLabel.setBounds(702, 30, 33, 15);
    add(timeLabel);

    // row 1 of product image buttons
    product1.setBounds(20, 75, 140, 150);
    product1.addActionListener(this);
    add(product1);
    product2.setBounds(180, 75, 140, 150);
    product2.addActionListener(this);
    add(product2);
    product3.setBounds(340, 75, 140, 150);
    product3.addActionListener(this);
    add(product3);
    product4.setBounds(500, 75, 140, 150);
    product4.addActionListener(this);
    add(product4);
    product5.setBounds(660, 75, 140, 150);
    product5.addActionListener(this);
    add(product5);

    // row 2 = row 1 product descriptions and prices
    product1des.setBounds(20, 225, 140, 15);
    add(product1des);
    product1price.setBounds(20, 250, 140, 15);
    product1price.setForeground(Color.red);
    add(product1price);
    product2des.setBounds(180, 225, 140, 15);
    add(product2des);
    product2price.setBounds(180, 250, 140, 15);
    product2price.setForeground(Color.red);
    add(product2price);
product3des.setBounds(340,225,140,15);
add(product3des);
product3price.setBounds(340,250,140,15);
product3price.setForeground(Color.red);
add(product3price);
product4des.setBounds(500,225,140,15);
add(product4des);
product4price.setBounds(500,250,140,15);
product4price.setForeground(Color.red);
add(product4price);
product5des.setBounds(660,225,140,15);
add(product5des);
product5price.setBounds(660,250,140,15);
add(product5price);

//row 3 = 2nd row of product image buttons (products 6-10)
product6.setBounds(20,270,140,150);
product6.addActionListener(this);
add(product6);
product7.setBounds(180,270,140,150);
product7.addActionListener(this);
add(product7);
product8.setBounds(340,270,140,150);
product8.addActionListener(this);
add(product8);
product9.setBounds(500,270,140,150);
product9.addActionListener(this);
add(product9);
product10.setBounds(660,270,140,150);
product10.addActionListener(this);
add(product10);

//row 4 = row 3 product descriptions and prices
product6des.setBounds(20,420,140,15);
add(product6des);
product6price.setBounds(20,445,140,15);
product6price.setForeground(Color.red);
add(product6price);
product7des.setBounds(180,420,140,15);
add(product7des);
product7price.setBounds(180,445,140,15);
product7price.setForeground(Color.red);
add(product7price);
product8des.setBounds(340,420,140,15);
add(product8des);
product8price.setBounds(340,445,140,15);
product8price.setForeground(Color.red);
add(product8price);
product9des.setBounds(500,420,150,15);
add(product9des);
product9price.setBounds(500,445,150,15);
product9price.setForeground(Color.red);
add(product9price);
product10des.setBounds(660,420,150,15);
add(product10des);
product10price.setBounds(660,445,150,15);
product10price.setForeground(Color.red);
add(product10price);

//product stock fields
product1stock.setBounds(161,75,17,20);
product1stock.setEditable(false);
add(product1stock);

product2stock.setBounds(321,75,17,20);
product2stock.setEditable(false);
add(product2stock);

product3stock.setBounds(481,75,17,20);
product3stock.setEditable(false);
add(product3stock);

product4stock.setBounds(641,75,17,20);
product4stock.setEditable(false);
add(product4stock);

product5stock.setBounds(801,75,17,20);
product5stock.setEditable(false);
add(product5stock);

product6stock.setBounds(161,270,17,20);
product6stock.setEditable(false);
add(product6stock);

product7stock.setBounds(321,270,17,20);
product7stock.setEditable(false);
add(product7stock);

product8stock.setBounds(481,270,17,20);
product8stock.setEditable(false);
add(product8stock);

product9stock.setBounds(641,270,17,20);
product9stock.setEditable(false);
add(product9stock);

product10stock.setBounds(801,270,17,20);
product10stock.setEditable(false);
add(product10stock);

//horizontal rule to seperate products from the lower section
ruleH.setBounds(15,465,800,2);
add(ruleH);

//Selected products section
selectedLabel.setBounds(15, 475, 220, 15);
add(selectedLabel);

selectedItems.setBounds(15, 495, 265, 140);
selectedItems.setEditable(false);
add(selectedItems);

// total price
selectedItemTotalLabel.setBounds(15, 645, 120, 20);
add(selectedItemTotalLabel);
selectedItemTotal.setBounds(140, 645, 125, 20);
selectedItemTotal.setEditable(false);
add(selectedItemTotal);

// vert rule separating the selected items from the money section
ruleV.setBounds(283, 475, 2, 185);
add(ruleV);

// money input section:

// money label:
moneyInputLabel.setBounds(290, 470, 150, 20);
add(moneyInputLabel);

// 5p
fivePenceButton.setBounds(290, 495, 68, 68);
fivePenceButton.addActionListener(this);
add(fivePenceButton);

// 10p
tenPenceButton.setBounds(379, 495, 68, 68);
tenPenceButton.addActionListener(this);
add(tenPenceButton);

// 20p
twentyPenceButton.setBounds(471, 495, 68, 68);
twentyPenceButton.addActionListener(this);
add(twentyPenceButton);

// 50p
fiftyPenceButton.setBounds(290, 565, 68, 68);
fiftyPenceButton.addActionListener(this);
add(fiftyPenceButton);

// £1
onePoundButton.setBounds(379, 565, 68, 68);
onePoundButton.addActionListener(this);
add(onePoundButton);

// £2
twoPoundButton.setBounds(471, 565, 68, 68);
twoPoundButton.addActionListener(this);
add(twoPoundButton);

// money stock values
fivepStock.setBounds(358,495,18,20);
fivepStock.setEditable(false);
add(fivepStock);

tenpStock.setBounds(447,495,18,20);
tenpStock.setEditable(false);
add(tenpStock);

twentypStock.setBounds(539,495,18,20);
twentypStock.setEditable(false);
add(twentypStock);

fiftyStock.setBounds(358,565,18,20);
fiftyStock.setEditable(false);
add(fiftyStock);

onepStock.setBounds(447,565,18,20);
onepStock.setEditable(false);
add(onepStock);

twopStock.setBounds(539,565,18,20);
twopStock.setEditable(false);
add(twopStock);

//money input section:
moneyInputLabel2.setBounds(290,638,100,20);
add(moneyInputLabel2);
moneyInput.setBounds(380,638,127,20);
moneyInput.setEditable(false);
add(moneyInput);

//vert rule seperating money section from right side section
ruleV2.setBounds(560,475,2,185);
add(ruleV2);

//enter / return coins / output area
enterCoins.setBounds(568,475,250,40);
enterCoins.addActionListener(this);
add(enterCoins);
returnCoins.setBounds(568,525,250,40);
returnCoins.addActionListener(this);
add(returnCoins);
outputMessage.setBounds(568,575,250,50);
outputMessage.setEditable(false);
add(outputMessage);

//cancel order
cancelOrderLabel.setBounds(676,635,122,25);
cancelOrderLabel.addCancelOrderLabel();
cancelOrder.setBounds(794,635,25,25);
cancelOrder.addActionListener(this);
add(cancelOrder);

//take change
takeChange.setBounds(565,635,109,25);
takeChange.setForeground(Color.red);
takeChange.setVisible(false);
takeChange.addActionListener(this);
add(takeChange);

//end pixel definition placing

//database handling - adapted from Kate Finney's example
try {
    Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
    String sourceURL = new String("jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=VendingMachineDB.mdb;");
    order = DriverManager.getConnection(sourceURL, "admin", "");
    myStatement = order.createStatement();
}
// The following exceptions must be caught
catch (ClassNotFoundException cnfe) {
    System.out.println(cnfe);
}
catch (SQLException sqle) {
    System.out.println(sqle);
}
setVisible(true);//Window is visible
setResizable(false);//Doesn't let users resize the window

//new class for the frame
public void changeFrameInit() {
    //JFrame changeFrame = new JFrame("Please Take Your Change");
    changeFrame.setSize(504, 390);//size of frame
    changeFrame.setLayout(null);//Null layout for absolute positioning
    changeFrame.setVisible(false);
    changeFrame.setResizable(false);
    changeFrame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);//disables the "X" button, so users dont close the window before inserting change

    //header label with imageicon
    cfLabelTakeChange.setBounds(22, 2, 450, 40);
    changeFrame.add(cfLabelTakeChange);

    CFpleaseTakeChange.setBounds(6, 50, 300, 20);
    changeFrame.add(CFpleaseTakeChange);

    //money
    //5p
    CFfivePenceButton.setBounds(10, 80, 68, 68);
    CFfivePenceButton.addActionListener(this);
    changeFrame.add(CFfivePenceButton);

    //10p
    CFTenPenceButton.setBounds(92, 80, 68, 68);
    CFTenPenceButton.addActionListener(this);
    changeFrame.add(CFTenPenceButton);

    //20p

CFtwentyPenceButton.setBounds(174,80,68,68);
CFtwentyPenceButton.addActionListener(this);
changeFrame.add(CFtwentyPenceButton);

//50p
CFfiftyPenceButton.setBounds(256,80,68,68);
CFfiftyPenceButton.addActionListener(this);
changeFrame.add(CFfiftyPenceButton);

//£1
CFonePoundButton.setBounds(338,80,68,68);
CFonePoundButton.addActionListener(this);
changeFrame.add(CFonePoundButton);

//£2
CFtwoPoundButton.setBounds(420,80,68,68);
CFtwoPoundButton.addActionListener(this);
changeFrame.add(CFtwoPoundButton);

//coin stock CF
CFfivepStock.setBounds(35,150,18,20);
CFfivepStock.setEditable(false);
changeFrame.add(CFfivepStock);

CFTenpStock.setBounds(117,150,18,20);
CFTenpStock.setEditable(false);
changeFrame.add(CFTenpStock);

CFTwentypStock.setBounds(199,150,18,20);
CFTwentypStock.setEditable(false);
changeFrame.add(CFTwentypStock);

CFFiftypStock.setBounds(281,150,18,20);
CFFiftypStock.setEditable(false);
changeFrame.add(CFFiftypStock);

CFonepStock.setBounds(364,150,18,20);
CFonepStock.setEditable(false);
changeFrame.add(CFonepStock);

CFtwoPStock.setBounds(445,150,18,20);
CFtwoPStock.setEditable(false);
changeFrame.add(CFtwoPStock);

//horizontal rule
CFruleH1.setBounds(4,180,490,2);
changeFrame.add(CFruleH1);

//changeremaining
CFchangeRemainingLabel.setBounds(6,190,120,20);
changeFrame.add(CFchangeRemainingLabel);

CFchangeRemaining.setBounds(125,190,60,20);
CFchangeRemaining.setEditable(false);
changeFrame.add(CFchangeRemaining);
private void closeChangeFrame()
{
    changeFrame.setVisible(false);
}

public void actionPerformed(ActionEvent e)
{
    //event handling code here
    //VendingMachine.refresh(timer);
    //showInputDialog(null, "Please select which type of file writer you want.\nType 'DB' for Database writer\nOR\nType 'FILE' for a Text File writer.\n");
    //formatting for prices
    DecimalFormat poundsFormat = new DecimalFormat("£0.00");
    //format for numbers (used for stock levels)
    DecimalFormat numFormat = new DecimalFormat("0");
    //format for date & time
    DecimalFormat twoFormat = new DecimalFormat("00");
    DecimalFormat decFormat = new DecimalFormat("0.00");

    String addToSelection; //adds the selected product to the output section

    //date:
    currentDate.setText(twoFormat.format(day) + "/" + twoFormat.format(month) + "/" + year);
    //time:
    currentTime.setText(twoFormat.format(hours) + ":" + twoFormat.format(minutes) + ":" + twoFormat.format(seconds));

    double errorGiveChange = totalInserted - productTotal;

    //change Frame actions:
    //product1
    if (e.getSource() == product1)
    {
        //adds product to the database
        writeString =
"INSERT INTO VendingMachine(Products, Price) VALUES('Mars Bar', '0.50');

//adds the text string the the selected items text field
addToSelection = selectedItems.getText() + "\nMars Bar - £0.50"
selectedItems.setText(addToSelection);

//adds 50p to the product total selected area
productTotal += 0.5;
selectedItemTotal.setText(poundsFormat.format(productTotal));

//updates the stock count by deducting 1
product1stockNum -= 1;
product1stock.setText(numFormat.format(product1stockNum));

//if the product stock is 0, disabled the product and brings up an error message
if (product1stockNum == 0)
{
    product1.setEnabled(false);
    showMessageDialog(null, "Sorry, Mars Bar's are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//product2
if (e.getSource() == product2)
{
    //adds product to the database
    writeString = 
"INSERT INTO VendingMachine(Products, Price) VALUES('Bounty Bar', '0.50');

    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + "\nBounty Bar Original - £0.50"
    selectedItems.setText(addToSelection);

    //adds 50p to the product total selected area
    productTotal += 0.5;
    selectedItemTotal.setText(poundsFormat.format(productTotal));

    //updates the stock count by deducting 1
    product2stockNum -= 1;
    product2stock.setText(numFormat.format(product2stockNum));

    //if the product stock is 0, disabled the product and brings up an error message
    if (product2stockNum == 0)
    {
        product2.setEnabled(false);
        showMessageDialog(null, "Sorry, Bounty Bar's are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
    }
}

//product3
if (e.getSource() == product3)
{
//adds product to the database
writeString =
"INSERT INTO VendingMachine(Products, Price) VALUES('Twix Double Bar', '£0.65');

//adds the text string the the selected items text field
addToSelection = selectedItems.getText() + "\nTwix Double Bar - £0.65"
selectedItems.setText(addToSelection);

//adds 65p to the product total selected area
productTotal += 0.65;
selectedItemTotal.setText(poundsFormat.format(productTotal));

//updates the stock count by deducting 1
product3stockNum -= 1;
product3stock.setText(numFormat.format(product3stockNum));

//if the product stock is 0, disabled the product and brings up an error message
if (product3stockNum == 0)
{
    product3.setEnabled(false);
    showMessageDialog(null, "Sorry, Twix Bar's are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//product4
if (e.getSource() == product4)
{
    //adds product to the database
writeString =
"INSERT INTO VendingMachine(Products, Price) VALUES('Walker's Ready Salted Crisps', '£0.55');

//adds the text string the the selected items text field
addToSelection = selectedItems.getText() + "\nWalker's Ready Salted Crisps - £0.55"
selectedItems.setText(addToSelection);

//adds 55p to the product total selected area
productTotal += 0.55;
selectedItemTotal.setText(poundsFormat.format(productTotal));

//updates the stock count by deducting 1
product4stockNum -= 1;
product4stock.setText(numFormat.format(product4stockNum));

//if the product stock is 0, disabled the product and brings up an error message
if (product4stockNum == 0)
{
    product4.setEnabled(false);
    showMessageDialog(null, "Sorry, Walkers Ready Salted Crisp's are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}
}
//product5
if (e.getSource() == product5) {
    //adds product to the database
    writeString = "INSERT INTO VendingMachine(Products, Price) VALUES('Monster Munch - Pickled Onion', '£0.70');"
    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + " Monster Munch - Pickled Onion - £0.70";
    selectedItems.setText(addToSelection);
    //adds 70p to the product total selected area
    productTotal += 0.7;
    selectedItemTotal.setText(poundsFormat.format(productTotal));
    //updates the stock count by deducting 1
    product5stockNum -= 1;
    product5stock.setText(numFormat.format(product5stockNum));
    //if the product stock is 0, disabled the product and brings up an error message
    if (product5stockNum == 0)
    {
        product5.setEnabled(false);
        showMessageDialog(null, "Sorry, Monster Munch Crisp's are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
    }
}

//product6
if (e.getSource() == product6) {
    //adds product to the database
    writeString = "INSERT INTO VendingMachine(Products, Price) VALUES('Walkers Sensations - Thai Sweet Chilli', '£1.10');"
    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + " Walkers Sensations - Thai Sweet Chilli - £1.10";
    selectedItems.setText(addToSelection);
    //adds 1.10 to the product total selected area
    productTotal += 1.1;
    selectedItemTotal.setText(poundsFormat.format(productTotal));
    //updates the stock count by deducting 1
    product6stockNum -= 1;
    product6stock.setText(numFormat.format(product6stockNum));
    //if the product stock is 0, disabled the product and brings up an error message
    if (product6stockNum == 0)
product6.setEnabled(false);
showMessageDialog(null, "Sorry, Walkers Sensation's are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//product7
if (e.getSource() == product7)
{
    //adds product to the database
    writeString =
        "INSERT INTO VendingMachine(Products, Price) VALUES('Coca-Cola (330ml)', '£0.75');"

    showMessageDialog(null, "Please exercise caution when opening this drink.\n\nIt may be shaken.", "Caution!", JOptionPane.WARNING_MESSAGE);
    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + "\nCoca-Cola (330ml) - £0.75"
    selectedItems.setText(addToSelection);

    //adds 75p to the product total selected area
    productTotal += 0.75;
    selectedItemTotal.setText(poundsFormat.format(productTotal));

    //updates the stock count by deducting 1
    product7stockNum -= 1;
    product7stock.setText(numFormat.format(product7stockNum));

    //if the product stock is 0, disabled the product and brings up an error message
    if (product7stockNum == 0)
    {
        product7.setEnabled(false);
        showMessageDialog(null, "Sorry, Coca-Cola is currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
    }
}

//product8
if (e.getSource() == product8)
{
    //adds product to the database
    writeString =
        "INSERT INTO VendingMachine(Products, Price) VALUES('Dr Pepper (330ml)', '£0.75');"

    showMessageDialog(null, "Please exercise caution when opening this drink.\n\nIt may be shaken.", "Caution!", JOptionPane.WARNING_MESSAGE);
    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + "\nDr Pepper (330ml) - £0.75"
    selectedItems.setText(addToSelection);

    //adds 75p to the product total selected area
    productTotal += 0.75;
selectedItemTotal.setText(poundsFormat.format(productTotal));

//updates the stock count by deducting 1
product8stockNum -= 1;
product8stock.setText(numFormat.format(product8stockNum));

//if the product stock is 0, disabled the product and brings up an error message
if (product8stockNum == 0)
{
    product8.setEnabled(false);
    showMessageDialog(null, "Sorry, Dr Pepper is currently out of stock",
        "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//product9
if (e.getSource() == product9)
{
    //adds product to the database
    writeString = 
        "INSERT INTO VendingMachine(Products, Price) VALUES('Fruit Pastels', '£0.60');"
    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + " \n Fruit Pastels - £0.60";
    selectedItems.setText(addToSelection);
    //adds 60p to the product total selected area
    productTotal += 0.6;
    selectedItemTotal.setText(poundsFormat.format(productTotal));

    //updates the stock count by deducting 1
    product9stockNum -= 1;
    product9stock.setText(numFormat.format(product9stockNum));

    //if the product stock is 0, disabled the product and brings up an error message
    if (product9stockNum == 0)
    {
        product9.setEnabled(false);
        showMessageDialog(null, "Sorry, Fruit Pastels are currently out of stock",
            "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
    }
}

//product10
if (e.getSource() == product10)
{
    //adds product to the database
    writeString = 
        "INSERT INTO VendingMachine(Products, Price) VALUES('Wine Gums', '£0.65');"
    //adds the text string the the selected items text field
    addToSelection = selectedItems.getText() + " \n Wine Gums - £0.65";
    selectedItems.setText(addToSelection);
//adds 65p to the product total selected area
productTotal += 0.65;
selectedItemTotal.setText(poundsFormat.format(productTotal));

//updates the stock count by deducting 1
product10stockNum -= 1;
product10stock.setText(numFormat.format(product10stockNum));

//if the product stock is 0, disabled the product and brings up an error message
if (product10stockNum == 0)
{
    product10.setEnabled(false);
    showMessageDialog(null, "Sorry, Wine Gums are currently out of stock", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//cancel order
if (e.getSource() == cancelOrder)
{
    selectedItems.setText("");
    moneyInput.setText(poundsFormat.format(productTotal - productTotal));
    selectedItemTotal.setText("£0.00");
    outputMessage.setText("");
    outputMessage.setBackground(Color.white);
    takeChange.setVisible(false);
    totalInserted = 0;
    productTotal = 0;
}

//enter
if (e.getSource() == enterCoins)//if enter coins button is pressed:
{
    String productsFW = selectedItems.getText();
    String totalFW = selectedItemTotal.getText();
    String inputFW = moneyInput.getText();
    dh.write(productsFW, totalFW, inputFW);
    double changePH = errorGiveChange;
    CFchangeRemaining.setText(decFormat.format(changePH));
    //first check if anything has been input, first money, then products
    //if no money is input at all
    if (totalInserted == 0)
    {
        outputMessage.setText("Error! Please input your money using the \n'Input Money' Section!\nYou need to input " + poundsFormat.format(productTotal) + " !");
        outputMessage.setBackground(Color.red);
    }
    //if no product is input
    else if (productTotal == 0)
    {

outputMessage.setText("Error! Please select at least one product!");
outputMessage.setBackground(Color.red);
}

else if (productTotal == totalInserted)//Correct money is input
{
    outputMessage.setText(poundsFormat.format(totalInserted) + " Accepted!
\n\nThank you for your purchase!");
    outputMessage.setBackground(Color.green);

    //clear all other areas once accepted:
    selectedItem.setText(" ");
    selectedItemTotal.setText("£0.00");
    moneyInput.setText("£0.00");

    //reset values to 0
    totalInserted = 0;
    productTotal = 0;
}
else if (totalInserted < productTotal)//Not enough money is input
{
    //this works out the rest of the input needed:
    double errorInputShort = productTotal - totalInserted;

    //this sum is then applied with pounds format to the output message to tell
the user:
    outputMessage.setText("Error! Please input the correct amount!
\n\nYou still have " + poundsFormat.format(errorInputShort) + " to pay!");
    outputMessage.setBackground(Color.red);
}
else if (totalInserted > productTotal)//if too much money is input
{
    //make the "take change button visible
    takeChange.setVisible(true);

    //disable other buttons, gives focus and is professional and eliminates error
product1.setEnabled(false);
product2.setEnabled(false);
product3.setEnabled(false);
product4.setEnabled(false);
product5.setEnabled(false);
product6.setEnabled(false);
product7.setEnabled(false);
product8.setEnabled(false);
product9.setEnabled(false);
product10.setEnabled(false);
enterCoins.setEnabled(false);
returnCoins.setEnabled(false);
cancelOrder.setEnabled(false);
fivePenceButton.setEnabled(false);
tenPenceButton.setEnabled(false);
twentyPenceButton.setEnabled(false);
fiftyPenceButton.setEnabled(false);
onePoundButton.setEnabled(false);
twoPoundButton.setEnabled(false);
this is then added to the output message (with pounds format)
outputMessage.setText(poundsFormat.format(totalInserted) + " Accepted!
Thank you for your purchase!
Please take your " + poundsFormat.format(errorGiveChange) + " change.
Press the 'Take Change' button below:"
outputMessage.setBackground(Color.green);

//clear textfields
selectedItems.setText(""");
selectedItemTotal.setText("£0.00");
moneyInput.setText("£0.00");

//reset values
totalInserted = 0;
productTotal = 0;

//take change button
if (e.getSource() == takeChange)
{
    //removes the button as it is clicked, so it cannot be shown again till an order has been made
    takeChange.setVisible(false);
    changeFrameInit();
    changeFrame.setVisible(true);

    //sets the money stock into that of the change frame
    CFfivepStock.setText(numFormat.format(fivepStockNum));
    CftenpStock.setText(numFormat.format(tenpStockNum));
    CFtwentypStock.setText(numFormat.format(twentypStockNum));
    CFfiftypStock.setText(numFormat.format(fiftypStockNum));
    CFonepStock.setText(numFormat.format(onepStockNum));
    CftwopStock.setText(numFormat.format(twopStockNum));
}

//CF 5p
if (e.getSource() == CFfivePenceButton)
{
    //deducts one count from the change stock
    fivepStockNum -=1;
    CFfivepStock.setText(numFormat.format(fivepStockNum));

    //deducts amount from the change remaining
    double calcChange = Double.parseDouble(CFchangeRemaining.getText());
    calcChange -=0.05;
    CFchangeRemaining.setText(decFormat.format(calcChange));

    //if user tries to take too much change
    if (calcChange < 0)
    {
        calcChange +=0.05; //adds the 1pound back to the value
        CFchangeRemaining.setText(decFormat.format(calcChange));

        //reset the stock value (as if it was never changed)
        fivepStockNum ++1;
        CFfivepStock.setText(numFormat.format(fivepStockNum));

//set error message
CFOutputMessage.setText("                                      Error!
You cannot take that much!
You only have £" + decFormat.format(calcChange) + " in change left to take!");
CFOutputMessage.setBackground(Color.red);
}

if (fivepStockNum == 0)
{
    CFFivePenceButton.setEnabled(false);
    showMessageDialog(null, "Sorry, there are no 5 pence coins left.
Please select other coins.", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//CF 10p
if (e.getSource() == CFTenPenceButton)
{
    //deducts one count from the change stock
tenpStockNum -=1;
    CFtenPenceStock.setText(numFormat.format(tenpStockNum));

    //deducts amount from the change remaining
double calcChange = Double.parseDouble(CFchangeRemaining.getText());
calcChange -=0.1;
    CFchangeRemaining.setText(decFormat.format(calcChange));

    //if user tries to take too much change
if (calcChange < 0)
{
    calcChange +=0.1; //adds the 1pound back to the value
    CFchangeRemaining.setText(decFormat.format(calcChange));

    //reset the stock value (as if it was never changed)
tenpStockNum +=1;
    CFtenPenceStock.setText(numFormat.format(tenpStockNum));

    //set error message
    CFOutputMessage.setText("                                      Error!
You cannot take that much!
You only have £" + decFormat.format(calcChange) + " in change left to take!");
    CFOutputMessage.setBackground(Color.red);
}

if (tenpStockNum == 0)
{
    CFTenPenceButton.setEnabled(false);
    showMessageDialog(null, "Sorry, there are no 10 pence coins left.
Please select other coins.", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//CF 20p
if (e.getSource() == CFTwentyPenceButton)
{
    //deducts one count from the change stock
twentypStockNum -=1;
CFtwentypStock.setText(numFormat.format(twentypStockNum));

//deducts amount from change remaining
double calcChange = Double.parseDouble(CFchangeRemaining.getText());
calcChange -=0.2;
CFchangeRemaining.setText(decFormat.format(calcChange));

//if user tries to take too much change
if (calcChange < 0)
{
    calcChange +=0.2; //adds the 1pound back to the value
    CFchangeRemaining.setText(decFormat.format(calcChange));

    //reset the stock value (as if it was never changed)
twentypStockNum +=1;
    CFtwentypStock.setText(numFormat.format(twentypStockNum));

    //set error message
    CFoutputMessage.setText("                                      Error!\nYou cannot take that much!\nYou only have £" + decFormat.format(calcChange) + " in change left to take!");
    CFoutputMessage.setBackground(Color.red);
}

if (twentypStockNum == 0)
{
    CFtwentyPenceButton.setEnabled(false);
    showMessageDialog(null, "Sorry, there are no 20 pence coins left.\nPlease select other coins.", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//CF 50p
if (e.getSource() == CFfiftyPenceButton)
{
    //deducts one count from the change stock
    fiftypStockNum -=1;
    CFfiftypStock.setText(numFormat.format(fiftypStockNum));

    //deducts amount from change remaining
    double calcChange = Double.parseDouble(CFchangeRemaining.getText());
    calcChange -=0.5;
    CFchangeRemaining.setText(decFormat.format(calcChange));

    //if user tries to take too much change
    if (calcChange < 0)
    {
        calcChange +=0.5; //adds the 1pound back to the value
        CFchangeRemaining.setText(decFormat.format(calcChange));

        //reset the stock value (as if it was never changed)
        fiftypStockNum +=1;
        CFfiftypStock.setText(numFormat.format(fiftypStockNum));

        //set error message
        CFoutputMessage.setText("                                      Error!\nYou cannot take that much!\nYou only have £" + decFormat.format(calcChange) + " in change left to take!");
    }
CFoutputMessage.setBackground(Color.red);

if (fiftypStockNum == 0)
{
    CFfiftyPenceButton.setEnabled(false);
    showMessageDialog(null, "Sorry, there are no 50 pence coins left.\nPlease select other coins.", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//CF 1pound
if (e.getSource() == CFonePoundButton)
{
    //deducts one count from the change stock
    onepStockNum -= 1;
    CFonepStock.setText(numFormat.format(onepStockNum));

    //deducts amount from change remaining
    double calcChange = Double.parseDouble(CFchangeRemaining.getText());
    calcChange -= 1;
    CFchangeRemaining.setText(decFormat.format(calcChange));

    //if user tries to take too much change
    if (calcChange < 0)
    {
        calcChange += 1; //adds the 1pound back to the value
        CFchangeRemaining.setText(decFormat.format(calcChange));

        //reset the stock value (as if it was never changed)
        onepStockNum += 1;
        CFonepStock.setText(numFormat.format(onepStockNum));

        //set error message
        CFoutputMessage.setText("\nError!\nYou cannot take that much!\nYou only have £" + decFormat.format(calcChange) + " in change left to take!");
        CFoutputMessage.setBackground(Color.red);
    }

    if (onepStockNum == 0)
    {
        CFonePoundButton.setEnabled(false);
        showMessageDialog(null, "Sorry, there are no 1 pound coins left.\nPlease select other coins.", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
    }
}

//CF 2pound
if (e.getSource() == CFtwoPoundButton)
{
    //deducts one count from the change stock
    twopStockNum -= 1;
    CFTwopStock.setText(numFormat.format(twopStockNum));

    //deducts amount from change remaining
    double calcChange = Double.parseDouble(CFchangeRemaining.getText());

calcChange -=2;
CFchangeRemaining.setText(decFormat.format(calcChange));

//if user tries to take too much change
if (calcChange < 0)
{
    calcChange +=2; //adds the 1pound back to the value
    CFchangeRemaining.setText(decFormat.format(calcChange));
    //reset the stock value (as if it was never changed)
    twopStockNum +=1;
    CFtwopStock.setText(numFormat.format(twopStockNum));

    //set error message
    CFoutputMessage.setText("Error!\nYou cannot take that much!\nYou only have £" + decFormat.format(calcChange) + " in change left to take!");
    CFoutputMessage.setBackground(Color.red);
}

if (twopStockNum == 0)
{
    CFtwoPoundButton.setEnabled(false);
    showMessageDialog(null, "Sorry, there are no 2 pound coins left.\nPlease select other coins.", "Stock Depleted!", JOptionPane.WARNING_MESSAGE);
}

//CF finished button
if (e.getSource() == CFfinished)
{
    //gets value of change remaining
    double calcChange = Double.parseDouble(CFchangeRemaining.getText());

    if (calcChange > 0)
    {
        CFoutputMessage.setText("Error!\nYou still have £" + decFormat.format(calcChange) + " in change to take!");
        CFoutputMessage.setBackground(Color.red);
    }

    if (calcChange == 0)
    {
        CFoutputMessage.setText("Thank You!\nPlease press the button below to close this window.");
        CFoutputMessage.setBackground(Color.green);

        //enables the close button visible
        CFclose.setVisible(true);

        //disables all other buttons on the frame, adds focus, gives user indication of what to do next
        CFfivePenceButton.setEnabled(false);
        CftenPenceButton.setEnabled(false);
        CFtwentyPenceButton.setEnabled(false);
        CFfiftyPenceButton.setEnabled(false);
        CFonePoundButton.setEnabled(false);

    }
CFtwoPoundButton.setEnabled(false);
CFfinished.setEnabled(false);
}

//change frame close button
if (e.getSource() == CFclose)
{
    //calls function close change frame
    closeChangeFrame();

    //re-enable all buttons on main frame
    product1.setEnabled(true);
    product2.setEnabled(true);
    product3.setEnabled(true);
    product4.setEnabled(true);
    product5.setEnabled(true);
    product6.setEnabled(true);
    product7.setEnabled(true);
    product8.setEnabled(true);
    product9.setEnabled(true);
    product10.setEnabled(true);
    enterCoins.setEnabled(true);
    returnCoins.setEnabled(true);
    cancelOrder.setEnabled(true);
    fivePenceButton.setEnabled(true);
    tenPenceButton.setEnabled(true);
    twentyPenceButton.setEnabled(true);
    fiftyPenceButton.setEnabled(true);
    onePoundButton.setEnabled(true);
    twoPoundButton.setEnabled(true);
    CFfivePenceButton.setEnabled(true);
    CftenPenceButton.setEnabled(true);
    CtwopenceButton.setEnabled(true);
    CFfiftyPenceButton.setEnabled(true);
    CFonePoundButton.setEnabled(true);
    CFtwoPoundButton.setEnabled(true);
    CFfinished.setEnabled(true);
    //reset the output message on main frame
    outputMessage.setText("\n\nPlease try again!");
    outputMessage.setBackground(Color.red);

    //return
    if (e.getSource() == returnCoins)
    {
        if (totalInserted == 0)//if no money has been input
        {
            outputMessage.setText("You have not input any money!
\nPlease try again!");
            outputMessage.setBackground(Color.red);

            //clear money input
            moneyInput.setText("£0.00");
            //no need to reset value to zero as nothing has been put in
else/\text{if} \text{ something HAS been input}
{
  \text{outputMessage}.setText(poundsFormat.format(totalInserted) + \text{" Returned. \n
Please take your money."});
  \text{outputMessage}.setBackground(Color.red);

  \text{//clear money input}
  \text{moneyInput}.setText("£0.00");

  \text{//reset money input value}
  \text{totalInserted} = 0;
}

\text{//moneyInput Section}

\text{//5 pence}
\text{if(e.getSource() == fivePenceButton)}
{
  \text{//updates the total input value}
  \text{totalInserted} += 0.05;
  \text{moneyInput}.setText(poundsFormat.format(totalInserted));

  \text{//updates the stock count by deducting 1}
  \text{fivepStockNum} += 1;
  \text{fivepStock}.setText(numFormat.format(fivepStockNum));
}

\text{//10 pence}
\text{if(e.getSource() == tenPenceButton)}
{
  \text{//updates the total input value}
  \text{totalInserted} += 0.1;
  \text{moneyInput}.setText(poundsFormat.format(totalInserted));

  \text{//updates the stock count by deducting 1}
  \text{tenpStockNum} += 1;
  \text{tenpStock}.setText(numFormat.format(tenpStockNum));
}

\text{//20 pence}
\text{if(e.getSource() == twentyPenceButton)}
{
  \text{//updates the total input value}
  \text{totalInserted} += 0.2;
  \text{moneyInput}.setText(poundsFormat.format(totalInserted));

  \text{twentypStockNum} += 1;
  \text{twentypStock}.setText(numFormat.format(twentypStockNum));
}

\text{//50 pence}
\text{if(e.getSource() == fiftyPenceButton)}
{
  \text{//updates the total input value}
  \text{totalInserted} += 0.5;
moneyInput.setText(poundsFormat.format(totalInserted));

//updates the stock count by deducting 1
fiftypStockNum += 1;
fiftypStock.setText(numFormat.format(fiftypStockNum));
}

// 1 pound
if(e.getSource() == onePoundButton)
{
    //updates the total input value
totalInserted += 1;
moneyInput.setText(poundsFormat.format(totalInserted));

    // updates the stock count by deducting 1
    onepStockNum += 1;
onepStock.setText(numFormat.format(onepStockNum));
}

// 2 pounds
if(e.getSource() == twoPoundButton)
{
    // updates the total input value
    totalInserted += 2;
moneyInput.setText(poundsFormat.format(totalInserted));

    // updates the stock count by adding 1
    twopStockNum += 1;
twopStock.setText(numFormat.format(twopStockNum));
}

try {
    myStatement.executeUpdate(writeString);
} catch (SQLException sqle) {
    System.out.println(sqle);
}
**FileHandler.java**

```java
import java.io.*;
import java.util.Date;

class FileHandler implements DataHandler {

    public boolean write(String productsFW, String totalFW, String inputFW) {
        try {
            PrintWriter outputFile =
                new PrintWriter(new FileWriter("VendingMachineData.txt", true));
            outputFile.println("-------------------------------
Time & Date of Purchase: " + new Date() + "
Product(s): " + productsFW + "
Total to be inserted: " + totalFW + "
Amount Input: " + inputFW + "
-------------------------------");
            outputFile.close();
        } catch (IOException ioe) {
            System.out.println("File VendingMachineData.txt not found");
            return false;
        }
        return true; // inserted OK
    }

    public void displayUsers(PrintStream outS) {
        try {
            BufferedReader inFile =
                new BufferedReader(new FileReader("VendingMachineData.txt"));
            String line;
            while ((line = inFile.readLine()) != null)
                outS.println(line);
            inFile.close();
        } catch (IOException ioe) {
            System.out.println(ioe);
        }
    }
}

**DataHandler.java**

```java
public interface DataHandler {
    public boolean write(String productsFW, String totalFW, String inputFW);
    public void displayUsers(java.io.PrintStream out);
}
VendingMachineDBHandler.java

import java.io.*;
import java.sql.*;
import java.util.Date;

class VendingMachineDBHandler {

    Connection order;
    Statement myStatement;

    public VendingMachineDBHandler() {
        try {
            Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
            String sourceURL = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=VendingMachineDB.mdb;";
            order = DriverManager.getConnection(sourceURL, "admin", "");
            myStatement = order.createStatement();
        } // The following exceptions must be caught
        catch (ClassNotFoundException cnfe) {
            System.out.println(cnfe);
        } catch (SQLException sqle) {
            System.out.println(sqle);
        }
    }

    public void madeChoice(String choice) {
        String writeString = "INSERT INTO VendingMachine(Date/Time, Products, Price) VALUES('" + new Date() + ", " + choice + ", " + choice + ")";
        try {
            myStatement.executeUpdate(writeString);
        } catch (SQLException sqle) {
            System.out.println(sqle);
        }
    }
}
