

What I did

For the senior project I decided to create a program that would allow people who made tips delivering food (specifically pizza in this situation, specifically at the Dominoes that I work for located in New Tampa) to track those tips as well as how much money would be expected at the end of the night.

The program is comprised of three GUI forms for the user to interact with each one performing a specific function, adding new runs, displaying an overview of runs, or displaying data pertaining to runs.

What I want to do

The program is nowhere near perfect and I am completely ok with that. Some things that I would want to add (and plan to add) to the program include the ability to track how many miles the user has driven using predetermined data and using that information calculate average mileage amount per mile gone.

I want to also add in the ability to look at averages for the shift to show things like average order amount or average amount the user received. I also want to look at the average tip amount the user receives and calculate about how much per hour the user is actually making when including the tips.

Another thing I want to add to the program is the ability to look up gate codes for various subdivision, and to use street names to locate directions, about

how long the delivery should take, and again, to determine how many miles it should be.

Why I did it

Working with Dominoes I have discovered just how annoying it is to keep track of tips and how inefficient it is to try to figure out tip amounts all in one's head. I also asked my manager at the store what she thought the drivers really needed. She said they needed something to help them keep track of tips and mileage and if possible, a way to locate houses without using gps but rather the driver's own experience.

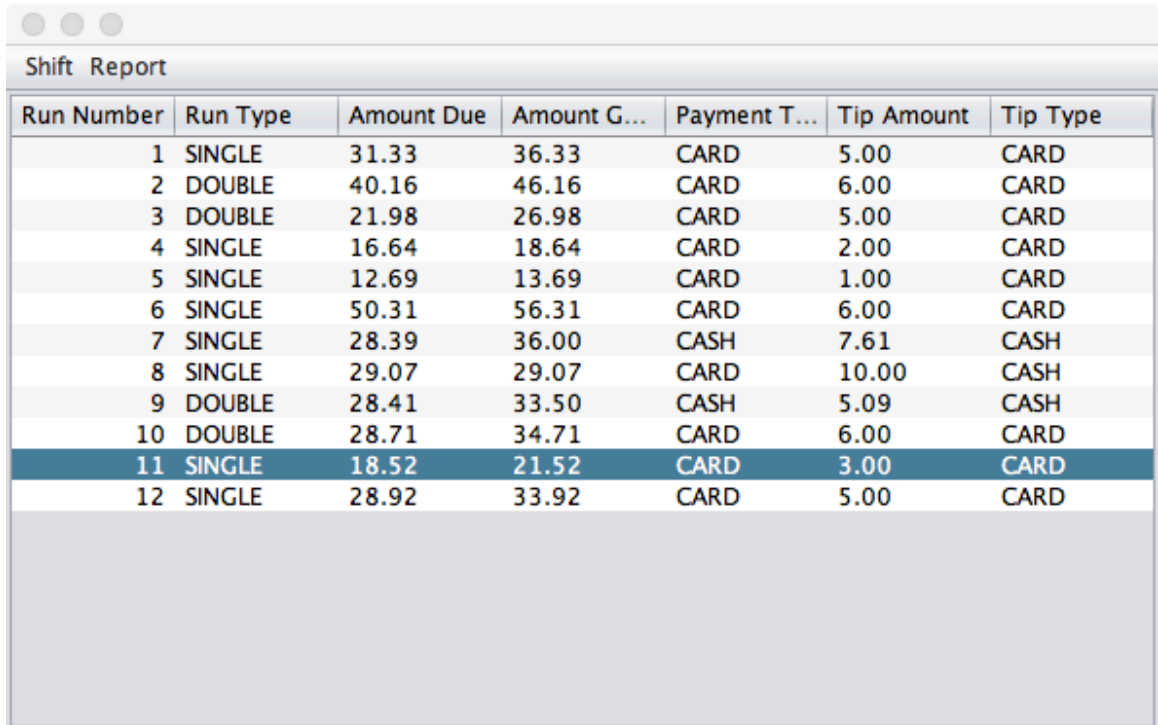
When I asked my fellow drivers what they would want to see in a tip tracking app they said 'something that actually worked and was easy to use'. I am still striving to create that but I think for a first step I have done well.

What tools I used

For the project I used NetBeans IDE 7.3.1 which is my 'go-to' program for when I am messing around with coding in Java and C++. I have had lots of issues learning to program in C++ so decided to work on that programming language after I had established the program in Java and had it working in that programming language.

How it works

Below I have an already populated screenshot of the program from when I used it last night.



Run Number	Run Type	Amount Due	Amount G...	Payment T...	Tip Amount	Tip Type
1	SINGLE	31.33	36.33	CARD	5.00	CARD
2	DOUBLE	40.16	46.16	CARD	6.00	CARD
3	DOUBLE	21.98	26.98	CARD	5.00	CARD
4	SINGLE	16.64	18.64	CARD	2.00	CARD
5	SINGLE	12.69	13.69	CARD	1.00	CARD
6	SINGLE	50.31	56.31	CARD	6.00	CARD
7	SINGLE	28.39	36.00	CASH	7.61	CASH
8	SINGLE	29.07	29.07	CARD	10.00	CASH
9	DOUBLE	28.41	33.50	CASH	5.09	CASH
10	DOUBLE	28.71	34.71	CARD	6.00	CARD
11	SINGLE	18.52	21.52	CARD	3.00	CARD
12	SINGLE	28.92	33.92	CARD	5.00	CARD

This image shows the layout of the main screen where the user is given an overview of how the night had gone. As can be seen from this image, the app keeps track of what number run the user is on, what type of run it is, the amount due, the amount gained, how the amount was paid, how much the tip was, and how the tip was paid.

```
123  /*
124  private void jMenuItem2ActionPerformed(java.awt.event.ActionEvent evt) {
125      NewRunUIDialog dialog;
126      dialog = new NewRunUIDialog(this, false);
127      dialog.setVisible(true);
128  }
129  /*
130  * Adds new rows to the table for easy viewing.
131  * @param rNum The run number for the delivery being added to the table.
132  * @param deliveryMethod Specifies whether or not the run is a Single, Double, or Triple.
133  * @param aDue The amount due from the delivery.
134  * @param aGained The amount recieved from the delivery.
135  * @param paymentType Specifies whether the amount was paid in cash or on their card, or if it was prepaid.
136  * @param tipAmount The amount of tip that was recieved.
137  * @param tipType Specifies whether the tip was put on the card (if a Card Tip) or if it was paid in cash.
138  */
139  public static void jTable1Update(int rNum, Delivery.DeliveryMethod deliveryMethod, BigDecimal aDue, BigDecimal aGained, Delivery.PaymentMe
140      DefaultTableModel model;
141      model = (DefaultTableModel) jTable1.getModel();
142      model.addRow(new Object[]{rNum, deliveryMethod, aDue, aGained, paymentType, tipAmount, tipType});
143  }
144  /*
145  * Closes the program
146  */
147  private void jMenuItem4ActionPerformed(java.awt.event.ActionEvent evt) {
148      // TODO add your handling code here:
149      System.exit(0);
150  }
151  /*
152  * Brings up the dialog where the user can check on totals from their shift.
153  */
154  private void jMenuItem5ActionPerformed(java.awt.event.ActionEvent evt) {
155      ReportTotalsUI dialog;
156      dialog = new ReportTotalsUI(this, false);
157      dialog.setVisible(true);
158  }
159  }
160  /**...*/
```

Here, it is slightly harder to see, but I have some of the major methods from the GUI shown above. All positions and such was computer generated by NetBeans IDE and is stated as such in the appropriate javadocs. The methods shown here include, opening a dialog to allow entry of a new run, the method that adds a new row to the table, the method that closes the application, and the method that opens the report showing the totals.

	Cash	Card	Total
Due	56.80	278.33	335.13
Gained	79.50	317.33	396.83
Tips	22.70	39.00	61.70
	Mileage	Runs	Total Total
	14.4	12	56.10

Close

This is the report that shows the total for the night by the user. It is divided into Cash amounts, CC amounts, totals from both amounts, the total mileage, the number of runs, and how much the user is under/even/over. Each 'shift' starts with a bank of 20 dollars so if there are no runs and the report were to be opened it would show a balance of -20.00 under Total Total.

New Run

Input New Run

Single
 Double
 Triple

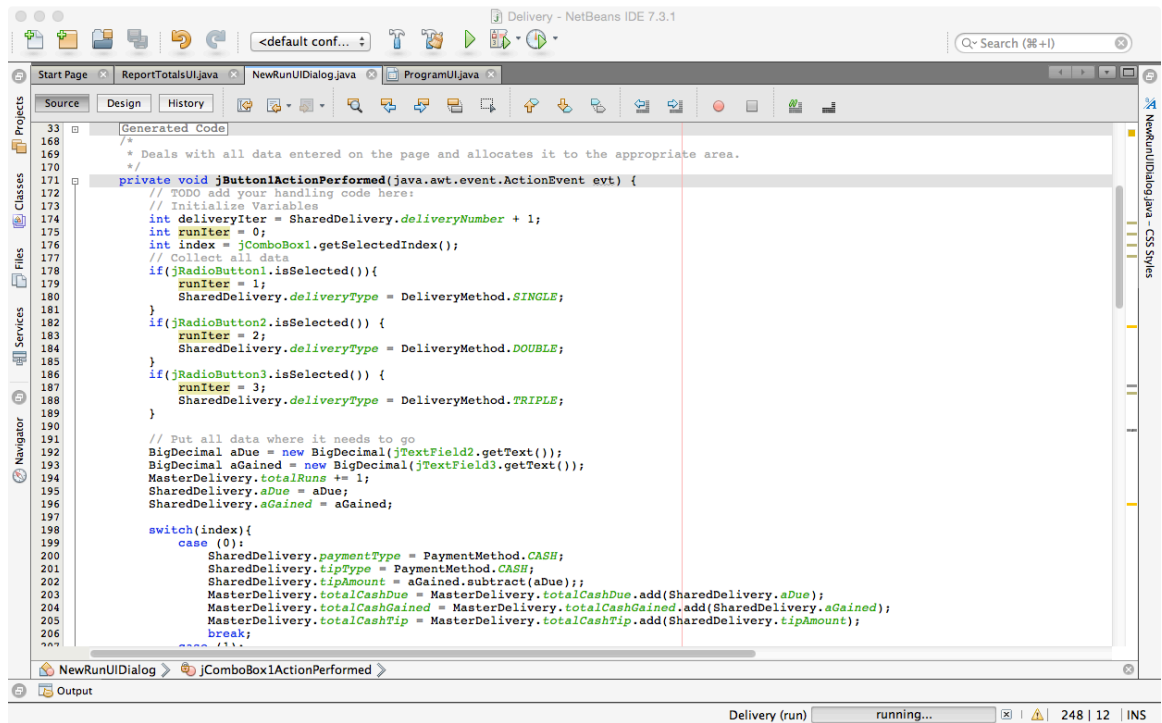
Payment Method: CASH

Amount Due:
 Amount Gained:

Cash Tip Amount:

Submit

This screenshot shows the dialog to input a new run, and the next couple of screenshots will show the coding behind it. In the future I will most likely be further cleaning up the code and removing things that I determine to be irrelevant or unnecessary.



```
33  Generated Code
168  /*
169  * Deals with all data entered on the page and allocates it to the appropriate area.
170  */
171  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
172      // TODO add your handling code here:
173      // Initialize Variables
174      int deliveryIter = SharedDelivery.deliveryNumber + 1;
175      int runIter = 0;
176      int index = JComboBox1.getSelectedIndex();
177      // Collect all data
178      if (jRadioButton1.isSelected()) {
179          runIter = 1;
180          SharedDelivery.deliveryType = DeliveryMethod.SINGLE;
181      }
182      if (jRadioButton2.isSelected()) {
183          runIter = 2;
184          SharedDelivery.deliveryType = DeliveryMethod.DOUBLE;
185      }
186      if (jRadioButton3.isSelected()) {
187          runIter = 3;
188          SharedDelivery.deliveryType = DeliveryMethod.TRIPLE;
189      }
190
191      // Put all data where it needs to go
192      BigDecimal aDue = new BigDecimal(jTextField2.getText());
193      BigDecimal aGained = new BigDecimal(jTextField3.getText());
194      MasterDelivery.totalRuns += 1;
195      SharedDelivery.aDue = aDue;
196      SharedDelivery.aGained = aGained;
197
198      switch (index) {
199          case 0:
200              SharedDelivery.paymentType = PaymentMethod.CASH;
201              SharedDelivery.tipType = PaymentMethod.CASH;
202              SharedDelivery.tipAmount = aGained.subtract(aDue);
203              MasterDelivery.totalCashDue = MasterDelivery.totalCashDue.add(SharedDelivery.aDue);
204              MasterDelivery.totalCashGained = MasterDelivery.totalCashGained.add(SharedDelivery.aGained);
205              MasterDelivery.totalCashTip = MasterDelivery.totalCashTip.add(SharedDelivery.tipAmount);
206              break;
207          case 1:
208              // ...
209      }
210  }
```

The first part of the code for when the 'Submit' button is clicked.

