Shankly Richard Cragg

shc27@aber.ac.uk

17/03/2015

Mrs Miggins PieShop

UML Diagram

Item
<<Java Class>>
Item
(default package)

- □ identifier: String
- □ name: String
- □ cost: int
- □ quanity: int

- Item(String)
- Item(String,String,int)
- Item(String,String,int,int)
- getIdentifier():String
- setIdentifier(String):void
- getName():String
- setName(String):void
- getCost():int
- setCost(int):void
- getQuanity():int
- setQuanity(int):void
- equals(Object):boolean
- toString():String

Shop
<<Java Class>>
Shop
(default package)

- □ shopName: String
- □ exist: boolean
- □ scan: Scanner
- □ totalCostDue: int
- □ newQuantity: int
- SHOP_STOCK_DATA_FILE: String
- SHOP_TILL_DATA_FILE: String

- Shop(String)
- stockShop():void
- startTill():void
- runTill():void
- getChange():void
- getBalance():void
- runMenu():void
- doContinue():boolean
- getInt(String):int
- displayCost(String,int):void
- getDenominationType():UKDenomination
- printMenu():void
- save():void
- saveTill():void
- saveStock():void
- load():void
- loadStock():void
- loadTill():void
- main(String[]):void

-items 0..*

-till 1..1

UKTill
<<Java Class>>
UKTill
(default package)

- UKTill()
- addFloat(DenominationFloat):void
- removeFloat(UKDenomination):void
- emptyTill():DenominationFloat[]
- getChange(int):DenominationFloat[]
- toString():String

-contents 0..*

DenominationFloat
<<Java Class>>
DenominationFloat
(default package)

- □ quantity: int

- DenominationFloat(UKDenomination)
- DenominationFloat(UKDenomination,int)
- getType():UKDenomination
- getQuantity():int
- setQuantity(int):void
- equals(Object):boolean
- toString():String

UKDenomination
<<Java Enumeration>>
UKDenomination
(default package)

- pence_1: UKDenomination
- pence_2: UKDenomination
- pence_5: UKDenomination
- pence_10: UKDenomination
- pence_20: UKDenomination
- pence_50: UKDenomination
- pound_1: UKDenomination
- pound_2: UKDenomination
- pound_5: UKDenomination
- pound_10: UKDenomination
- pound_20: UKDenomination
- pound_50: UKDenomination
- □ value: int

- UKDenomination(int)
- getValue():int
- fromString(String):UKDenomination
- toString():String

-name 0..1

Write-up

I began with the stockShop method. To ensure the shopkeeper could only enter a barcode of the correct form (4 digits), I used regular expressions and a do while loop so that the shopkeeper must enter a correctly formed barcode in order to proceed. I created a method to do this as I would want to use it more than once. This meant I wouldn't have to re-use the code. Since the name of an item can include letters or numbers, I just ensure that something is entered. The cost and quantity are gotten using the getInt() method already created.

The loading and saving went by without incident. The only thing worth noting is the use of a bufferedReader to check that the file is not empty.

In runTill(), there are a lot of checks to ensure no errors happen. First of all, I check if there is any stock in the shop at all, and if there is no stock I display a message to the user saying we are out of stock. If there is stock then I call the method I created earlier for stocking the shop to get the barcode of an item. I then loop through all the items in stock and compare the barcode to the one the customer entered. If no match is found an error message tells the user that the item either doesn't exist, is out of stock, or they entered the barcode incorrectly. If it does exist, then I display the cost of the item, add that to their total cost and decrement the quantity of the item by one. If the item has a quantity of 0, then it is considered "out of stock" and removed from the list of stock. If this happens and it was the last piece of stock, then I break out of the loop as there is nothing else to be done. The breaks prevent errors that the code was creating without impacting the use of the system in any way. These errors included a commodification error.

For the user paying for items, I then read in the denomination they want to pay in and the amount of it they want to use. For example using 4, 5p pieces. With this information I multiple the value of the currency by the amount and take it away from the cost they have to pay. This is then added to the till, and repeated as long as the remaining cost of the customer's items is 0 or more. If the customer pays too much then I need to give them some money back.

When I need to give the customer change, if it turns out I cannot give the correct change (most likely if they need an odd amount and there are no 1ps in the till), then I let the customer know that I owe them a certain amount of change.
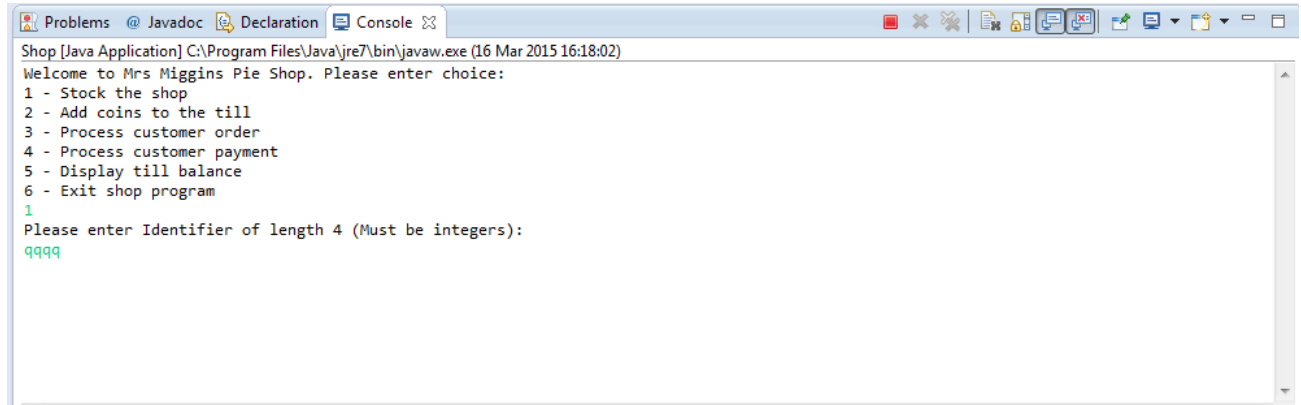
The most difficult part of the assignment was grasping the concept of denominations and how their values were stored and accessed. Once I understood this after playing with the code, it fell into place pretty quickly.

For this assignment I believe I deserve a 1st, as all the functionality is implemented, and the code has shown no large errors according to my testing.
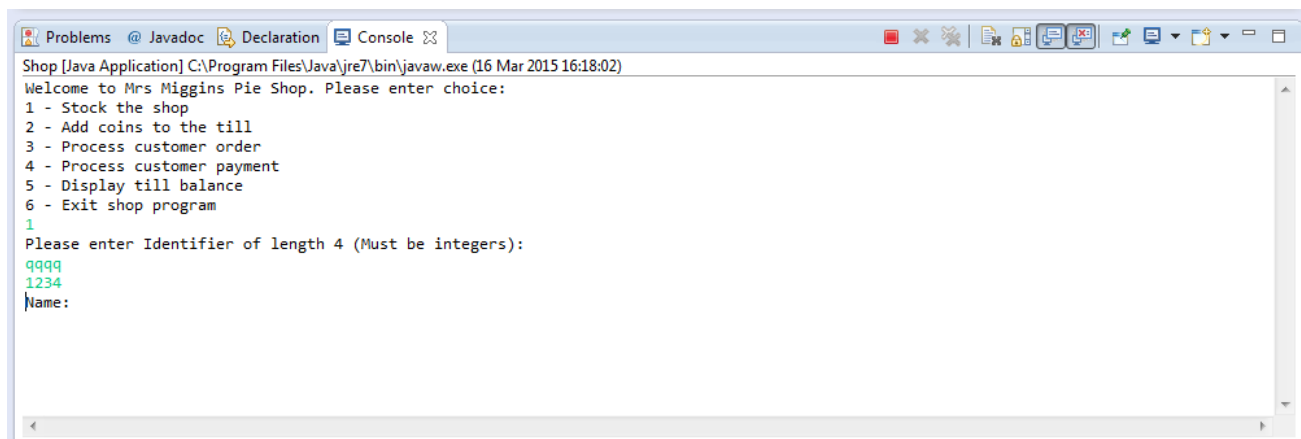
Testing

Adding an item to stock

Entering an incorrect barcode

```
Problems  @ Javadoc  Declaration  Console

Shop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (16 Mar 2015 16:18:02)
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
1
Please enter Identifier of length 4 (Must be integers):
qqqq
```
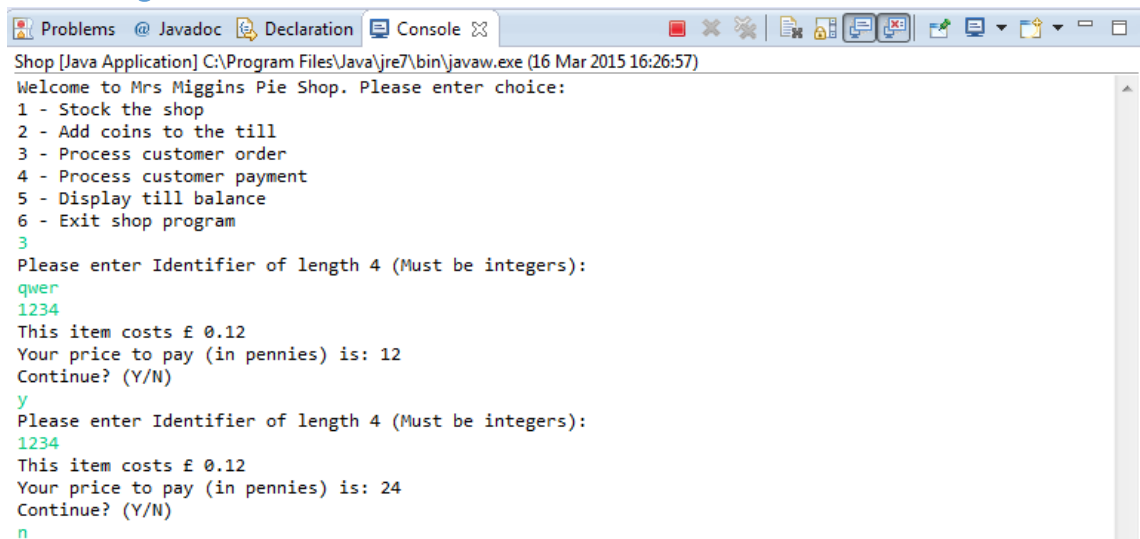
The code doesn't advance until a correct barcode is entered.

```
Problems  @ Javadoc  Declaration  Console

Shop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (16 Mar 2015 16:18:02)
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
1
Please enter Identifier of length 4 (Must be integers):
qqqq
1234
Name:
```

Now a correct barcode has been entered, it will ask for the next piece of information. This works exactly the same for "name", "cost" and "quantity".
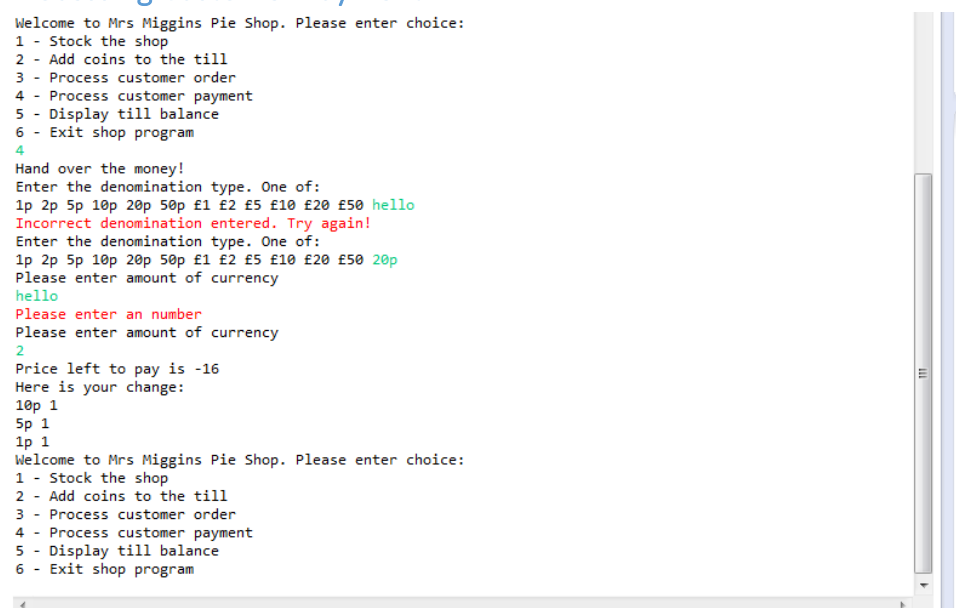
Processing customer order

```
Problems  @ Javadoc  Declaration  Console

Shop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (16 Mar 2015 16:26:57)
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
3
Please enter Identifier of length 4 (Must be integers):
qwer
1234
This item costs £ 0.12
Your price to pay (in pennies) is: 12
Continue? (Y/N)
y
Please enter Identifier of length 4 (Must be integers):
1234
This item costs £ 0.12
Your price to pay (in pennies) is: 24
Continue? (Y/N)
n
```
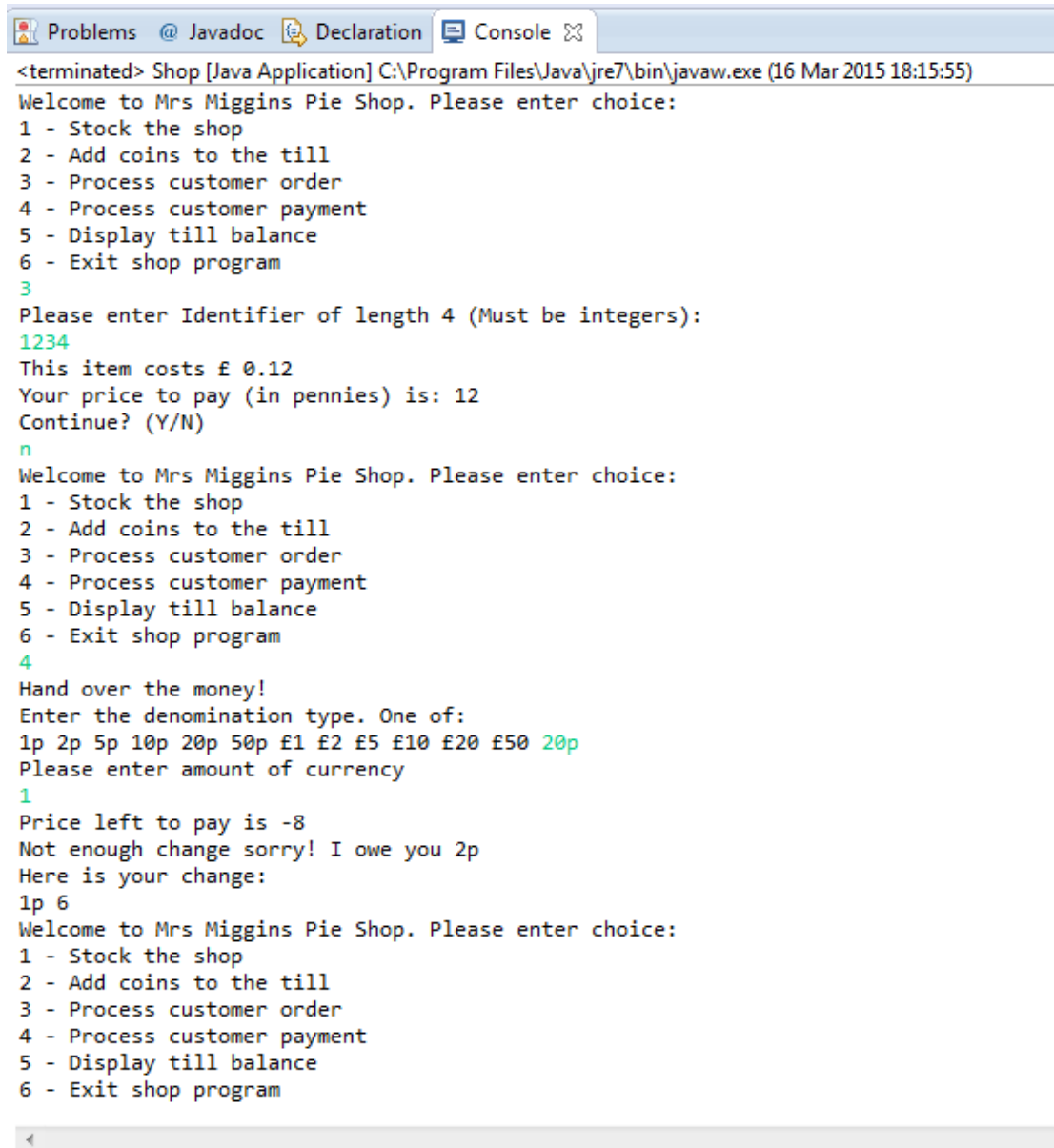
Once again, when entering the barcode it doesn't continue unless it is correct. You can see the cost of the item they have bought, and the total cost of all items printed.

Processing Customer Payment

```
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
4
Hand over the money!
Enter the denomination type. One of:
1p 2p 5p 10p 20p 50p £1 £2 £5 £10 £20 £50 hello
Incorrect denomination entered. Try again!
Enter the denomination type. One of:
1p 2p 5p 10p 20p 50p £1 £2 £5 £10 £20 £50 20p
Please enter amount of currency
hello
Please enter an number
Please enter amount of currency
2
Price left to pay is -16
Here is your change:
10p 1
5p 1
1p 1
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
```

Once asked to enter a denomination, if you enter an incorrect denomination it asks again until you get it right. When entering the amount of the denomination using, it also errors and asks for an integer if you enter a strong. Using the above example where the price to pay is 24p, you can see that the amount of change needed is printed, followed by the type of money and amount given that the customer receives as change.

In this example their cost of items is 12p, and they paid with a 20p. The till only contains 6 1ps.

The system recognises that it cannot completely pay all the change, but still gives as much as it can (all6 of the 1ps), and lets the customer know that we owe them 2p.

```
Problems  @ Javadoc  Declaration  Console ⊠
<terminated> Shop [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (16 Mar 2015 18:15:55)
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
3
Please enter Identifier of length 4 (Must be integers):
1234
This item costs £ 0.12
Your price to pay (in pennies) is: 12
Continue? (Y/N)
n
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
4
Hand over the money!
Enter the denomination type. One of:
1p 2p 5p 10p 20p 50p £1 £2 £5 £10 £20 £50 20p
Please enter amount of currency
1
Price left to pay is -8
Not enough change sorry! I owe you 2p
Here is your change:
1p 6
Welcome to Mrs Miggins Pie Shop. Please enter choice:
1 - Stock the shop
2 - Add coins to the till
3 - Process customer order
4 - Process customer payment
5 - Display till balance
6 - Exit shop program
```