Bakery Software Proposal

Prepared by: Brian Desnoyers (bdesnoy@ccs.neu.edu), Lucas Magee (mageel@ccs.neu.edu)
Group number: 10
June 18, 2014

TABLE OF CONTENTS

Requirements Specification 2
Module Dependency Diagram 5
UML Diagram 6
Team Member Contributions 7

REQUIREMENTS SPECIFICATION

Use Case 1: Enter Order of Two Items for New Customer

1. User selects “Add Order”.
2. System prompts user to select customer from list.*
3. User selects “Add a new customer” from the drop-down menu.
4. System prompts user to enter the customer’s last name.
5. User enters client’s name and clicks “OK”.
6. System prompts user to enter the customer’s address.
7. User enters client’s address and clicks “OK”.
8. System prompts user to enter the customer’s city
9. User enters client’s city and clicks “OK”.
10. System prompts user to select the customer’s state.
11. User selects customer’s state from the drop-down menu and clicks “OK”.
12. System prompts user to enter the customer’s zip code.
13. User enters client’s zip code and clicks “OK”.
14. System prompts user to enter the order pick-up date from a series of drop down menus.
15. User selects the order pick-up date and clicks “OK”.
16. System prompts user to select the item to add to the order.
17. User selects the item from a drop-down menu and selects “OK”.
18. System prompts user to enter the quantity of the item.
19. User enters the quantity and selects “OK”.
20. System asks user if they would like to add another item.
21. User selects “Yes”.
22. User selects the item from a drop-down menu and selects “OK”.
23. System prompts user to enter the quantity of the item.
24. User enters the quantity and selects “OK”.
25. System asks user if they would like to add another item.
26. User selects “No”.
27. System asks user if the client is paying for this item now.
28. User selects “Yes” or “No”.
29. System returns to start screen and prints the order receipt to the console.

*Note: The previous use case asked user to search for client by phone number. This was changed due to the file given by the client which did not include phone numbers. The addition of the pick-up date was added based on the client's specifications as well.

Use Case 2: Enter Order of Two Items for Existing Customer

1. User selects "Add Order".
2. System prompts user to select customer from list.*
3. User selects the existing customer from a drop-down menu and selects "OK".
4. System prompts user to enter the order pick-up date from a series of drop down menus.
5. User selects the order pick-up date and clicks "OK".
6. System prompts user to select the item to add to the order.
7. User selects the item from a drop-down menu and selects "OK".
8. System prompts user to enter the quantity of the item.
9. User enters the quantity and selects "OK".
10. System asks user if they would like to add another item.
11. User selects "Yes".
12. User selects the item from a drop-down menu and selects "OK".
13. System prompts user to enter the quantity of the item.
14. User enters the quantity and selects "OK".
15. System asks user if they would like to add another item.
16. User selects "No".
17. System asks user if the client is paying for this item now.
18. User selects "Yes" or "No".
19. System returns to start screen and prints the order receipt to the console.

Use Case 3: View* Client's Information (includes viewing loyalty card balance)

1. User selects "Show Customer Information"
2. User selects the customer from the drop-down and clicks "OK".
3. They system shows the user the selected customer's information.
4. The user selects "OK".
5. There system to the start screen.

*Note: Editing a client's information has been made a separate step for GUI simplicity. A user can edit a client's information by selecting "Edit Customer".

Use Case 4: Add an Item for Sale

1. User selects "Add Item" under item management.

2. System prompts for item name.
3. User enters item name and selects submit.
4. System prompts user for the item category.
5. User selects the item category from the drop-down menu and selects “OK”.
6. System prompts user for item price.
7. User enters the price and selects “OK”.
8. System returns to start screen.

*Note: Categories were added due to new specification by the client.

Use Case 5: Viewing All Items and Their Prices Without Placing an Order

1. User selects “Printed Reports” under “Bakery Management”
2. System prompts for the report type.
3. User selects “Items for Sale” and clicks “OK”.
4. System prints list of all items and their prices to the console and returns to start screen.

Use Case 6: Viewing All Orders for A Customer

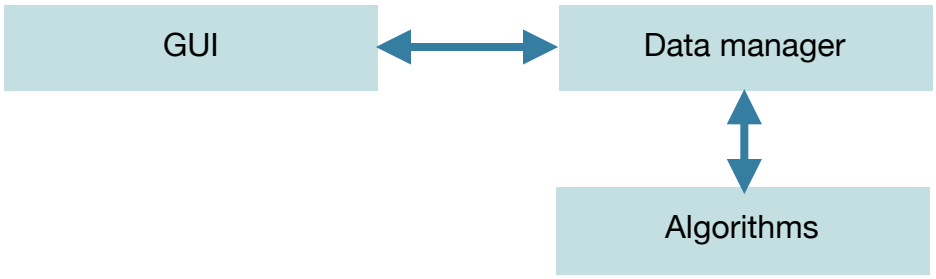
1. User selects “Printed Reports” under “Bakery Management”
2. System prompts for the report type.
3. User selects “Orders By Customer”.
4. User selects the customer from the drop-down menu and clicks “OK”.
5. System shows user clients orders.
6. User clicks done.
7. System returns to start screen.

List of Functional and Non-Functional Requirements

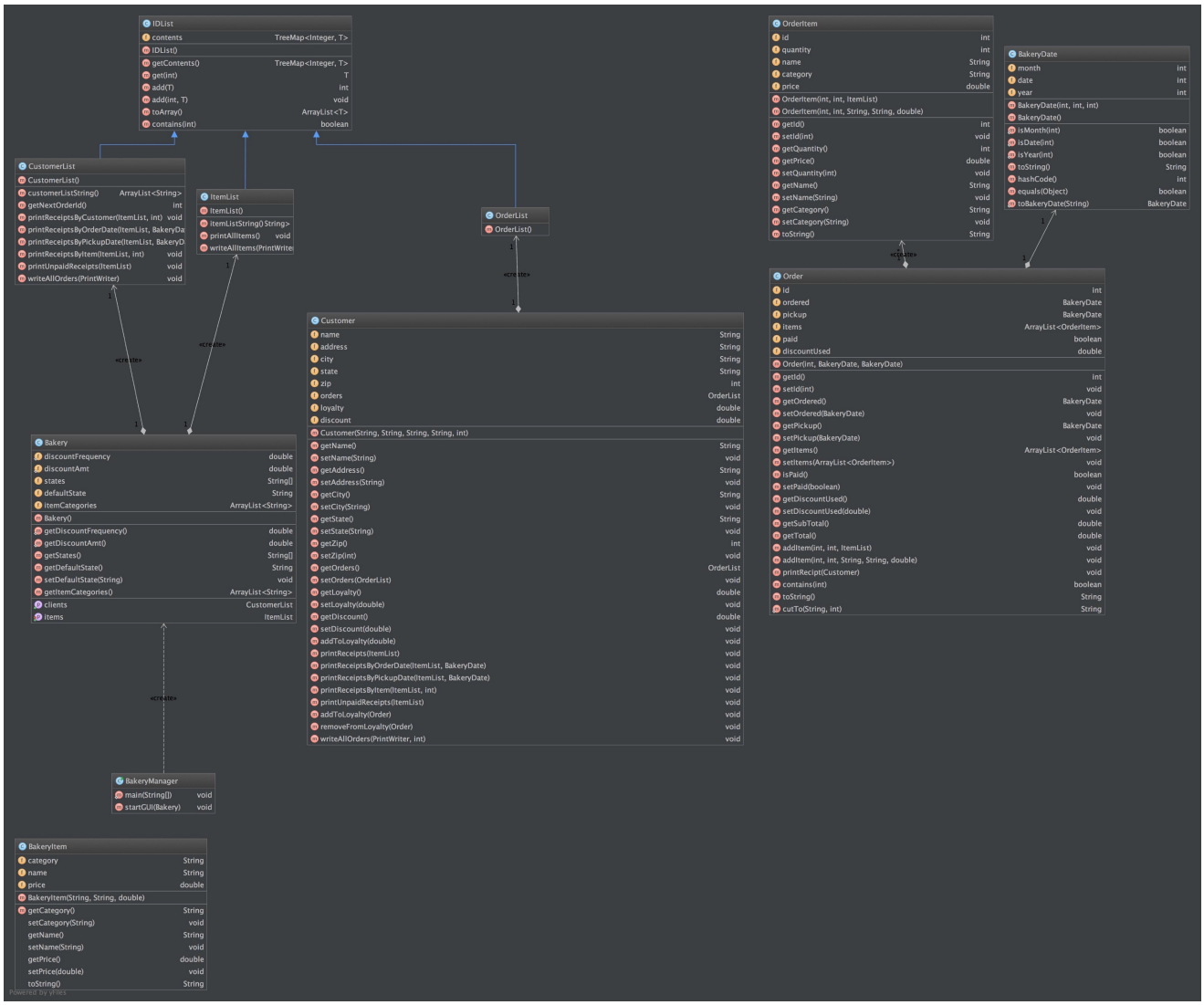
- The system shall allow the user to enter orders for existing customers by selecting them from a menu.
- The system shall allow the user to enter orders for non-existing customers by prompting the user for the customer’s information when the customer does not appear in the drop down.
- The system shall allow the user to enter orders for an unlimited number of items.
- The system shall allow the user to select the item being sold from a list and automatically update the price to that of the item.
- The system shall automatically update the client’s royalty card balance when an order is placed and automatically apply any royalty card discounts to the order total.

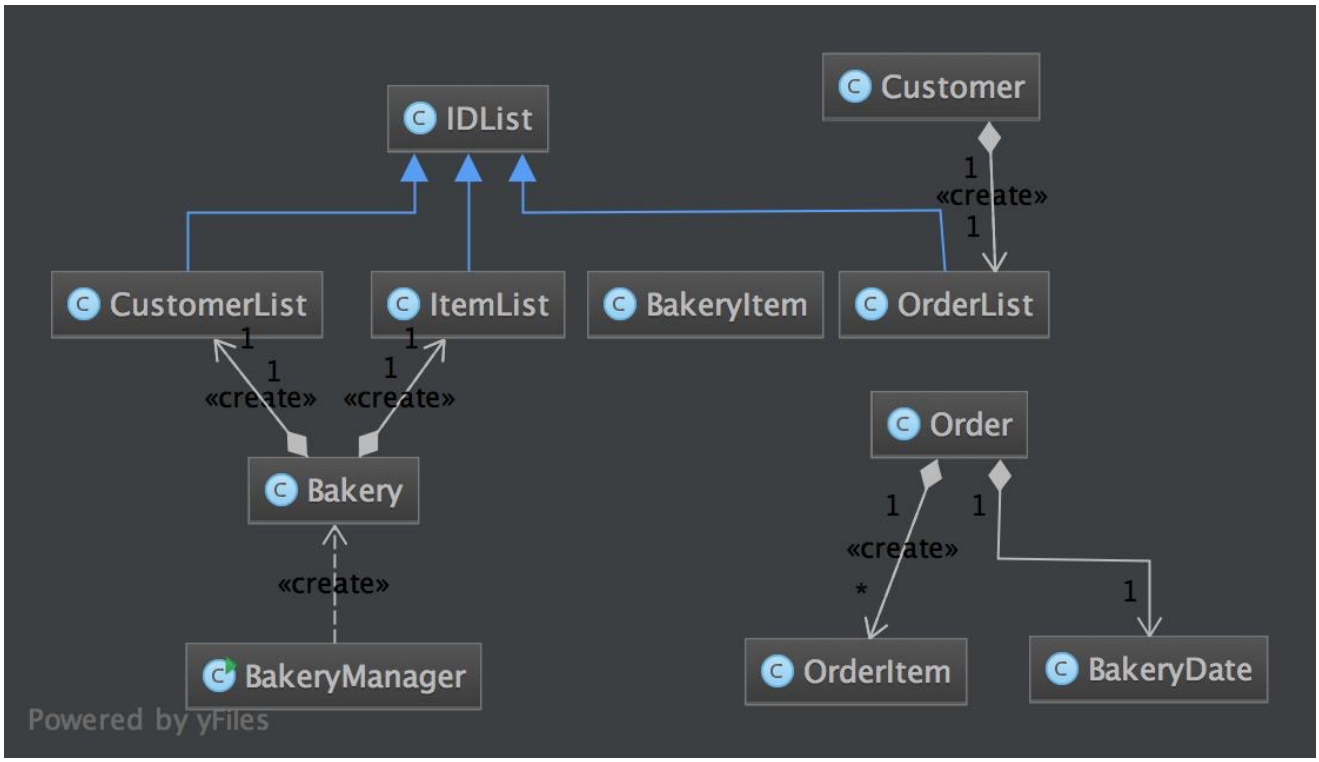
- The system shall allow the user to enter additional items that the bakery sells.
- The system shall allow the user to modify a client's information.
- The system shall allow the user to view all of the bakery items and their prices without placing an order.
- The system shall allow the user to view all of the orders for a particular user.
- The system shall respond within a reasonable amount of time, such as ten seconds, when there are under 1,000 customers and 1,000 bakery items.
- The system shall allow for the import of the current state in two files, in the same format as those provided by the bakery owner.

MODULE DEPENDENCY DIAGRAM



UML DIAGRAM


IDList
contents TreeMap<Integer, T>
IDList()
getContents() TreeMap<Integer, T>
get(int) T
add(T) int
add(int, T) void
toArray() ArrayList<T>
contains(int) boolean
CustomerList
CustomerList()
customerListString() ArrayList<String>
getNextOrderId() int
printReceiptsByCustomer(ItemList, int) void
printReceiptsByOrderDate(ItemList, BakeryDa
printReceiptsByPickupDate(ItemList, BakeryD
printReceiptsByItem(ItemList, int) void
printUnpaidReceipts(ItemList) void
writeAllOrders(PrintWriter) void
ItemList
ItemList()
itemListString() String>
printAllItems() void
writeAllItems(PrintWrite
OrderList
OrderList()
«create»
Customer
name String
address String
city String
state String
zip int
orders OrderList
loyalty double
discount double
Customer(String, String, String, String, int)
getName() String
setName(String) void
getAddress() String
setAddress(String) void
getCity() String
setCity(String) void
getState() String
setState(String) void
getZip() int
setZip(int) void
getOrders() OrderList
setOrders(OrderList) void
getLoyalty() double
setLoyalty(double) void
getDiscount() double
setDiscount(double) void
addToLoyalty(double) void
printReceipts(ItemList) void
printReceiptsByOrderDate(ItemList, BakeryDate) void
printReceiptsByPickupDate(ItemList, BakeryDate) void
printReceiptsByItem(ItemList, int) void
printUnpaidReceipts(ItemList) void
addToLoyalty(Order) void
removeFromLoyalty(Order) void
writeAllOrders(PrintWriter, int) void
Bakery
discountFrequency double
discountAmt double
states String[]
defaultState String
itemCategories ArrayList<String>
Bakery()
getDiscountFrequency() double
getDiscountAmt() double
getStates() String[]
getDefaultState() String
setDefaultState(String) void
getItemCategories() ArrayList<String>
clients CustomerList
items ItemList
BakeryManager
main(String[]) void
startGUI(Bakery) void
BakeryItem
category String
name String
price double
BakeryItem(String, String, double)
getCategory() String
setCategory(String) void
getName() String
setName(String) void
getPrice() double
setPrice(double) void
toString() String
OrderItem
id int
quantity int
name String
category String
price double
OrderItem(int, int, ItemList)
OrderItem(int, int, String, String, double)
getId() int
setId(int) void
getQuantity() int
getPrice() double
setQuantity(int) void
getName() String
setName(String) void
getCategory() String
setCategory(String) void
toString() String
BakeryDate
month int
date int
year int
BakeryDate(int, int, int)
BakeryDate()
isMonth(int) boolean
isDate(int) boolean
isYear(int) boolean
toString() String
hashCode() int
equals(Object) boolean
toBakeryDate(String) BakeryDate
Order
id int
ordered BakeryDate
pickup BakeryDate
items ArrayList<OrderItem>
paid boolean
discountUsed double
Order(int, BakeryDate, BakeryDate)
getId() int
setId(int) void
getOrdered() BakeryDate
setOrdered(BakeryDate) void
getPickup() BakeryDate
setPickup(BakeryDate) void
getItems() ArrayList<OrderItem>
setItems(ArrayList<OrderItem>) void
isPaid() boolean
setPaid(boolean) void
getDiscountUsed() double
setDiscountUsed(double) void
getSubTotal() double
getTotal() double
addItem(int, int, ItemList) void
addItem(int, int, String, String, double) void
printRecipt(Customer) void
contains(int) boolean
toString() String
cutToString, int) String



TEAM MEMBER CONTRIBUTIONS

Both team members, Brian Desnoyers and Lucas Magee, worked on the requirements and specifications together at the same time on the same computer.

They worked together on most of the implementation with Brian Desnoyers focusing mostly on the GUI design and black box testing and Lucas Magee working on additional documentation and white box testing.