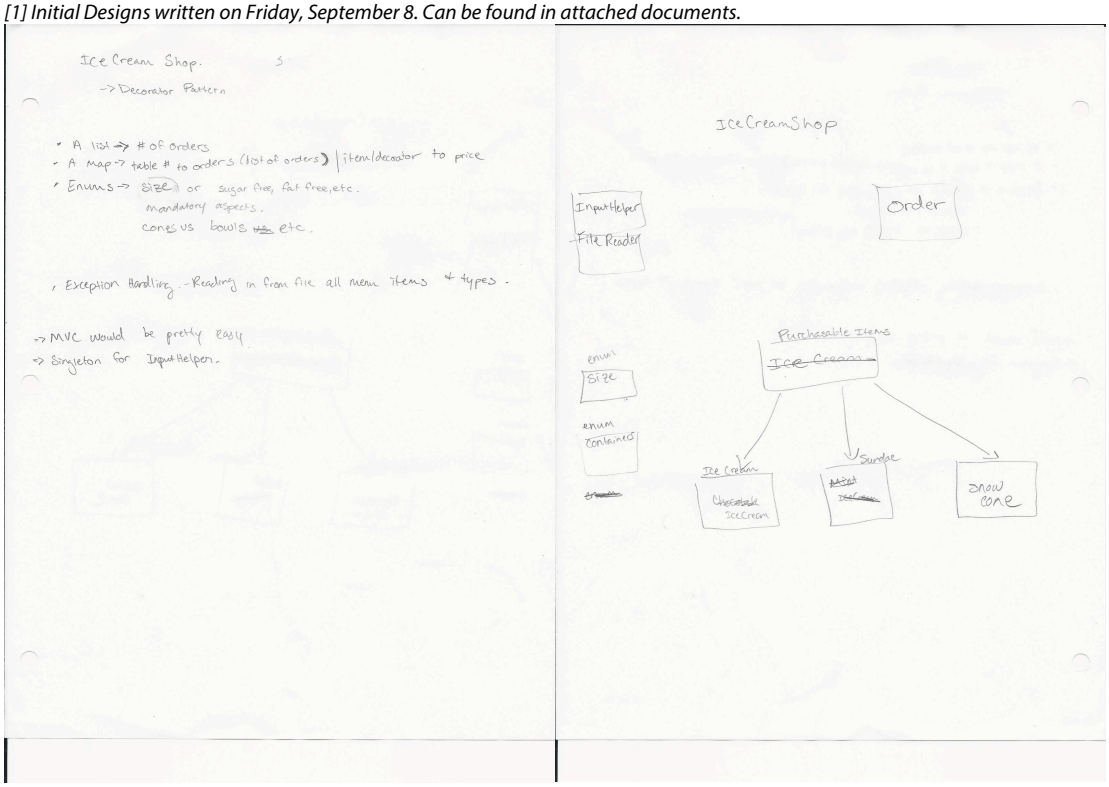
Ice Cream Shop: Team Write-up
Paul Wat, Jeremy Aftem

I. Timeline

The team met on Friday to discuss the general design of the program. At that meeting, the team brainstormed two documents for the overall design of the project. These scanned documents are attached in pdf form alongside this document [1]. We decided on developing an Ice Cream Shop initially with the decorator design pattern to decorate ice cream/purchasable items, but this conflicted with an idea of ours to read information in from a file to construct the Ice Cream Shop's menu according to the file. It was settled to use a MVC (model-view-controller) pattern. Since Friday, the team has been working separately to finish the first iteration of the project due Wednesday.

[1] Initial Designs written on Friday, September 8. Can be found in attached documents.



II. Model View Controller

The 'View' is everything that the user sees, or the input prompts and output command line "UI." Enums used for types of ice cream or gelato and the order history (a map) are used as a model and our "controllers" would consist of all of the logic mostly seen in the IceCreamShop class.

A public Github repository was also created to work on this project on Friday. The following README.md was created to outline our "Design.pdf" designs [2]:

[2] Github repository README.md

README.md

Assignment 4: Group Project OOP design principles

Brainstorm

Program will be an Ice Cream Shop.

Requirements

- A list : list of orders
- A map : table # to orders (decided was bad) or item/decorator to price
- Enums : size or sugar free, fat free, etc.
- Exception handling: Reading in from file all menu items + types.

Classes

'View'

- InputHelper

'Model'

- FileReader
- Enums -> size, containers (cone vs cup, etc)

'Controllers'

- IceCreamShop
- Order
- PurchasableItems -> SnowCone, Sundae, IceCream

Design Paterns

We could use MVC easily and Singleton design pattern for InputHelper.

Since these initial designs, the project now has an Enum Container to represent the type of container (bowl or cone) of the ice cream and gelato. Initially Gelato and Ice Cream classes (subclasses of Item) were factories but served no purpose in being a factory and was scrapped to just be regular subclasses of Item.

III. Software Requirements

1. **An Interface**
    a. `ShopUI` is an interface of which `ConsoleShop` implements
    b. `Item` is an abstract class of which `IceCream` and `Gelato` extends
2. **Inheritance**
    a. `IceCream` and `Gelato` inherits from `Item`
3. **A list**
    a. The Item class has a `Toppings` list and a `Type` list. Used Java's ArrayList.
4. **A map**
    a. An order history map which mapping a customer's name (string) to their `Order`. Used Java's HashMap
5. **Exception Handling**
    a. ConsoleShopUI –
    b. IceCreamShop – Catches an exception if enum value not recognized in `askForEnumOption` method
6. **Enums**
    a. `Topping` – describes the toppings used on an Item
    b. `Type` – describes if Item is fat free, half and half or sugar free.
    c. `Container` – bowl or cone

IV. Documentation

**Only includes documentation for more complicated classes. Eg., enums are not documented. Classes are sorted in alphabetical order.*

ConsoleShopUI

Description:
A view. Acts as an input handler to retrieve input from the user.

Constructors:

- ConsoleShopUI()
  Creates a scanner

Methods:

- promptUserForBoolean(String prompt) : void
  Prompts the user for a boolean variable. Displays prompt before it receives input.

- promptUserForInteger(String prompt) : void
  Prompts the user for an integer variable. Displays prompt before it receives

input.

- promptUserForString(String prompt) : void
  Prompts the user for a string variable. Displays prompt before it receives input.
- displayMessage(String prompt) : void
  Displays prompt.

- displayMenu(String menu) : void
  Displays menu.

- grabInput(String prompt) : void
  Uses scanner.nextLine() to receive input after prompt is displayed to user.

Gelato

Description:
Inherits from Item. Gelato is one of two menu items which can be bought by customers.

Constructors:

- Gelato()
  Calls parent class constructor and adds a Gelato's higher price to the base price.

- Gelato(Boolean hasLowButterfat, boolean isItalianImported, FruitFlavor flavor)
  Calls parent class constructor Item() and sets class variables to remaining parameter values.

- Gelato(Size size, Container container, List<Types> types, Boolean hasLowButterFat, Boolean isItalianImported, FruitFlavor flavor)
  Calls parent class constructor Item(size, types, container) and sets class variables to remaining parameter values.

Methods:

- Standard Getters and Setters for class variables.

- toString() : String
  Returns a string descriptor of the Item: low butterfat or normal, Italian imported or not, fruit flavor, size, container, type.

IceCream

Description:
Inherits from Item. IceCream is one of two menu items which can be bought by customers.

Constructors:

- IceCream()
  Calls parent class constructor Item()

- IceCream(Size size, Container container, List<Type> types, TraditionalFlavor flavor)
  Calls parent class constructor Item(size, types, container) and sets class variables to remaining parameter values.

Methods:

- Standard Getters and Setters for class variables.

- toString() : String
  Returns a string descriptor of the Item: size, container, type, and name.

IceCreamShop

Description:
Contains main program method which creates an IceCreamShop. The IceCreamShop is the controller which handles the model and views of the program. It has a ShopUI, which is the view, and asks users for action and optionally items to add to an order. It allows a user to view his or her order.

Conceptually the IceCreamShop is open all day and closes at night. Orders then are constantly being made until the IceCreamShop closes (the program ends).

Constructors:

- IceCreamShop()
  Constructs a new ConsoleShopUI and HashMap for order history. Displays a welcoming message to user.

Methods:

- askForAction() : void
  Asks if you want to buy items, view order, cancel order, or pay.

- askForItem() : Item
  Asks if user wants a Gelato or IceCream.

- askForIceCream() : IceCream
  Asks for size, container, types, and traditional flavor of ice cream and constructs a new IceCream object to be returned.

- askForGelato() : Gelato
  Asks for size, container, types, fruit flavor of gelato, if low butterfat and if Italian imported, and constructs a new Gelato object to be returned.

- viewOrder(Order order) : void
  Displays current order to user.

- closeShop() : void
  Calls prettyOrderHistoryString() and ends program.

- continueSelling() : boolean
  Asks user if we are still selling orders at IceCreamShop. Returns true if user wishes to continue selling, false otherwise.

- prettyOrderHistoryString() : String
  Prints order history – prettily.

Item

Description:
An item which can be sold in the IceCreamShop. This is an abstract class.

Constructors:

- Item()
  Does nothing but construct the object.

- Item(Size size, List<Type> types, Container container)
  Sets class variables to constructor parameters.

Methods:

- Standard getters and setters for all class variables.

- toString() : String
  An abstract method that is not implemented but to be implemented by inherited classes.

Menu

Description:
Reads menu from file

Constructors:

- Menu(String filename)
  Calls readFromFile

Methods:

- readFromFile(String filename) : boolean
  Reads menu names from file

Order

Description:

- An object which represents an individual customer's order. Has a link of Items which the customer has purchased.

Constructors:

- Order()
  Creates a new list of Items.

Methods:

- addItem() : void
  Adds an item to the order.

- calculatePrice() : double
  Calculates and returns the price of the order.

- getSize() : int
  Returns size of order

- toString() : String
  Returns a string of all the items in the order.