**Final Project Documentation**

Aaron Borjas and Trenton Nogle

**Outline of problem/purpose statement**

Generally, we want to create a simulation of an ice cream shop, where customers come in and give orders, and those orders are made based on currently available ingredients. When the customer is served, the store gains money, and when the store has to buy new ingredients for the ice cream they lose money.

How it works:

-customer enters the simulation

-customer has certain preferences, and makes an order based on what ingredients are available or if the customer is desperate for their top preference, the worker has to make new ice cream and/or toppings that the customer wants.

-the shop makes the ice cream that the customer ordered

-the shop gives the customer their ice cream and the customer leaves

-this process repeats for a certain number of "days" and finishes once the final day is over, saying the total profit/loss, the number of customers, and the ending amount of money.

**Requirements**

-This project needs a shop that interacts with customers, each who has their own order preferences

-The shop contains a line of customers, and has functions to help customers make orders and receive those orders

-customers have different ice cream preferences and will order stuff that is currently available

-Ice cream is composed of *ONLY ONE* flavor, topping, and base.

-Data files with ice cream flavors, bases, toppings, and customer names are required for file I/O and to make it so that we don't have to hard-code different elements. We can just select a random flavor, topping, and base for an ice cream so that life is easier when going to implement the code.

**Assumptions:**

-ice creams only have one base, flavor, and topping

-the shop buys the ice cream instead of making it

-each customer uses one serving of a base, flavor, and topping when getting their ice cream

-each respective category of stuff on the ice cream costs a fixed amount, even if the actual item is different

-the customers aren’t picky about what they want

**UML Diagrams:**

|  |
| --- |
| Shop |
| -vector<int> flavorServings  -vector<int> toppingServings  -vector<int> baseServings  -vector<string> flavors  -vector<string> toppings  -vector<string> base  -vector<string> names  -queue<Customer> line  -float money |
| +Shop()  +Shop(vector<int> totalFlavor, vector<int> totalTopping, vector<int> totalBase, vector<string> f, vector<string>t, vector<string> b, vector<string> n, float start)  +float getMoney()  +void makeMoney(float amount)  +void spendMoney(float amount)  +void addServings(int numServings, vector<int> servings, int index, string type)  +void removeServings(int numServings, vector<int> servings, int index, string type)  +queue<Customer> getQ()  +void addCustomer()  +void removeCustomer()  +IceCream makeOrder(Customer& cust) |

shop has customers

|  |
| --- |
| Customer |
| -vector<IceCream> preferences  -string name |
| + Customer()  +Customer(vector<string> bases, vector<string> flavors, vector<string> toppings, vector<string> names)  +string getName()  +IceCream getPreference(int index)  +vector<IceCream> getPreferences()  +void generatepreferences(int numPreferences, vector<string> bases, vector<string> flavors, vector<string> toppings)  +void printPreferences()  +string toString()  +friend ostream& operator<<(ostream& out, Customer cust) |

customers has an ice cream

|  |
| --- |
| IceCream |
| -string base  -string flavor  -string topping |
| +IceCream()  +IceCream(string b, string f, string t)  +string getBase()  +string getFlavor()  +string getTopping()  +string toString()  +friend ostream& operator<<(ostream& out, IceCream, cream) |

**Shop:**

Shop is a class that helps the customers interact with the ice cream. The shop considers the preferences of the customer and helps decide what they actually order by seeing what ice cream the store has in stock currently. If the store is out of stock of any base, flavor, or topping of a preferred ice cream, the store checks what their next preference is. If the store detects that it is out of anything, it spends money to get those ingredients. Once the customer buys an ice cream, they pay the store money and then leave.

**Customer:**

Customer is a class that interacts with ice cream. Each customer has a name, for distinction, and their own set of ice cream preferences. Upon creation, the customer's preferences are randomly created based on the number of toppings, flavors, and bases available, respectively.

**Ice Cream:**

The Ice Cream class is the most basic, but most essential class of this program. Each ice cream has a base, a flavor, and a topping. ONE OF EACH, no more, no less. The ice cream class only really contains interface functions to get these variables, but it also has an operator overload for << which prints the ice cream easily.

**Sources:**

<https://www.mother.ly/news/the-most-popular-baby-names-of-2018> was used to get data for the names.dat data file.

<http://www.cplusplus.com/reference/queue/queue/> was used to learn about the queue data structure and how the methods worked.