Socially Responsible Computing - Computer Aided Farming

Technical Paper

Team Name: Ctrl-Alt-Defeat

Hasti Abbasi Kenarsari

Ryan Wei

Medha Swarnachandrabalaji

Kenzie Lam

Alex Auyon

Professor Qichao Dong

CS 2400 Section 1

May 5, 2024

Table of Contents

Introduction

Socially responsible computing entails the ethical design principles intended to guide decision making processes within the realm of programming. Tasked with the challenge of designing an application to tackle a specific problem at a local farm, the foundation of our program was built on the principles of socially responsible computing.

Lopez Urban Farm, a large-scale initiative nestled in Pomona, California, operates in a synergistic collaboration with the Pomona Unified School District and other entities, including Cal Poly Pomona. Situated adjacent to Lopez Elementary School, the farm serves as a beacon of hope in addressing food security issues through its accessible “take what you need, pay what you can” model. In addition to cultivating a variety of fresh produce, the farm leverages its close ties with Pomona Unified to provide invaluable eco-education opportunities for students, fostering a new generation of environmentally conscious citizens. As a cornerstone of the community, the farm’s role extends beyond agriculture; it is a hub for learning, sharing, and growing together.

The Problem

The farm currently faces challenges in maintaining its bookkeeping, particularly in tracking inventory, donation income, and clientele. During our interview with the farm's owner, it became evident that the establishment lacks a robust system for maintaining concrete records pertaining to the availability of products, client list, and weekly donation amounts. The availability of produce is loosely recorded on a whiteboard, and the flow of clients is not adequately documented. This lack of structured record-keeping makes it difficult to understand the specific needs of each client and how the farm can optimally meet those needs while sustaining its operations through donations.

Not only does this present an issue for the workers at the farm, but also for the members of the community who depend on the valuable resources it provides. As computer scientists, we have a social responsibility to foster the growth of programs that are curated towards the public interest. Specifically, programs that increase the greater good of society. In recognition of this task, we worked diligently to tackle this specific problem within our community.

Our Deliverable

To address the farm's bookkeeping issues, we created a customer-facing program which allows the user to view, order, and generally interact with the products of the farm. Users can login by entering their choice of personal identification - information that will later be used to keep track of and differentiate clients. There are several fields where users can opt in to personalize their data. Factors such as username, address, phone number, email address, and age are accounted for. Clients can also edit their personal information later. The most critical piece of client information is the username, which is predominantly linked to each customer.

Upon finalizing their information, clients are able to select certain quantities of produce depending on their in-season availability and general stock count. Edge cases have been addressed to ensure that there are no issues in removing items from the inventory and associating them with the client. Extra functionality has been added to ensure that canceled orders are properly addressed. In essence, the program allows potential customers to have a better, clearer grasp on what is available and makes products more accessible. This system is much more efficient and less prone to error in comparison to the current white board inventory publication that resides at the farm.

Additionally, workers at the farm are able to keep track of their clients, pending orders, and inventory through a terminal-based application. To ensure optimal efficiency for this component of our project, we provided a separate farmer user interface that allows easy access to the list of clients, general inventory, and other key information about the farm. Not only will this functionality allow individuals working at the farm to quickly access the information they need about their customers, but it will also pave the way for statistical analyses of the overall function. To further explain, the extrapolation of client data will allow the hardworking team at Lopez Urban Farm to predict future trends regarding the demand for specific produce items per season, allowing them to maximize their output once our program is initiated. Our application is simple and intuitive, and is expected to greatly increase the efficiency and organization of the farm.

Implementation

In order to meet the needs of our client, we employed two abstract data types, a dictionary and a list, primarily to manage the data of the farm and its clients. The dictionary is used as a key value pair of the list of the farm's clients and their orders. Since the farm's clients will frequently be added and removed from the dictionary, we opted for a sorted linked implementation. This implementation is just a standard implementation. The list data type is used to keep track of the farm's inventory. Since this list will also see frequent addition and removal, we decided to implement a linked list. However, we had to modify the standard implementation. As each produce in the inventory requires a stock count, we added an additional integer field count with the required accessor and mutator to modify the value as needed. As a direct result of this decision, LinkedList required additional functions to be able to access the count field within Node. However, it is important to note that this doesn't affect the client's interfacing with the code, only reducing the overall efficiencies of the program.

We additionally created two objects to encapsulate the data of the farm's client and produces, called Client and Produce. The Client object takes in five fields: a string called username, an integer age, a string address, a string phone, and a string email, of which only the former two are required. The address is completely optional, and between the phone and email, either or is optional, but not both. We chose this approach as the farm primarily focuses on helping those in need, and those people might wish to remain as anonymous as possible. This approach allows us to respect that wish without having the farm come upon legal trouble. The Produce object contains three fields: a string name, a string season, and a boolean inSeason. The constructor takes in the name of the produce and the season in which it is grown in, and then automatically checks whether the crop is in season or not using Java's LocalDate library.

Class ClientBase utilizes the dictionary to effectively manage the client's data. Its main job is to handle the interfacing between the user and the sorted linked dictionary data type, making add/remove/view operations clearer and more concise. When an object of class ClientBase is first initialized, it creates (if the file doesn't already exist) a plain text file called clients.txt to allow for saving of information. If the file does exist, it reads in the file instead. Each client in the file is separated by lines, and each piece of information regarding the client separated by semicolons. The constructor sequentially reads in each string or integer, which corresponds to the fields of the Client object, and subsequently adds it to a Client object. After the fields are filled, it begins to add information into Produce objects, alternating between name and season until it hits a new line character, then adds the Client object and an array of Produce objects as a key value pair into a newly initialized dictionary. Once it hits a new line character, the constructor stops reading in Produce objects and restarts the process. This continues until it hits the end of the file.

The array of Produce objects serves as the list of orders the client requested. Whenever a Client object or Produce object is added, removed, or modified, the method SaveToFile is called and the new changes are updated to client.txt. The class contains methods for adding, removing, or retrieving a specific client or all clients, checking if a client exists, checking the quantity of an order, and getting a specific order or orders of all clients. Adding an order is worth mentioning separately, as well as methods removeOrder and cancelOrder. When an order is added, the

Produce stock is directly removed from the Inventory. This removal returns the Produce object, which is then added to the Produce array representing the client's orders. For the methods removeOrder and cancelOrder, the difference lies within the intention of the removal/cancellation. When removeOrder is called, this assumes that the order has been fulfilled and therefore the number of Produce objects equaling the quantity fulfilled is removed from the Produce array of the client. When cancelOrder is called, this assumes that the order has not been fulfilled and is otherwise canceled for one reason or another, and as a result the Produce is added back into the stock of Inventory as well as removed from the Produce array.

Class Inventory utilizes the list data type to manage the data of the farm's inventory. It handles the interfacing between the user and the list data type containing the farm's inventory. Like class ClientBase, class Inventory has its own corresponding plain text file to save data to, inventory.txt. Again, the data is separated by semicolons and Produce objects are created and added to a newly initialized LinkedList, and the data is updated for every modification. The class contains methods to add/remove/retrieve produce, to check if a produce is in stock, and to retrieve the stock of a produce.

The classes FarmerUI and ClientUI combine the above classes and methods to present an interactive menu for our client. When first running, the program will ask whether the user's role is as a farmer or a client of the farm. Choosing the former runs FarmerUI, presenting the user with options to manage the inventory, orders, or clients as well as exit the program. Choosing the latter will run ClientUI, which brings up a graphical user interface (GUI) for the farm's clients to interact with and presents options to sign up or log in. After the user logs in, they're able to view and change their own information, view inventory stock, and request or cancel their orders. We opted for a terminal based front-end for the farmer as we wanted to emphasize simplicity and efficiency, allowing the user to easily select options and enter inputs. For the farmer's client, we opted for something more presentable and user friendly, therefore utilizing java swing to create a GUI for form as well as function.

Discussion

In the final analysis, the culmination of our project components was successful. We fulfilled all of the requirements and even went beyond them by creating a GUI for the client with Java Swing. The Lopez Urban Farm is carrying out its honorable mission of helping people who are undergoing food insecurity. The farm has challenges with keeping records in a structured way, so we fulfilled our intention of constructing a program that helps address this issue. In short, our application will help the farm become more efficient. It gives both the workers at the farm and the consumers a better user experience than they have right now. Our thorough testing using the JUnit5 framework made sure our application worked as intended.

However, as there are in any project, there are some things we could improve on. In hindsight, as mentioned in the "Implementation" section of this report, adding the integer field count in the Node class decreased the overall efficiency of our application. Also, more functions could have been moved to Inventory from LinkedList. This is because Inventory calls some LinkedList functions that could have been in Inventory itself. Additionally, instead of using a LinkedList for Inventory, we could have chosen a different data type such as a Bag or a Dictionary for better efficiency. We also have inconsistent method names that could be edited to make variable names more aligned with the name of the method they are in. For example, in the "addProduce" method in clientBase, the Produce array is named "orders", but it could have been named "produceOrders". Furthermore, in ClientBase, the clients are stored in a dictionary called clientBase where the key is the client and the value is an Array of Produce. This implementation's efficiency could be improved by using a LinkedList of produce for the value of clientBase instead of using an Array of Produce.

To improve our efficiency for our next projects, we will create a clearer outline of the project specifications. Even though our program has imperfections, our program still meets and exceeds the project's requirements.

In regards to scalability, we have many plans for our project in the future. In order to increase outreach, we hope to construct a public website with more advanced user interface details. This addition will help clients easily access our program in an organized manner, attributing to its extended accessibility. Moreover, we hope to implement a system that takes donations into consideration. This will corroborate with the farm's "take what you need, pay what you can" model, increasing its contribution to the greater society. Another component of our project's future involves an updated login system that uses encryption, ensuring that user data (username, address, etc.) is protected. Likewise, we hope to set up two factor authentication to increase security.

Conclusion

As part of our socially responsible computing initiative, we prioritized ethical design principles to guide decision making processes within the realm of our programming. We made a program to help workers at the farm with efficiently keeping track of their records and made an appealing GUI for the clientele to use. In essence, our project deliverable successfully addresses the bookkeeping shortcomings of the Lopez Urban Farm. The farm serves a vital purpose in the community, not only providing easier access to food for everyone but also cultivating an environment of closeness and comradery. We are certain that our program will make a significant impact on the farm and the community that depends on it.