CIS18A Course Project : Documentation of the Project

Java program: Grocery App Documentation Guide -2024 (Immanuel Morris)

Introduction

1.1 Purpose

The purpose of this program is to give a user the option of choosing a meal plan for delivery. The options are very simple and direct, and a generous menu option between a group of meats and vegetables serve as choices for the user to select. As the user makes their package choice, a total of their order will be calculated, along with a desired pickup time. At the end of their order, a recap of their selected options will be displayed.

1.2 Intended Audience and Users

This program is for users from all walks of life. Whether the user is a student, parent, professional, its accessibility remains the same for all.

Features

Below are the main features of the program

2.1 Options

- Display Food Options: Lists all available veggies and meats with their prices, enabling users to make informed decisions.
- **Selection Process**: Users can choose between purchasing veggies, meats, or a combination of both. The class then calculates the total cost based on the selection.
- **Pickup Time Selection**: After choosing their items, users can select a preferred pickup time slot. The system prompts for re-selection if an invalid option is chosen.
- **Order Recap**: Provides a summary of the user's order, including selected items and the total cost.

Methods

Below are the methods that are carried through the program (in camel case):

3.1 displayFoodOptions

- Description: Displays a list of all available food items categorized into veggies and meats, along with their prices.
- Parameters: None.
- Returns: Void.

3.2 displaySelection

- Description: Initiates the selection process by displaying a welcome message and options for the user to choose from (veggies, meats, or a combo).
- Parameters: None.
- Returns: Void.

3.3 getUserChoice

- Description: Captures the user's selection. Validates the input to ensure it is within the expected range (1-3).
- Parameters: None.
- Returns: An **int** representing the user's choice.

3.4 processUserChoice

- Description: Processes the user's choice by calculating the total cost of selected items.
- Parameters: **choice** - an **int** representing the user's selection.
- Returns: Void.

3.5 selectPickupTime

- Description: Allows the user to select a pickup time slot. Validates the input and loops until a valid choice is made.
- Parameters: None.
- Returns: Void.

3.6 getTotalCost and getSelectedTimeSlot

- Description: Getter methods to retrieve the total cost of selected items and the chosen pickup time slot, respectively.
- Parameters: None.
- Returns: **getTotalCost** returns a **double** representing the total cost. **getSelectedTimeSlot** returns a **String** representing the chosen pickup time slot.

3.7 Example Usage of Code:

```
GroceryList groceryList = new GroceryList();
groceryList.displaySelection();
int choice = groceryList.getUserChoice();
groceryList.processUserChoice(choice);
groceryList.selectPickupTime();
```

Program Strengths

One strength of the program lies in its robust layout in the GroceryList class. This class contains the heart of the program and allows for user error contingencies, while keeping a consistent flow and ease for user input. The adequate use of private components to protect the array display, and the use of the Switch statements show a testament to the functionality of its package and how its integration acts as a crucial component of the overall code. Additionally, the implementation of inheritance allows for seamless connections across all packages into the main class call.

Operating Envrionment

This program is designed for Java (Java Development Kit) JDK 11 and above, and can be run on an IDE of user choice.

Java can run on a PC environment with a Windows or Linux OS, along with Mac OS.

Recommended specs for Windows: Windows 7, 64 bit

Windows 10, 11, 64 bit

Design and Implementation Constraints:

If working in a Mac OS environment, please make sure a working version of Java is installed.

This program/app can be run online through various IDEs, examples include: Notepad++, Visual Studio, One compiler, etc.

Documentation Developer (Immanuel Morris)

Responsible for research that embodies the details that are contained within the program. In charge of communicating to the reader the fundamental functions of the program, and the basics of implementation along with the ideal computer specs.

Provides tips, tricks, shortcuts (if any) for users while running the program.

Displays thorough guidelines for running the program, and possible capabilities and limitations within a Mac or PC environment.

Software Developer (Immanuel Morris)

Responsible for the actual code written to produce and create the program. This person(s) has the knowledge in basic data structure and constantly provides updates and revisions for code. This person is also takes with debugging issues, and general oversight of the program.

Pseudocode (Immanuel Morris)

Approaches from a blueprint perspective of what the overall program needs to do prior to the actual program being created. Some functionalities, naming convention, and algorithmic implementations might be different than what's in the actual program, but the pseudocode captures the essence and integrity of what the developer wishes to crate.

The person writing the pseudocode is typically in the Software Development Department, and must have a clear view and understanding of the idea of the program.

The designer is well versed in connecting the thoughts of the program in a clear and concise order of steps, and have a vision for the final output of the program.

Assumptions and Dependencies

Since Java can be installed and run on a local machine (PC/Windows/Linux environment) there is no need for a true dependency if the user meets this requirement. For a short term, internet access may not be necessary.

On the contrary, if the user does not have access to a Windows or Linux environment, they may access a simulator online. This is to assume that the user has basic knowledge of internet accessibility, and will be required to do so barring that is the necessary route they choose to take.

Program Outlook (Future Improvements)

In future versions of the program, the expectation is to combine a UI element for ease of reading on the user eyes. A clear, well construed app with visuals will help the user experience improve mobility around the app, instead of showing a direct console display. Additionally, the goal is to allow for individual food section rather than just package output.

The plan is to allow for more user comfort, mobility on the screen, and adequate choices to customize their grocery order.