OOP Major Practical Design Document

McDonald's Self Serve Kiosk

By Michael, Reniel, and Jonty

Introduction

The project will implement a McDonald's self-serve kiosk system to facilitate the ordering of items from the menu without interacting with a staff member. It will be completely operated from the terminal, and include the ability to add, remove and customize items. This includes the ability to customize ingredients in burgers. It will also handle the collection of necessary payment information depending on the payment method chosen (PayPal, credit card, etc.).

Design Description

Assessment Concepts

Memory allocation from the stack and the heap

Arrays: Current cart will require a dynamically allocated array (stored on heap)
Strings: Item names will need to be stored on the stack
Objects: Each item of food will be a represented as an object

Parent: Payment
Children: Visa, Mastercard, PayPal, Cash, Bank

User input and output

I/O of different data types: Reading in the menu prices from a CSV file (strings and floats), and adding or removing items from the cart.

Input strings:

- Names of food and drink items.
- Customer name* (could just do a preset number instead)
- Payment information (cash and card)
- Quantity of items ordered
- Dine in or takeaway

Output:

- Receipt number
- Sum of price for all items.
- Menu and options

Object oriented programming and design

Inheritance: Different foods will all inherit from a base food object
Polymorphism: The utilization of the PaymentMethod class will
Abstract classes: PaymentMethod will be an abstract class, with a pure virtual method ‘pay’

Testing -

Test Inputs / Expected Outputs:

We will use individual text files to test the input and expected outcomes and record the results. We will do this for each class and file using multiple inputs that cover a range of situations.

Automated Testing:

We will utilise makefiles and automated testing files to automatically test multiple values and scenarios in every file and class. Each

Regression Testing:

Regression testing will be done often to make sure old code is still performing and integrating well with the new code.

Plan and roles -

Week 10:

Michael - Working on the input of maps and items into the files and finishing the items functionality. Specifically CSV files.
Jonty - Working on inheritance within the payment system and polymorphism within the different types of payments to make sure the payments are working.
Reniel - Working on inheritance with the checkout system, ensure that both dine-in and takeout options are functional after once the user has completed/confirmed their order.

Week 11:

Michael - Write unit tests for Burger, Fries, Item, SizedItem and Drink.
Reniel - Write and perform unit tests for checkout system, for both dine in and take away systems. Ensure they work with comments and stuff.
Jonty - Write and perform unit tests for payment system, credit card system, bank system, cash system, and PayPal system. Polish classes and make sure notation, comments, and cases are accurate.

All: integrating the payment system, items, and checkout together to make a finished product.
Also all: finishing the functionality of the entire program ready for presentation on Thursday.

Naming system -

File names, variables and methods in snake case.
Class names in camel case.

UPDATED WEEK 11 PLAN:

Reniel: finish off checkout system, add polymorphism method as well.

Jonty: add user input and other methods to payment system.

Michael: fix bugs and complete section of menu and alternative input.

ALL: On Tuesday combine all our parts and notation in order to get a completed whole project. On Thursday do the final polish and presentation practice ready for the final practical session.