Project Specification - Major Practical

Introduction

Our project is a café ordering system. It allows users to add or remove items to their liking. Users are able to see the descriptions of the menu items. Users can also print a summary of their current order. Finally, users can decide to checkout their current order.

Design Description

Assessment Concepts

Memory allocation from stack and the heap

- **Arrays:** We will use a dynamic array to store the items that are ordered by the customer.
- **Strings:** Items in the menu and details
- **Objects:** Briefly – Menu, Table, Restaurant

User Input and Output

- **I/O of different data types:** Command-line. User selects options from the given instructions corresponding to different numbers.

Object-oriented programming and design

- **Inheritance:** There are different types of items in a café. For example, we will have an item parent class with price and name as its attribute, subsequently we will have child classes that consist of more specific item types such as food or drinks with their own specific attributes.
- **Abstract class** : The base class, Menu will be an abstract class that serves as the base for various types of items in the café. Contains pure virtual function printSelection.
- **Polymorphism** : printSelection prints details about the choices that are associated with each type of item ordered by the customer

Class Descriptions

Order class

Class which stores the array of items ordered by the customer. Functions to checkout, add or remove items from the order are examples of methods that will be contained in this class.

Item parent class
This is an abstract class which contains general attributes of the item. It will have a pure virtual function.

Specific Item Classes(Child classes of the 'Item' class)
These are classes that contain details that are specific to the items. For example, types of foods and drinks.

User Interface
A user of this program is a guest to the café. They use the command-line. Users are presented with a list of options to choose from like:

Welcome to Hello Café!
Please select from one of the following options:
1. View menu
2. Add an item
3. Remove an item
4. Check your ordered list
5. Complete the order
Enter a number from the options above and press Return:

Code Style

All code in the program will be properly indented using 4 space tabs.
Classes will begin with a capital letter. Class files (.h and .cpp) will ...etc.
Function Comments will be given for each function that describes what the function does, what the return type represent and what any arguments are available for the function. Code Comments will be used to describe what each block of code performs if it is not obvious from just reading the code or the code is short. Comments will be written as the code is written.

Testing Plan

All our tests will run through our makefile each time we compile our code, automatically ... etc.

Unit Testing

Our unit tests will cover all public functions in our classes. Each class will have a corresponding test file like ClassNameTester.cpp , which will include a main method that will exhaustively test each function with a set of inputs and test it matches an expected output.

Regression testing

Regression testing is re-running functional and non-functional tests to ensure that previously developed and tested software still performs after a change. After new

functions are integrated into the system. We will run regression testing to ensure that the system as a whole, still functions as intended.

Functions that are coded will be reviewed together as a group. Makefiles will be made to test these functions.

Schedule Plan

Stretch Goals

Our goal is to complete this set of features.

Week 8 - Preparing for assessment in Week 9

Check-list:
Decide the overall structure of this program
Develop project plan and upload to svn.
Create SVN structure. - Jiajun
Organise code sharing with group members. - Hungyee
Write makefile. – Liewguan
Finalise testing strategy with makefiles and input/output files outlined in Testing Plan.

Break Week 1

Hold one or two meetings to discuss the overall direction of our coding and implementation.

Break Week 2

Work together to define objects and classes.

Week 9

Show the skeleton code and our design document
Add something fancy.
Add any feature that we haven’t thought yet.

Week 10

Show the early version of our program by implementing rest of objects. Test each part of our project to see if all requirements are satisfied.

Week 11

Make some refinement for our project. Add something fancy to our project to make it more challenging. Show the final version of the program.