Practical 1

COS214

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science

Deadline: 29/07/2024 at 23:59

Marks: 250

1 General instructions:

- This assignment should be completed individually.
- Be ready to upload your practical well before the deadline, as no extension will be granted.
- You can use any version of C++
- You can import the following libraries:
  - vector
  - string
  - iostream
  - map

2 Plagiarism

The Department of Computer Science considers plagiarism a serious offence. Disciplinary action will be taken against students who commit plagiarism. Plagiarism includes copying someone else's work without consent, copying a friend's work (even with permission) and copying material from the Internet. Copying will not be tolerated in this course. For a formal definition of plagiarism, the student is referred to http://www.library.up.ac.za/plagiarism/index.htm. **If you have any questions regarding this, please ask one of the lecturers to avoid misunderstanding.**

3 Mark Distribution

Activity	Mark
Total	**250**

Table 1: Mark Distribution

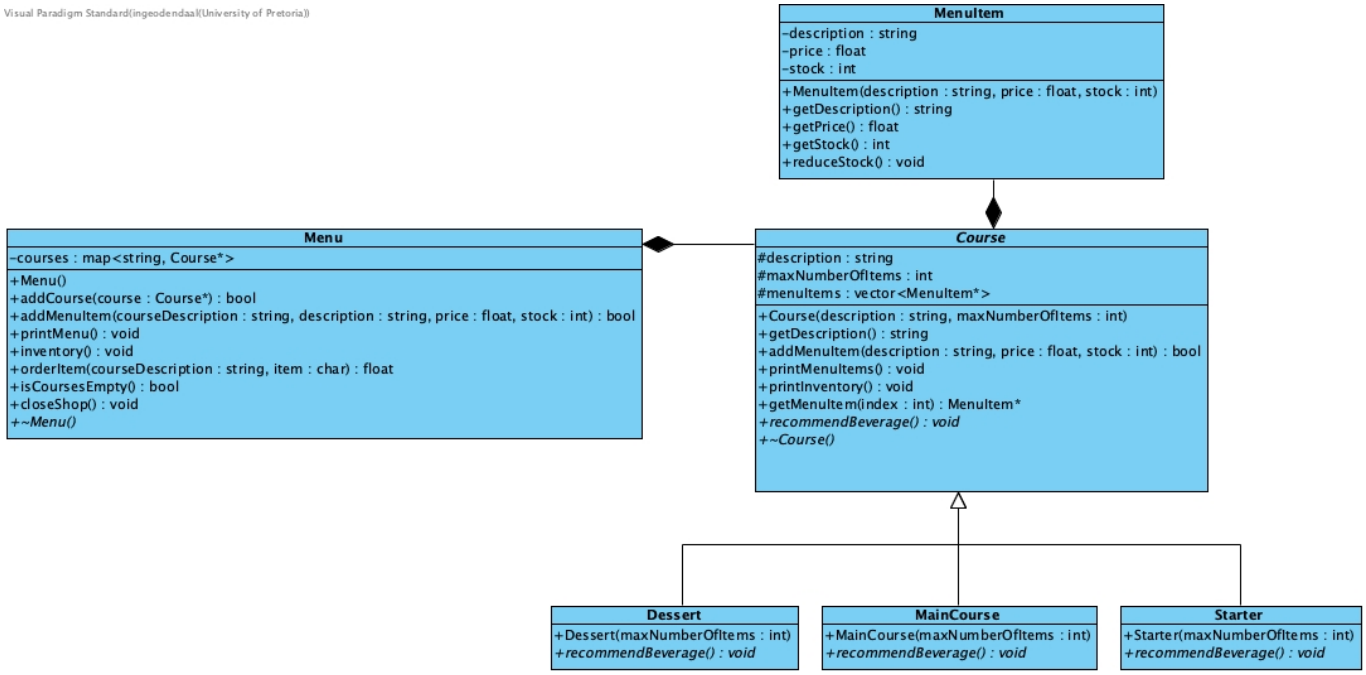
4 Task 1

You have been approached by your aunt who has purchased a license to open a road-side kiosk near the university. She has asked you to help her manage her daily inventory and menu for her kiosk. Each morning she will stock her kiosk and enter the items in the program you are to write for her. She will have a laptop for order management and a second screen where she can display the menu for prospective clients to choose from.

Remember to test your code, to ensure you do not have any memory leaks and to validate input to functions.

We will also test your code for memory leaks.

4.1 UML Class Diagram



4.2 MenuItem

Please make sure to define all functions headers in this section in `MenuItem.h` and implement the functions in `MenuItem.cpp`

- `MenuItem(string description, float price, int stock)`
  This is the constructor for a MenuItem object and should initialise the appropriate variables.

- `string getDescription()`
  Returns the description
- `float getPrice()`
  Returns the price
- `int getStock()`
  Returns the stock
- `void reduceStock()`
  Decrements the **remaining** stock.

4.3 Course

Please make sure to define all functions headers in this section in `Course.h` and implement the functions in `Course.cpp`

- `Course(string description, int maxNumberOfItems)`
  This is the constructor for a Course object and should initialise the appropriate variables.
- `bool addMenuItem(string description, float price, int stock)`
  Attempts to add a new menu item to the menuItems vector. Returns true if it succeeds, false otherwise.
- `void printMenuItems()`
  The `printMenuItems` function outputs the list of menu items for a specific course. Each menu item is displayed on a new line, preceded by a tab character and an alphabetical index (starting from 'a'), followed by a full stop and another tab character, and then the description of the menu item.

  Expected Output:

  For a course with the following menu items:

  – Onion Soup
  – Caesar Salad

  The output should be:

  ```
  <tab>a.<tab>Onion Soup<newline>
  <tab>b.<tab>Caesar Salad<newline>
  ```

  Where `<tab>` and `<newline>` are the respective whitespace characters.
- `void printInventory()`
  The `printInventory` function outputs the detailed list of menu items for a specific course. Each menu item is displayed on a new line, preceded by a tab character and an alphabetical index (starting from 'a'), followed by a dot and another tab character, then the description of the menu item, its price, and its stock level, all separated by tab characters.

Expected Output:

For a course with the following menu items:

- Onion Soup, priced at 35.99, with 6 in stock
- Caesar Salad, priced at 45.99, with 4 in stock

The output should be:

```
<tab>a.<tab>Onion Soup<tab>35.99<tab>6<newline>
<tab>b.<tab>Caesar Salad<tab>45.99<tab>4<newline>
```

Where `<tab>` and `<newline>` are the respective whitespace characters.

- `MenuItem* getMenuItem(int index)`
  Returns the MenuItem at the specified index if one exists, else return nullptr.

- `virtual void recommendBeverage()`
  This is a pure virtual function. It should not be implemented in Course.cpp and should be defined as pure virtual.

- `~Course()`
  The Course destructor (something to consider - should it be virtual?). It should clean up any dynamic memory to avoid memory leaks (does this mean memory on the heap/stack?)

  Note that the composition relationship in the UML means that Course owns its MenuItem(s).

4.4 Starter

Please make sure to define all functions headers in this section in `Starter.h` and implement the functions in `Starter.cpp`

- `Starter(int maxNumberOfItems)`
  This is the constructor for a Starter object. It should call the Course constructor with the appropriate variables and description "Starter"

- `void recommendBeverage()`
  This method prints the recommended beverage followed by a newline. The recommended beverage is "Sparkling Water".

4.5 MainCourse

Please make sure to define all functions headers in this section in `MainCourse.h` and implement the functions in `MainCourse.cpp`

- `MainCourse(int maxNumberOfItems)`
  This is the constructor for a MainCourse object. It should call the Course constructor with the appropriate variables and description "Main"

- `void recommendBeverage()`
  This method prints the recommended beverage followed by a newline. The recommended beverage is "Coke".

4.6 Dessert

Please make sure to define all functions headers in this section in `Dessert.h` and implement the functions in `Dessert.cpp`

- `Dessert(int maxNumberOfItems)`
  This is the constructor for a Dessert object. It should call the Course constructor with the appropriate variables and description "Dessert"

- `void recommendBeverage()`
  This method prints the recommended beverage followed by a newline. The recommended beverage is "Coffee".

4.7 Menu

Please make sure to define all function headers in this section in `Menu.h` and implement the functions in `Menu.cpp`

- `Menu()`
  Default constructor

- `bool addCourse(Course* course)`
  Adds a new course to the menu. The course is added if a course with the same description does not already exist in the menu.

  You should return:

  - `true` if the course was successfully added.
  - `false` if a course with the same description already exists.

- `bool addMenuItem(std::string courseDescription, std::string description, float price, int stock)`
  Adds a new menu item to the specified course with the appropriate parameter values.

  You should return:

  - `true` if the menu item was successfully added.
  - `false` if the course does not exist or the item could not be added.

- `void printMenu() const`
  Prints the descriptions of all courses and their respective menu items in the menu.

  You should output:

- For each course in the menu, prints the course description followed by a newline and a list of its menu items (use the appropriate function in Course).
- Example output:

```
Starters
    a.  Onion Soup
    b.  Caesar Salad
Main
    a.  Steak
    b.  Chicken
    c.  Fish
Dessert
    a.  Ice Cream
```

- `void inventory() const`
  Prints the inventory of all courses and their respective menu items in the menu.

  You should output:

  - For each course in the menu, prints the course description followed by a newline and a detailed list of its menu items, including their price and stock (use the appropriate function in Course).
  - Example output:

```
Starters
    a. Onion Soup    35.99    6
    b. Caesar Salad  45.99    4
Main
    a. Steak         105.99   5
    b. Chicken       95.99    2
    c. Fish          85.99    3
Dessert
    a. Ice Cream     65.99    7
```

- `float orderItem(std::string courseDescription, char item)`
  Orders (or sells) an item from the specified course.

  The Parameters to the functions are:

  - `courseDescription` - The description of the course from which the item should be ordered.
  - `item` - The alphabetical index of the item to be ordered (obtained from printMenu)

  You should return:

  - The price of the item if it was successfully ordered.

- `0` if the item is out of stock or the course/item does not exist.

- `bool isCoursesEmpty()`
  Should return whether courses is empty or not.

- `void closeShop()`
  Closes the shop by deleting all courses and clearing the menu.

  Note that the composition relationship in the UML means that Menu owns its Course(s).

  You should output:

  - Prints a message indicating the shop is closing and the number of courses being deleted.
  - Format:

    `Closing shop. Deleting all <courseSize> courses<newline>`

    Where `<courseSize>` is the number of courses and `<newline>` is the appropriate whitespace character

- `~Menu()`
  Destructor for the Menu class. Ensures that all dynamically allocated memory is freed.

  The output:

  - Prints a message indicating the destructor was called and calls closeShop() if there are any remaining courses.
  - Message Format:

    `Menu destructor called<newline>`

    Where `<newline>` is the appropriate whitespace character

5 Submission checklist

For Task 1:

- Zip all necessary files:
  - Course.cpp
  - Course.h
  - Dessert.cpp
  - Dessert.h
  - MainCourse.cpp
  - MainCourse.h
  - Menu.cpp

  - Menu.h
  - MenuItem.cpp
  - MenuItem.h
  - Starter.cpp
  - Starter.h
- Ensure you are zipping the files and not the folder containing the files.
- Upload to the appropriate slot on FitchFork well before the deadline as no extensions will be granted