C++ & Software design

Assignment

Week 2 - A simple class

Purpose

This exercise provides a simple introduction to object-oriented programming and the use of Dev-C++

Outline

Create a simple class that represents a meeting object to be held as an item in a diary.

The meeting object will have the following attributes :-

- Name of person being met - string
- Time of meeting - hour (int) and minute (int)
- Duration of meeting - minutes (int)
- Location of meeting - string
- Purpose of meeting - string

You should be able to initialise the object using the constructor and display the details with a display function which returns a formatted string ready for display on the screen.

Required

Create the following three components :-

- A header file containing the class definition - meeting.h
- A cpp file containing function implementations - meeting.cpp
- A main which creates a meeting object and then displays it on the screen - main.cpp

Week 3 - Simple aggregation

Purpose

This exercise involves the use of aggregation. Placement of an object inside of another object.

Outline

Using the Meeting class created last week, create a Diary class which should have the following attributes :-

- Owners name
- A single meeting object reference

and should provide the following functionality :-

- A constructor to initialise the created objects
- A method - BOOL addMeeting(Meeting &) - to add a meeting object
- A method - string display() - which displays the meeting object if there is one
- A destructor

Finally create a main method which creates a meeting object, adds it to the diary and displays the contents of the diary.

Week 4 - Day object - assessed coursework

Purpose

This exercise gives further practice in developing new classes and handling arrays of objects.

Outline

Using the classes so far created for the diary, create a Day class capable of holding an array of meetings. The day object should know the date it refers to and be able to return it to client code and (as with the diary itself) should allow for the addition of a meeting object. For the moment disregard the possibility of meetings overlapping.

The Day class should also provide for a display function which will return a formatted string with details of all meetings for the day and in addition provide two new functions - one to return the first meeting of the day (as a meeting *) and the second to return the next meeting. This later function is called repeatedly after the previous one and delivers meetings in sequence until the last meeting has been delivered after which it returns NULL (rather like reading the records from a file).

The diary object should be modified to contain a single day object and should also now support these two new functions.

Finally modify main so that it will prompt repeatedly for the addition of meetings until terminated by entering "end" as the name and then display the meetings for the day, firstly using the display() function and then using the firstMeeting() and nextMeeting() functions.

Submission

This is an assessed piece of work. You should submit your solution on or before 7th November as a zip file attachment to an email to peter.martin@uwe.ac.uk . Once all submissions have been received, you will receive feedback and marks by email and a suggested solution will be posted here.

Deliverables

```
Source code for :-

    day.h
    day.cpp
    diary.h
    diary.cpp
    main.cpp
```

Note: There should be no changes made to meeting.h or meeting.cpp

Week 5 - Menu object - assessed coursework

Purpose

This exercise gives further practice in developing new classes and handling arrays of objects - yet again.

Outline

You may have wished you had some simple way in the last assignment to create a menu to select options for adding meetings, displaying them and exiting the program. You can code this every time of course or if you are using a GUI menus are provided.

Develop, for use with console interface, a Menu class that has the following :-

- An array of strings representing the menu options, numbered 1 to n (maximum of 9)
- A constructor which takes the title of the menu as a parameter
- A method - bool addOption(string option) - to add a new item to the menu
- A method - int select() - which displays the menu, requests user input, checks the input for a valid integer in the correct range, producing an appropriate error message if wrong and re-requesting input and returns the selected index if valid.

You will need to check for valid integer input in case the user presses random characters in the menu selection.

Create a main.cpp which calls select() in a switch statement and displays an appropriate message in a loop.

Submission

This is an assessed piece of work. You should submit your solution on or before 14th November as a zip file attachment to an email to peter.martin@uwe.ac.uk . Once all submissions have been received, you will receive feedback and marks by email and a suggested solution will be posted here.

Deliverables

```
Source code for :-

    menu.h
    menu.cpp
    main.cpp
```

Week 5 - Simple Diary - assessed coursework

Purpose

This exercise gives further practice in developing new classes and handling arrays of objects - yet again.

Outline

The Diary objects created in previous weeks were hardly realistic as they had either a single meeting object inside or a single day containing an array of meetings. This practical attempts to build a more realistic diary having 365/6 day objects built in (depending on leap year or not), each with the capability of holding a variable number of meetings and having the ability to both add and delete meetings.

The new Diary class will provide the following :-

- A constructor specifying the year for the diary as an integer
- An addMeeting function/method which specifies a meeting object and the date of the meeting
- A deleteMeeting function/method which specifies the date and time of the meeting to be removed
- A display function/method which specifies the date and displays all meetings on that day
- A displayFirst function/method which specifies a date and displays the first meeting on that day or returns an empty string if none
- A displayNext function/method which specifies the date and displays the next meeting on that date or returns an empty string if no more

The day class should be revised to provide for the following :-

- A variable number of meeting objects - using a Vector collection class rather than an array
- Meetings should be added in time of day order
- If an attempt to add a meeting starting at the same time as another on any given day, this should not be allowed and an error signalled to the user
- Meetings may overlap but an error message should be displayed to the user

The meeting class should be revised to provide for the following :-

- A function/method to return the start time of the meeting (e.g. in minutes from midnight)
- A function/method to return the duration of the meeting

Submission

This is an assessed piece of work. You should submit your solution on or before 28th November as a zip file attachment to an email to peter.martin@uwe.ac.uk . Once all submissions have been received, you will receive feedback and marks by email and a suggested solution will be posted here.

Deliverables

```
Source code for :-

    diary.h    diary.cpp    day.h    day.cpp    meeting.h    meeting.cpp
    main.cpp
```

Week 6 - Diary - final version

Purpose

This completes the diary project by adding more functionality and introduces the use of inheritance.

Outline

The following changes/additions are required to the previous version of the diary as follows :-

- Instead of meetings, the diary should contain Item objects
- Items should be sub-classed either as meetings or as Events
- Events simply have a description - e.g. an anniversary or a reminder note
- When adding an Item, it should be possible to specify the periodicity, i.e.:-
    - Once only
    - Repeated every week
    - Repeated every two weeks
    - Repeated every four weeks

Submission

This is an assessed piece of work. You should submit your solution on or before 12th December as a zip file attachment to an email to peter.martin@uwe.ac.uk . Once all submissions have been received, you will receive feedback and marks by email and a suggested solution will be posted here.

Deliverables

```
Source code for :-

    diary.h
    diary.cpp
    day.h
    day.cpp
    item.h
    item.cpp
    event.h
    event.cpp
    meeting.h
    meeting.cpp
    main.cpp
```

Practical 6 - hints

In the final version of the Diary, the Day objects contain a collection of Item objects rather than Meeting objects.

These Item objects may actually be Meeting objects or Event objects.

Some important considerations :-

- Item is the super class from which the other two are derived.
- Meeting is a subclass of Item and inherits from it any methods and attributes defined there
- Similarly, Event is a subclass of Item
- The Item class may have no attributes unless you can find one that is common to both Event and Meeting, which both can inherit.

- The item class MUST define methods - display(), displayFirst() and displayNext() and They MUST be defined as e.g. :-
- `    public:`
- `        virtual string display()`
- `        etc`

  in Item. The word virtual must not appear in the definitions in Event and Meeting.

- These methods/functions may be empty if there are no attributes in the Item class or they may contain some code if you have found some common attributes.
- The Diary will add and delete Items rather than Meetings.
- The only place where you need to make a decision to add an Item or a Meeting is in main, the code/pseudocode for this could be like :-

```
Diary myDiary;
Item * newItem;
if adding a meeting then
  newItem = new Meeting(.......);
else if adding an event then
  newItem = new Event(.......);
endif
myDiary.addItem(newItem);
```