ACS223 Computer Systems and Applications
Semester 1 C++ Assessment (ACS223-002)

James Gu (james.gu@sheffield.ac.uk). Room D13, AJB

Assignment weighting: 5% of module mark

Assignment released: 31/10/2017 (Semester 1, Week 6)

Assignment due: 13/11/2017 (Semester 1, Week 8)

How to submit

The assignment must be submitted to MOLE by **8am on the 13/11/2017**. The assignment dropbox link is in ACS223 MOLE -> Coursework Content-> 'Assignment Submissions' folder, and is entitled "Semester 1 C++ Assessment – ACS223-002". You must submit all of the .cpp and .h files which you create for this assignment at this link.

Marking and Feedback

The assignment will be marked by appointment during the timetabled lab session on 13/11/2017. The GTA/module leader who marks the assessment in the lab will give the overall mark, individual component marks and comments on performance on the program. The electronic submission will be reviewed after the lab session, by the module leader. The attached marksheet provides a guide to the areas on which feedback will be provided.

Learning outcomes

By doing this assignment, students will

- Develop skills in following set requirements to produce a solution to a problem
- Develop skills in C++ programing, specifically with objects and classes

Assignment briefing

Your task is to write a C++ program to help a UK customer plan the heating requirements and costs for an office building: the program allows the customer to pick different rooms which will be heated, the number of days in which heating is required, the number of hours in which heating is required per day and the average temperature to heat the room by.

The program then calculates the cost of upgrading the energy efficiency rating, the savings associated with upgrading and the payback period. Three energy efficiency savings are compared.

This program assumes that the cost of increasing the temperature of the room is linear (i.e. increasing the temperature by $2^{o}C$ costs twice as much as $1^{o}C$).

Your code must run on CodeBlocks.

The requirements for the program are as follows (please also refer to the marking scheme to see what other things will be looked at during the marking):

1. The user shall be able to determine the total **annual** cost of heating an office building.
2. The user shall be able to select a room types from a table which are in use in the building under consideration.
   2.1. The program shall input from a file containing a table of room types with typical heating costs. This information shall be stored internal to the program (i.e. in a suitable data structure or data object).
   2.2. The program shall allow the user to enter a number (corresponding to the type of room) in order to determine the size of room.
   2.3. The program shall use 'pass by reference' arguments in an appropriate method so as to record the type of room and its associated heating cost.
3. The user shall be able to enter an estimate for the number of hours each day the heating will be required for each individual room.
   3.1. The program shall allow the user to enter values that are less than 1 to represent a portion of an hour (eg, users can enters 0.5 to indicate half an hour).
4. The user shall define the number of days per year which heating is required and the average temperature in degrees Celsius) which the room is heated by
   4.1. The program shall allow the user to enter the temperature increase to 1 decimal place (in degrees Celsius).
   4.2. The program shall then display the total estimated cost of heating in the building for a year.
5. The user shall have the option be able to manually set the cost of heating or be chose the default value of £0.015/^{o}C/m^{2}/hr.
6. The program shall allow the user to compare savings to the heating cost by upgrading to **three possible building energy efficiency ratings**
   6.1. The program shall allow the user to select from 6 efficiency ratings: A,B,C,D,E and F.
   6.2. The program shall use an array of objects to create 3 objects of the same class (where each object refers to a scenario).
7. The program shall calculate the total cost of upgrading the building energy efficiency rating.
   7.1. The program shall store the cost of upgrading, the annual saving and the payback period in a suitable array.
   7.2. To upgrade to grade F the cost will be £3.20/m^{2} and will reduce heating costs by 5%.
   7.3. To upgrade to grade E the cost will be £12.75/m^{2} and will reduce heating costs by 10%.
   7.4. To upgrade to grade D the cost will be £32.20/m^{2} and will reduce heating costs by 15%.
   7.5. To upgrade to grade C the cost will be £45.33/m^{2} and will reduce heating costs by 20%.
   7.6. To upgrade to grade B the cost will be £77.80/m^{2} and will reduce heating costs by 25%.
   7.7. To upgrade to grade A the cost will be £143.50/m^{2} and will reduce heating costs by 33%.
8. The program shall display the three scenarios for calculating the cost of heating an office building in the UK.
   8.1. For each scenario, the program shall display: the cost of heating before improving on energy efficient rating, the cost of upgrading the building energy efficiency rating and payback period of the upgrade (to the nearest day).
   8.2. The program shall display the information for each scenario required in a table.
   8.3. The program will display the best efficiency upgrade based on the shortest payback period.
9. The program shall have guards against incorrect user inputs.

HINT1: For information of how to create an array of classes/objects, refer to page 18 of the C++ notes where an example of creating and using an array of objects myAcct[3] of class bankAccount is given.

** The input text file is given and can be found on MOLE in Course Work section.
Figure 1 shows the services to be provided by the software, via a UML use case diagram.

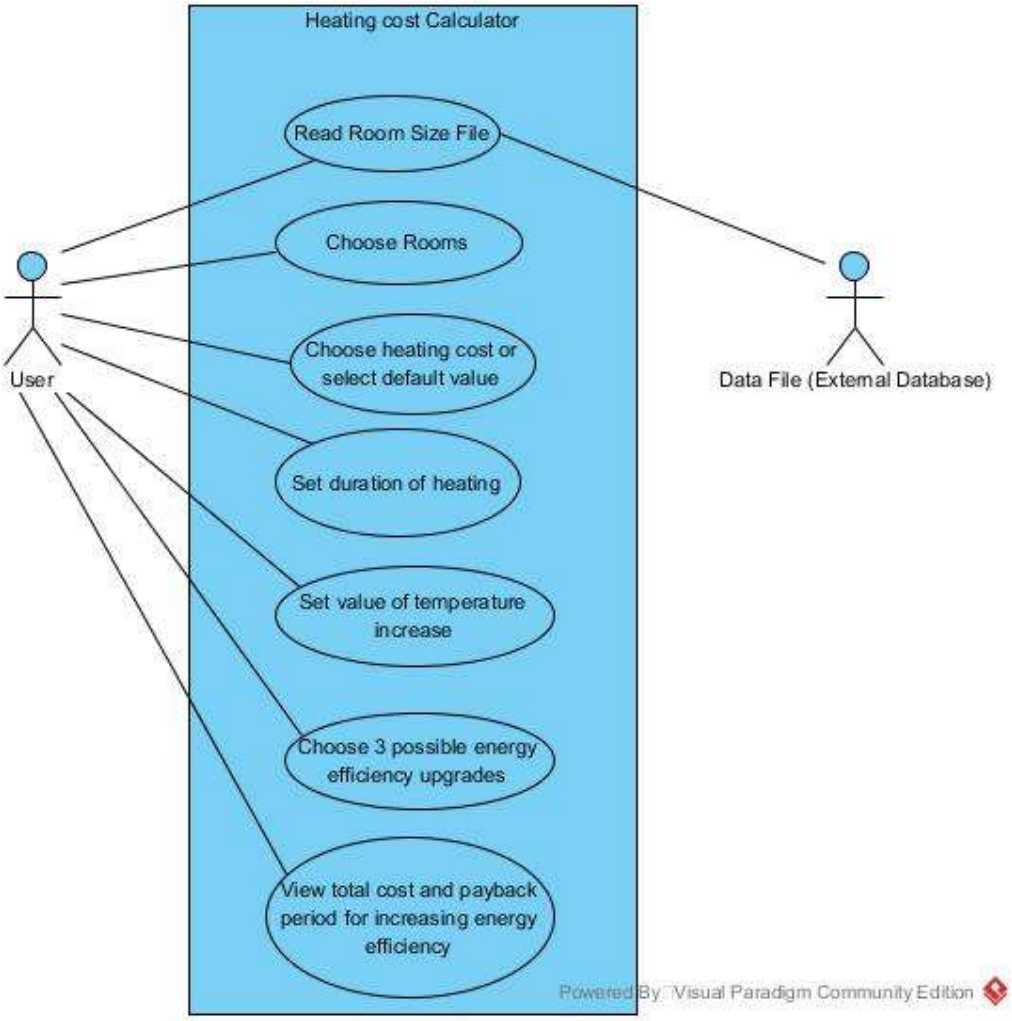

Figure 1. UML use case diagram to show the services provided by the software

Figure 2 shows a sample structure for the software, via a skeletal UML class diagram. Note: you must have at least a main program (main.cpp – represented as the controller class in my class diagram), and at least two classes shown in the class diagram in Figure 2.

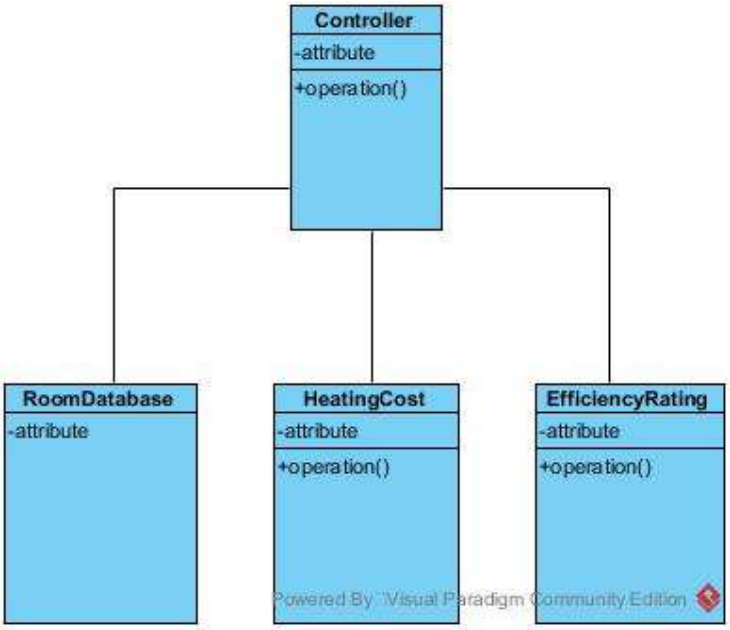

Figure 2. Skeletal class diagram for the software. (You do not need to break down the problem like this).