CS2810 OOAIA: L1

January 21, 2025

Objective

- To learn basic OO concepts and understand some issues that we would address during the course.
- To learn good software development practices.

Data Structures

Create two classes in C++: `BTech` and `DualDegree`.
They have the following (approximate) interface:

- `BTech(RollNumber r, string hostel, Faculty facad, float cgpa);`
- `string getHostel();`
- `Faculty getFacad();`
- `void changeHostel(string);`

In addition, BTech class contains the following methods:

- `Faculty getBTPGuide();`
- `setBTPGuide(Faculty);`

DualDegree class contains the following methods:

- `Faculty getDDPGuide();`
- `Faculty getTASupervisor();`
- `setDDPGuide(Faculty);`
- `setTASupervisor(Faculty);`

`Faculty` class contains a unique name as string and a boolean which indicates if the Faculty is permanent or Adjunct. You can add other fields and methods as appropriate, change arguments to functions as necessary.

Initialization

Given an `IITM` class which consists of a vector of N students (BTech and DualDegree) and a vector of all M Faculty, provide the following functionality in main (N <= 10^5, M <= 100, number of hostels <= 10).

- create students and faculty members randomly
- assign facads to students randomly
- assign BTP guides to BTech students by CGPA order. Only a permanent faculty can be a guide (not Adjunct). No guide should have more than 2 BTech projectees.
- assign DDP guides to DualDegree students ordered by hostel name. Only a permanent faculty can be a guide. No guide should have more than 2 DD projectees.
- assign TA supervisor to DualDegree students randomly. This can be either permanent or adjunct faculty. No supervisor should have more than 5 TAs. Assume that each faculty teaches only one course.

Algorithms

STEP 3: Allow the following functionality in `IITM` class:

- Write functions for checking if the constraints are satisfied (number of projectees, number of TAs).
- find percentage of students whose DDP guide and TA supervisor are the same.
- print students sorted by (i) roll number (ii) guide name (iii) hostel.
- print all the students who could not be assigned a guide.
- print all the DD students from a given hostel working under a given faculty member either for project or for TA duty.
- find the faculty member who has projectees (BTech + DD) in maximum number of different hostels.

STEP 1: Before implementing, create a design document:

- List all the important classes, and their interfaces.
- Comment important fields / methods with their expected functionality.
- When a method m1 of class C1 is supposed to call C2.m2, add that comment for both m1 and m2 or add an arrow from m1 to m2.
- Mark all the methods that perform I/O (file or screen or network or …) with a color (say, purple).
- Mark all the (likely) time-consuming methods with a color (say, red).
- Mark all the (likely) memory-hungry methods with a color (say, blue).

STEP 2: Before implementing, create a test document:

- L1: List the basic scenario that you would like to test.
- L2: Now list exceptional scenarios / corner-cases which you should test.
- L3: List scenarios where the program should produce error messages.

- L4: List scenarios where the program may misbehave (segfault or infinite loop).

STEP 3 Procedure: During implementation:

- Create a compilable interface of the classes, before adding code to the methods.
- Now add cout to all the methods, implement main, do not implement class methods, and make sure the methods are called in the expected order when you compile and execute the program.
- Now implement it for L1. Compile and test.
- Now implement it for L2 and L3. Compile and test.
- Make sure L4 is empty.

Some suggestions:

- Keep each class in a separate file.
- Move your code to github. If you don't have an account, create one. Add a link to it in your CV / biodata. Keep all your projects there.
- Create a `README.md` file.
- Use `makefile` or `cmake`.
- Distribute code in separate directories: doc, src, include, bin, test (even if the project is small).
- Discuss with the instructor / TAs about good practices.