Introduction

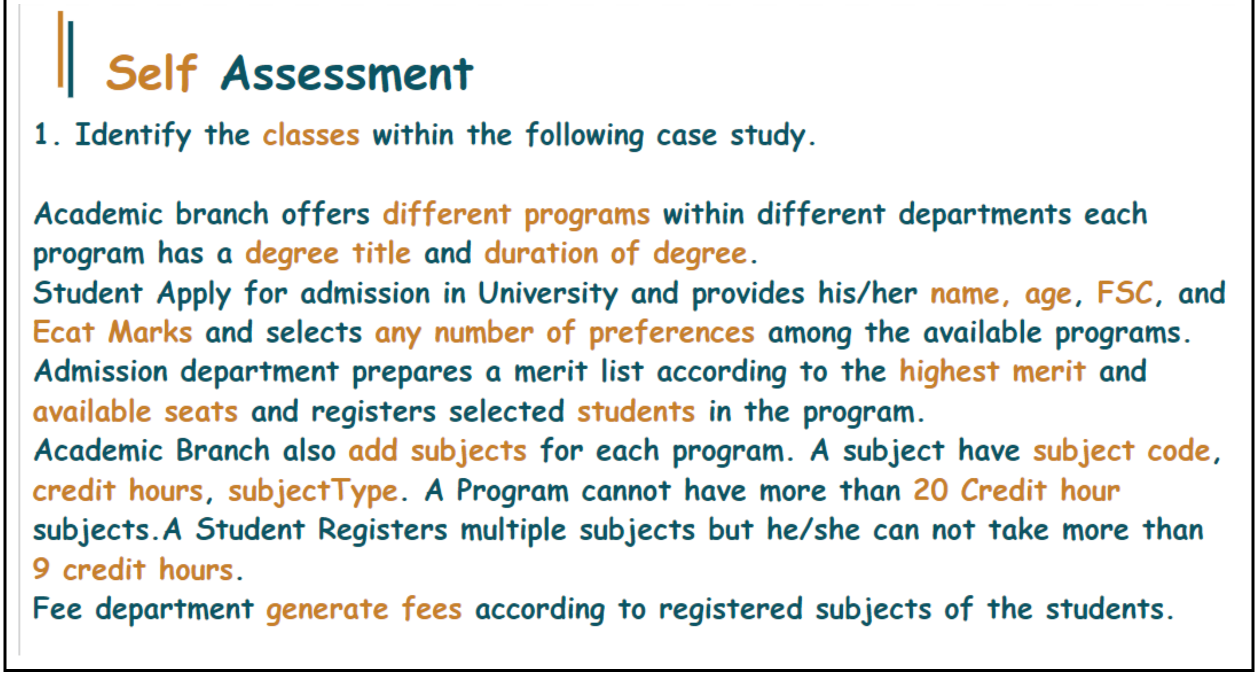
After a week of rigorous coding, Welcome back!

You have learned all about the Classes, Constructors, and member functions in the previous lab manuals. Let's move on to the next, new, and interesting concepts.

Students, In Object-Oriented Programming, the Class is a combination of data members and member functions. In this Lab, we will learn about creating separate BL, DL and UI for our Applications.

University Admission Management System (Case Study with Separate BL, DL and UI)

Read the following question carefully.



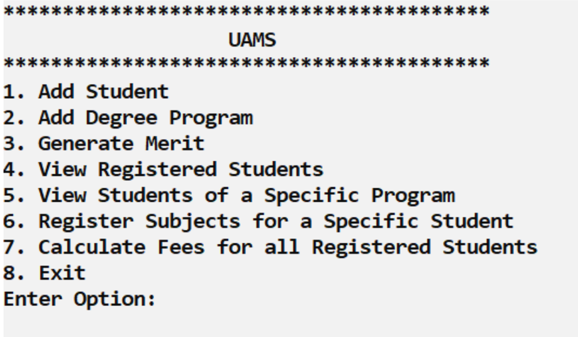
Self Assessment

1. Identify the classes within the following case study.

Academic branch offers different programs within different departments each program has a degree title and duration of degree.
Student Apply for admission in University and provides his/her name, age, FSC, and Ecat Marks and selects any number of preferences among the available programs.
Admission department prepares a merit list according to the highest merit and available seats and registers selected students in the program.
Academic Branch also add subjects for each program. A subject have subject code, credit hours, subjectType. A Program cannot have more than 20 Credit hour subjects.A Student Registers multiple subjects but he/she can not take more than 9 credit hours.
Fee department generate fees according to registered subjects of the students.

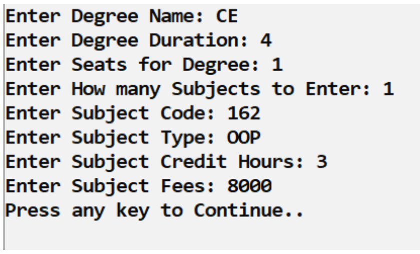
Wireframes of the UAMS:

1. Main Menu



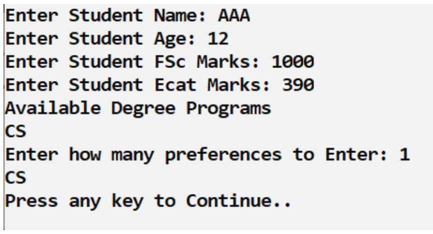
```
******************************************
                   UAMS
******************************************
1. Add Student
2. Add Degree Program
3. Generate Merit
4. View Registered Students
5. View Students of a Specific Program
6. Register Subjects for a Specific Student
7. Calculate Fees for all Registered Students
8. Exit
Enter Option:
```

2. Option 2: Degree Program



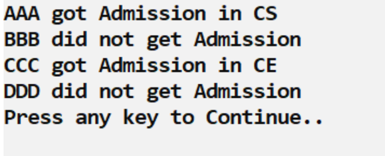
```
Enter Degree Name: CE
Enter Degree Duration: 4
Enter Seats for Degree: 1
Enter How many Subjects to Enter: 1
Enter Subject Code: 162
Enter Subject Type: OOP
Enter Subject Credit Hours: 3
Enter Subject Fees: 8000
Press any key to Continue..
```

3. Option 1: Add Student



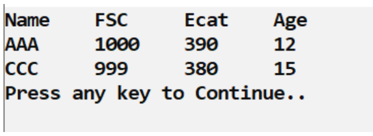
```
Enter Student Name: AAA
Enter Student Age: 12
Enter Student FSc Marks: 1000
Enter Student Ecat Marks: 390
Available Degree Programs
CS
Enter how many preferences to Enter: 1
CS
Press any key to Continue..
```

4. Option 3: Generate Merit



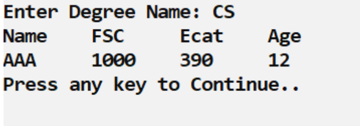
```
AAA got Admission in CS
BBB did not get Admission
CCC got Admission in CE
DDD did not get Admission
Press any key to Continue..
```

5. Option 4: Registered Students



```
Name    FSC     Ecat    Age
AAA     1000    390     12
CCC     999     380     15
Press any key to Continue..
```

6. Option 5: Specific Degree



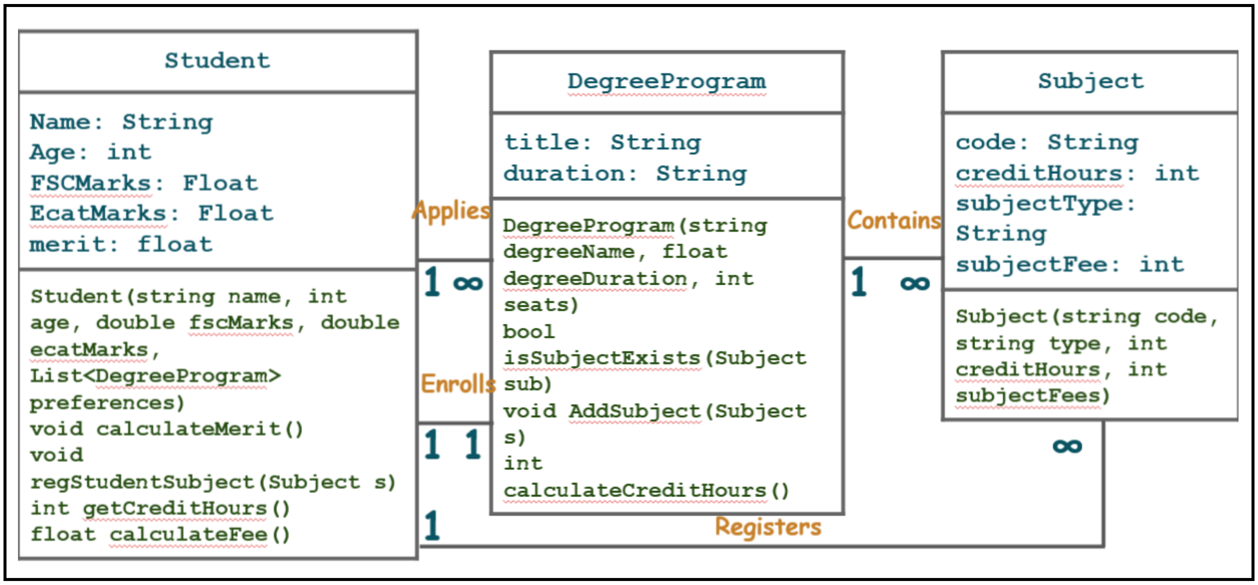
```
Enter Degree Name: CS
Name    FSC     Ecat    Age
AAA     1000    390     12
Press any key to Continue..
```

7. Option 6: Register Subjects

Ask the Student name and then ask for the subject code. If the conditions are satisfied then the student's subject should be registered.

8. Option 7: Generate Fee
   Fees should be generated for all the registered students

Class Diagram with the member functions


Student
Name: String
Age: int
FSCMarks: Float
EcatMarks: Float
merit: float
Student(string name, int age, double fscMarks, double ecatMarks, List<DegreeProgram> preferences)
void calculateMerit()
void regStudentSubject(Subject s)
int getCreditHours()
float calculateFee()
Applies
1
∞
Enrolls
1
1
1
Registers
DegreeProgram
title: String
duration: String
DegreeProgram(string degreeName, float degreeDuration, int seats)
bool isSubjectExists(Subject sub)
void AddSubject(Subject s)
int calculateCreditHours()
Contains
1
∞
Subject
code: String
creditHours: int
subjectType: String
subjectFee: int
Subject(string code, string type, int creditHours, int subjectFees)
∞

Following are the Tasks related to this problem.

Task 01:

1. Create a separate BL class for Subject, DL class for storing the list of subjects, and UI class for handling all the input output functions related to subjects.
2. Create a separate BL class for DegreeProgram, DL class for storing the list of DegreePrograms, and UI class for handling all the input output functions related to DegreePrograms.
3. Create a separate BL class for Student, DL class for storing the list of students, and UI class for handling all the input output functions related to Students.
4. If there are some UI functions that do not go into the UI of Subject, DegreeProgram or Student then add a separate UI class with name **ConsoleUtility.**
5. Write the Driver Code (main function) for implementing the menu driven program.

You have made it through all that. Excellent work students !!!
You guys are successfully en route to be Kamyab Programmers.

There are a few challenges given ahead. Have a try at those. Good Luck :)

Challenge 01:

Miss Client wants to develop a software system for her departmental store. She wants this system to have the following functionalities.

As an Admin, she can

- Add Products.
- View All Products.
- Find Product with Highest Unit Price.
- View Sales Tax of All Products.
- Products to be Ordered. (less than threshold)

Following is the information that is required to save for the product.

Name of Product. Product Category. Product Price. Available Stock Quantity. Minimum Stock threshold Quantity after which the owner wants to order the product.

On All Grocery type of products, the sales tax is 10%, on all fruit types the tax is 5% and if there is any other type the tax is 15%

Customer information includes username, password, email, Address and Contact Number.

She also wants that

1. The Customers to view all the products
2. Customers can buy the products (When a customer buy a product then its quantity should decrease from the stock)
3. Generate invoice (While calculating the price of the products that the customer has bought, sales tax should be applied.)

Make 3 classes

1. Product
2. Customer
3. Admin (MUser) that we have previously developed with file handling

Menus on the Console.

1. SignIn
2. SignUp

3. Exit

If the user enters 1 then

Enter Username: AAA
Enter Password: 111

(if the user is valid and admin then show the admin menu)

1. Add Product.
2. View All Products.
3. Find Product with Highest Unit Price.
4. View Sales Tax of All Products.
5. Products to be Ordered. (less than threshold)
6. View Profile (Username, Password)
7. Exit

(if the user is valid and customer then show the customer menu)

1. View all the products
2. Buy the products
3. Generate invoice
4. View Profile (Username, Password, Email, Address and Contact Number)
5. Exit

Implementation Notes:

1. Make BL, DL and UI Class for Admin.
2. Make BL, DL and UI Class for Customer.
3. Make BL, DL and UI Class for Product.
4. If there are some UI functions that do not go into the UI of Admin, Customer or Product then add a separate UI class with name **ConsoleUtility.**

Challenge 02:

In this programming exercise, you are tasked with designing a simplified version of Twitter and Facebook.

Classes:

1. **User**

Data Members:

- password: A password string.
- messages: A list of stored messages.
- user: A field to store the username.

Member Functions:

- **constructor(user, pass):** Initializes a User with a username and password, and messages are initialized as an empty List.
- **encryptPassword(str):** A method to return the encrypted password using the following formula.
  - Divide the input str into 3 equal parts, if the str is not equally divisible by 3 then add # at the end of the str until its size gets divided by 3.
  - Concatenate the first part at the start of the password, second part at the middle of the password (round off to integer part incase of odd size password) and third part at the end of the password.
  - For Example, if the password is “123” and the str is “HELLO”, then add 1 # at the end. Final str becomes “HELLO#”. Now, the final encrypted password becomes “HE1LL23O#”
- **checkPassword(candidate):** Checks if the provided string matches the password and returns either true or false.
- **getPassword():** A function that returns number of asterisks (****) according to the size of the original password.
- **changePassword(newPassword):** A function that updates the password to a new password.
- **AddMessage(message):** Adds a message to the list of messages.
- **getMessage(index):** Returns a message stored at the specific index. Return invalid in case the index is not correct.
- **deleteMessage(index):** Deletes the message on a specific index.
- **deleteAllMessages():** Deletes all messages.

2. **Twitter:**

Data Members:

- **users:** A List to store instances of TwitterUser.

Member Functions:

- **constructor():** Initializes an empty list of users.
- **addTwitterUser(user):** Adds a User to the list of users.
- **getTwitterUserByUsername(name):** Retrieves a TwitterUser instance based on the provided username or returns null if the user is not in the list.
- **getTweets(user):** Retrieves the messages of the user the user is available.

3. **Facebook:**

Data Members:

- **users:** A List to store instances of FacebookUser.

Member Functions:

- **constructor():** Initializes an empty list of users.
- **addFacebookUser(user):** Adds a User to the list of users.
- **getFacebookUserByUsername(name):** Retrieves a FacebookUser instance based on the provided username or returns null if the user is not in the list.
- **getPosts(user):** Retrieves the messages of the user if the user is available.

Menu Options

```
Welcome to Social Media Simulator!

1. User Operations
2. Twitter Operations
3. Facebook Operations
4. Exit

Enter your choice: _
```

```
User Operations:

1. Create a new user
2. Log in (existing user)
3. Exit

Enter your choice: _
```

```
Valid User
1. View Profile
2. Encrypt Password
3. Check Password
4. Change Password
5. Get Password
6. Log Out

Enter your choice: _
```

```
Twitter Operations

1. Add TwitterUser
2. View Tweets for a Specific User
3. Post Tweet for a Specific User
4. Back to main menu

Enter your choice: _
```

```
FaceBook Operations

1. Add FacebookUser
2. View Posts for a Specific User
3. Add a Post for a Specific User
4. Back to main menu

Enter your choice: _
```

Implementation Notes:

1. Make a BL Class for User, DL class for all the available users in the system, and UI class for all the input and output functions of the user.
2. Make BL and UI classes for Twitter and Facebook. Do not need to make a DL class for these.
3. Implement the Driver Program (Main Function) for the above menus. Make sure to make an instance of Twitter and Facebook at the start of your program.

Good Luck and Best Wishes !!

Happy Coding ahead :)