Namal Institute Mianwali

Computer Science Department

Object Oriented Programming

SP 21

Document:	Solution to Project	Date:	27-July-2021
Prepared By:	**ATTIQ ULLAH SHAH (BSCS-2020-21)** **MUHAMMAD FARHAN (BSCS-2020-40)** **SAID ANWAR (BSCS-2020-58)** **MUHAMMAD IDRESS (BSCS-2020-07)**		
Prepared For:	**SIR SADIQ ULLAH** **SIR SAQIB**		
Document Details:	This document contains solutions to the project assigned in June 5, 2021, for the said course of OBJECT ORIENTED PROGRAMMING.		

OOP PROJECT

COURSE MANAGEMENT SYSTEM:

For our semester project for the said course of Object Oriented Programming we have selected our own project of course management system.

INTRODUCTION:

As we are moving forward with technology, teachers are utilizing the internet as a means to communicate with their students.

Many of the online educational tools are available in which the instructors are doing their jobs effectively. But there come certain problems while using these tools or soft wares. For example, many of the instructor only have knowledge

about using word documents, printing their class materials like syllabus, date sheet etc. Additionally, computer literacy among educators does not typically include programming experience. For these reasons, a program or software is needed that is both customizable and easy to use.

In this project our main focus is on the management of courses for Instructor and Student. Using this program courses can be registered to the students of some institute and also courses can be assigned to the instructor for teaching. This program will be easier and simple to use.

In every college, institute or university courses are assigned to instructor for teaching. Also courses are register for students to be studied.

CONCEPTS OF OOP USED IN OUR PROJECT:

Following concepts of OOP are used in our project:

1. INHERITANCE:

In the programming world, duplicate code is considered evil. We should not have multiple copies of the same, or similar, code in different places. Inheritance is a mechanism that allows code reusability (removes duplication of code) in our programs. In inheritance the common code is put in a class called the **Base** class or **Super** class or **Parent** class and the specific code is put in classes called the **Sub** classes or **Child** classes. All the methods and variables in the superclass become available to the sub classes except the private variables.

In our code we have one parent class named as Person and two child classes named as Student and Instructor which are inherited from base class Person.

2. AGGREGATION:

Aggregation is stronger relationship than association.

In aggregation, one class contains an abject of another class, but in such a way that both objects have their own lifetime. Aggregation is called a **"has a"** relationship. It is represented with an empty diamond box near the containing object.

In our project we have used object of course class in Student or Instructor class as aggregation.

3. COMPOSITION:

Composition is the strongest relationship between classes. It is called a **whole/part** relationship. In composition one class contains an object of another class in such a way that the life of the part depends on the life of the whole. When the whole is destroyed, the part also get destroyed.

4. METHOD OVERRIDING:

Overriding is a mechanism through which a class changes the implementation of a method provided by one of its base classes. Through method overriding a class may "copy" another class, avoiding duplicated code, and at the same time enhance or customize part of it. Method overriding is thus a strict part of the inheritance mechanism.

In our project we have over ride method of Display().

5. EXCEPTIONAL HANDLING:

Sometime a program may be totally correct syntactically, still an error can occur when the program is executed

Errors that occur during the execution of a program is called an exception

For example: Dividing something by zero, converting a string to an integer, opening a file that does not exists etc.

Usually when an exception occurs, the execution of the program is terminated by python and the code that comes after the point where the exception has occurred is not executed. However, this is not desirable. We should be able to handle the exception instead of abnormal termination of the program.

This is called exception handling. Exception Handling is done using the *try-except* Block. Code that has the potential of giving an exception is put inside

the **try** clause. Code that is used to handle that exception is put inside the **except** clause.

CODE EXPLANATION:

Our program has four classes:

1. Person
2. Student
3. Instructor
4. Course

Student and Instructor both are Inherited from the Person class. The Person class contains common attributes such as name, age, gender etc. The Student class and the Instructor class contains attributes that are specific to them such as regNo, regcourses, department, semester etc. in case of Student and experience, salary, assignedcourses etc. in case of the Instructor. Also, both the Student class and the Instructor class holds a list of course objects. Here a concept of aggregation is used in our code.

In case of the student class, the list is a set of registered courses by the student. While in case of the Instructor, the list is a set of objects of courses assigned to him/her for teaching.

The Course class contains information such as coursecode, title, credits, grade, etc.

PERSON CLASS FUNCTION:

Person class contain only one function of Display() which will be used to display attributes of object from person class.

COURSE CLASS METHODS:

This class only contains attributes related to a course such as course code, title, credits, grade. Also this class contains only one method of Display() which will display attributes of course class that is course code, title, grade, credits.

STUDENT CLASS METHODS:

Display():

This function displays all personal attributes of the student including those in the parent class by calling the display method of the parent class.

Add_Student():

In this method Student can be added to our Student_List. This method will ask about name, age, gender, department, Semester and courses that should be registered by the student. After that an object will be created of Student class and it is stored in a list that we have defined as **Student_List**.

Register_Courses():

This method is used to register a course or courses to some specific student. This method will first ask about registration number of a student and this registration number will be searched in a list named as Student_List. If the input registration number is available in a list then for that student, course details like course code, course title, course credit and course grade will be asked and that course will be assigned to the student.

Set_Grade():

This method will be used to set grade of some specific student. First this method will ask about name of a student and that name will be searched in a Student_List. After that this method will ask the user to enter course code and then this course code will be search in the list of registered courses. If the course exists, it will ask to enter the grade (grade is an attribute of the course class) for that course and set it. Otherwise, it will display a message saying “Sorry No such course is Available with such Course Code”.

Search_Student():

This method will be used to search for a student by giving registration number of a student. This registration number will be searched for a Student_List, if registration number is found in a list then details of that student will be displayed on a screen. If registration number is not found in a list, then it will display “Sorry no Student with this Registration Found”.

Remove_Course():

This method will unregister a course from the list of registered courses. First this method will search for a student by taking registration number. After that this method will ask about course code and course will be removed from course list of Registered Course.

INSTRUCTOR CLASS FUNCTIONS:

Display():

This method of instructor class will display details of instructor like name, age, gender, salary, experience. Also it will print attributes of parent class **PERSON** that is inherited in Instructor class. This function also displays all personal attributes of the instructor including those in the parent class by calling the Display() method of the parent class.

Add_Instructor():

In this method instructor can be added to our list. This method will ask about name, age, gender, experience, salary and courses that should be assign to the instructor. After that an object will be created of instructor class and it is stored in a list that we have defined as **Instructor_List**.

Assign_Course():

This method should assign a course to an instructor. In this method, first a name of instructor will be asked for which we have to assign a course. If the instructor name is available in our instance list, then it will ask about

course detail to be assign to the said instructor. So, in this way course will be assigned to the Instructor.

Search_Instructor():

In this method, user can search for the details of an instructor. This method will ask about name of Instructor. The input name will be search in Instrcutor_List, if name is present in Instructor_List then details i.e. name, age, gender, experience, salary and courses assigned of that instructor will be displayed on a screen otherwise it will display sorry.

Remove_Course():

What if we have assign wrong course to some instructor. For that we have introduced a method called **Remove_Course()**. First this method will ask for the name of instructor then this method should accept course code and remove that course from the instructor's courses list. If the course does not exist in the list, then this method should display a message saying "Sorry No course found".

UML DIAGRAM:

UML stands for Unified Modeling language. UML is a graphical language for modeling computer programs. Modeling means to create a simplified representation of something while hiding unnecessary details, like a sketch for a house. UML provides a way to visualize the higher-level organization of programs without going into much detail.

WHY WE NEED UML:

It is hard to understand by simply looking at the code that how the parts of the program are related to each other.

It would be nice if we were able to see a bigger picture of the program. One that shows the major parts of the program and how they work together. The UML is the answer.

So that is why we have to use UML diagram for better understanding.

The UML Diagram for our project is given below: