**CS-212 Object Oriented Programming**

**Assignment # 1**

|  |  |  |  |
| --- | --- | --- | --- |
| Applications of Applications of OOP in real life | | | |
| Maximum Marks: 30 | | Instructor: **Muhammad Khurram Shahzad** | |
| Submission Date: 19th November 2021 | | Type: Written/Individual/LMS/Hardcopy | |
| Name: Muhammad Umer | Reg. #: **345834** | Degree: **BEE** | Section: **12C** |

**Instructions:**

* You need to implement the concepts using skills/concepts learned in this course.
* You can get internet help, but assignments are individual
* You can submit on LMS before due date and submit hard copy in the class
* Submission after due date get zero marks

Write a simple program showing application of friend function, constructors, and objects

**A1.** Design (Example is Lecture 8 | Inheritance | Slide 62) *(10 marks)*

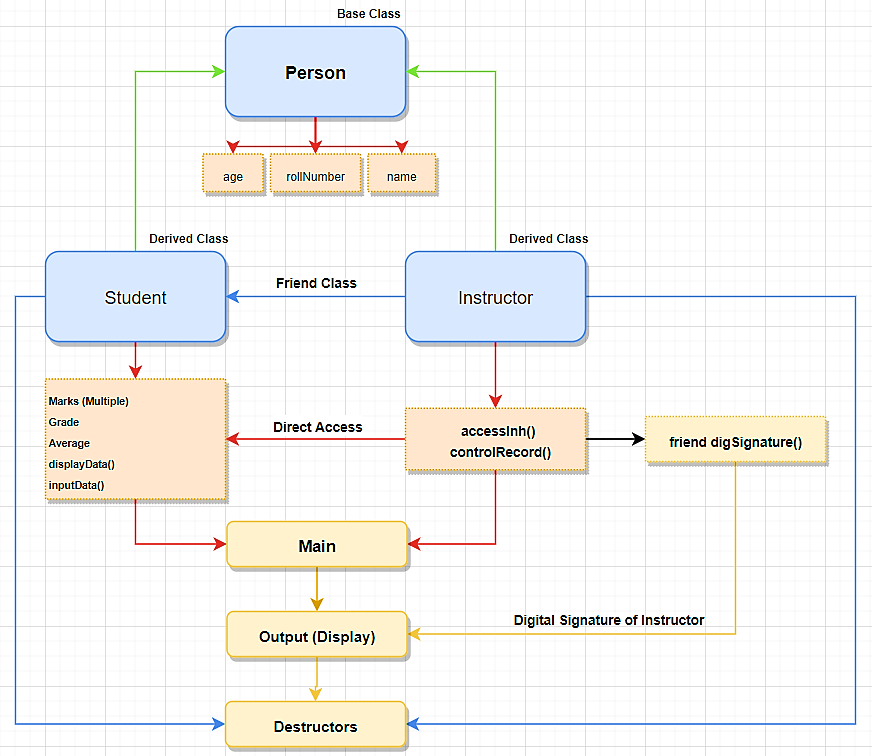
**Statement**

*Designing a Grading & Result Display System pertaining to an individual’s (student) performance in his/her exams, utilizing the concepts of Object Orinted Programming and specifically the use of friend functions, constructors, and objects to manage the flow of data.*

**Design**

* **Classes:** Three classes; **Person, Student, and Instructor**. Both Student and Instructor inherit the characteristics of the common base class *Person*.The Instructor is made the friend class of Student, such that it may access the private members of the Student class freely, which is then later used to update an individual’s marks.
* **Constructors:** A default constructor is made in the Student class to initialize the values of data members to the lowerst value, contextually. A parametrized constructor is also provided for testing purposes to assign grades to the Student in the main().
* **Destructors:** A destructor that prints whenever the scope is out of that specific class. Since we initialize two classes, we should expect two such messages.

The following block diagram sums up the layout of the code; utilization of classes, friend functions, constructors, on a collective basis.

****

**Block Diagram**

**A2.** Write the program, what kind of application you select is up to your imagination. Copy code and output screen shot here *(20 marks)*

**Code**

**// Student Grade & Record System**

**// Importing libraries**

**#include <iostream>**

**#include <string>**

**using namespace std;**

**// Common Class to both Student and Instructor**

**class Person {**

**protected:**

**string name;**

**int age, rollNumber;**

**public:**

**void setInfo();**

**};**

**// Scoped Function of Person Class**

**void Person::setInfo(){**

**cout << "\n\tInput a Name: "; // Prompt**

**getline(cin, name);**

**cout << "\tInput an Age: "; // Prompt**

**cin >> age;**

**cout << "\tInput an ID: "; // Prompt**

**cin >> rollNumber;**

**cout << "\t\t Successfully Assigned Data!\n";**

**}**

**// Forward Declaration**

**class Instructor;**

**// Constructing Student Class with inherent properties of Person**

**class Student : public Person {**

**private:**

**int huMarks, mathMarks, eeMarks, csMarks;**

**double average;**

**char grade;**

**public:**

**// Default constructor that initializes data to their minimum values**

**Student() {**

**huMarks = 0;**

**mathMarks = 0;**

**eeMarks = 0;**

**csMarks = 0;**

**grade = 'F';**

**}**

**// Parametrized Constructor (for hard coding the results)**

**Student(int a, int b, int c, int d, char f) : huMarks(a), mathMarks(b),**

**eeMarks(c), csMarks(d),**

**grade(f) {}**

**// Prototyping functions that are used in the display function**

**double computeAverage();**

**void inputData();**

**// Display function called within the forward declared class Instructor**

**void displayData();**

**friend class Instructor;**

**};**

**// Scoped Functions of Student Class**

**// Provides the necessary prompts to input data for records**

**void Student::inputData(){**

**int i = 0;**

**// A straightforward loop that keeps on repeating until unless**

**// the instructor enters marks within the range of 0 - 100**

**do**

**{**

**if (i > 0) {**

**cout << "\n\tPlease input the marks in the range of 0 - 100!\n";**

**}**

**cout << "\nGrade Marking Prompts";**

**cout << "\n\tEnter marks in HU Course: ";**

**cin >> huMarks;**

**cout << "\tEnter marks in MATH Course: ";**

**cin >> mathMarks;**

**cout << "\tEnter marks in CS Course: ";**

**cin >> csMarks;**

**cout << "\tEnter marks in EE Course: ";**

**cin >> eeMarks;**

**i++;**

**} while (!((huMarks > 0 && huMarks < 100) && (csMarks > 0 && csMarks < 100)**

**&& (mathMarks > 0 && mathMarks < 100) && (eeMarks > 0 && eeMarks < 100)));**

**}**

**// Computing average of all the subjects**

**double Student::computeAverage(){**

**average = (huMarks + mathMarks + eeMarks + csMarks) / 4;**

**return average;**

**}**

**// Record Output Interface**

**void Student::displayData(){**

**cout << "\n\t\t Student Record";**

**// Printing Information**

**cout << "\n\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";**

**cout << "\n\t|\t " <<"\t\t|"; // Spacing**

**cout << "\n\t|\tName: " << name <<"\t|";**

**cout << "\n\t|\tRoll Number: " << rollNumber <<"\t|";**

**cout << "\n\t|\tAge: " << age <<"\t\t\t|";**

**cout << "\n\t|\t " <<"\t\t|"; // Spacing**

**// Printing out results**

**cout << "\n\t|\tHU Marks: " << huMarks <<"\t\t|";**

**cout << "\n\t|\tMATH Marks: " << mathMarks <<"\t\t|";**

**cout << "\n\t|\tCS Marks: " << csMarks <<"\t\t|";**

**cout << "\n\t|\tEE Marks: " << eeMarks <<"\t\t|";**

**cout << "\n\t|\t " <<"\t\t|"; // Spacing**

**cout << "\n\t|\tAverage Marks: " << computeAverage() << "\t|";**

**cout << "\n\t|\tGrade: " << grade <<"\t\t|";**

**cout << "\n\t \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_";**

**// End**

**}**

**// Constructing Instructor Class with inherent properties of Person**

**class Instructor : public Person {**

**public:**

**void accessInh(){**

**setInfo();**

**}**

**// Controls and modifies the data members within the Student class**

**void controlRecord(Student s){**

**s.inputData();**

**cout << "\nInput a Grade for the Current Student: ";**

**cin >> s.grade;**

**s.displayData();**

**}**

**friend void digSignature(Instructor);**

**// Friend function to display the information of Instructor**

**// incase a student may have a query about grading**

**// Destructor**

**~Instructor(){**

**cout << "Destructor was called!\n";**

**}**

**};**

**// Body of Friend Function**

**void digSignature(Instructor i){**

**cout << "\n\n\t\tGraded by: " << i.name << ", ID: " << i.rollNumber;**

**cout << "\n\n";**

**}**

**// Main Program, Instantiating Objects and Printing Records**

**int main() {**

**// Welcome message**

**cout << "\n\t\tStudent Grade & Record System";**

**// Initializing Student Object**

**Student s\_A;**

**cin.ignore();**

**cout << "\nSet Information for Student";**

**s\_A.setInfo();**

**// Initializing Instructor Object and Accessing Student Data Members**

**// And modifying them through the friend class**

**Instructor c;**

**cin.ignore();**

**cout << "\nSet Information for Instructor";**

**c.accessInh();**

**c.controlRecord(s\_A);**

**digSignature(c);**

**return 0;**

**}**

***Focused Usage***

**Inheritance**

**class Person {**

**.......**

**};**

**class Instructor;**

**// Constructing Student Class with inherent properties of Person**

**class Student : public Person {**

**.......**

**};**

**// Constructing Instructor Class with inherent properties of Person**

**class Instructor : public Person {**

**.......**

**};**

The Student and Instructor class inherit the common characteristics; ***age, rollNumber, name*** from the parent class named Person through the above block of code.

**Friend Class**

**class Instructor;**

**// Constructing Student Class with inherent properties of Person**

**class Student : public Person {**

**.......**

**friend class Instructor;**

**};**

**// Constructing Instructor Class with inherent properties of Person**

**class Instructor : public Person {**

**void accessInh(Student){**

**.......**

**}**

**void controlData(Student){**

**.......**

**}**

**};**

The above block of code allows the Instructor class to access the all the data members and member functions of the Student class, essentially allowing a faculty member to mark and grade the student.

**Friend Function**

**// Constructing Instructor Class with inherent properties of Person**

**class Instructor : public Person {**

**.......**

**friend void digSignature(Instructor);**

**// Friend function to display the information of Instructor**

**};**

**// Body of Friend Function**

**void digSignature(Instructor i){**

**cout << "\n\n\t\tGraded by: " << i.name << ", ID: " << i.rollNumber;**

**cout << "\n\n";**

**}**

The body of the friend function is declared outside the Class, and can be accessed inside the main function directly, without acting as a member function of the Class. Such a block of code helps in reusability that only requires passing the object of the Class Instructor, in this case, a digital signature.

**Constructors & Destructors**

**class Student : public Person {**

**.......**

**// Default constructor that initializes data to their minimum values**

**Student() {**

**huMarks = 0;**

**mathMarks = 0;**

**eeMarks = 0;**

**csMarks = 0;**

**grade = 'F';**

**}**

**// Parametrized Constructor (for direclty coding the results)**

**Student(int a, int b, int c, int d, char f) : huMarks(a), mathMarks(b),**

**eeMarks(c), csMarks(d),**

**grade(f) {}**

**};**

**Output**

**Student Grade & Record System**

**Set Information for Student**

**Input a Name: Muhammad Umer**

**Input an Age: 19**

**Input an ID: 345834**

**Successfully Assigned Data!**

**Set Information for Instructor**

**Input a Name: Random ABC**

**Input an Age: 45**

**Input an ID: 123563**

**Successfully Assigned Data!**

**Grade Marking Prompts**

**Enter marks in HU Course: 78**

**Enter marks in MATH Course: 89**

**Enter marks in CS Course: 95**

**Enter marks in EE Course: 92**

**Input a Grade for the Current Student: A**

**Student Record**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**| |**

**| Name: Muhammad Umer |**

**| Roll Number: 345834 |**

**| Age: 19 |**

**| |**

**| HU Marks: 78 |**

**| MATH Marks: 89 |**

**| CS Marks: 95 |**

**| EE Marks: 92 |**

**| |**

**| Average Marks: 88 |**

**| Grade: A |**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Graded by: Random ABC, ID: 123563**

**Destructor was called!**

**Destructor was called!**