Object Oriented Programming Lab

SPRING - 2024

LAB 11

FAST National University of Computer and Emerging Sciences

Learning Outcomes

In this lab you are expected to learn the following:

- Inheritance

Inheritance

It allows us to create a new class (derived class) from an existing class (base class). The derived class inherits the features from the base class and can have additional features of its own. Inheritance is an is-a relationship. We use inheritance only if an is-a relationship is present between the two classes.

```
class Animal {
   public:
    void eat() {
        cout << "I can eat!" << endl;
    }
    void sleep() {
        cout << "I can sleep!" << endl;
    }
};
// derived class
class Dog : public Animal {
   public:
    void bark() { cout << "I can bark!" << endl; }
};
int main() {
    // Create object of the Dog class
    Dog dog1;
    // Calling members of the base class
    dog1.eat();
    dog1.sleep();
    // Calling member of the derived class
    dog1.bark();
    return 0;
}
```

Output:

```
I can eat!
I can sleep!
I can bark!
```

Public and private inheritance:

The public keyword in the inheritance means that publicly accessible members inherited from the base class stay publicly accessible in the derived class. But for private keyword in the inheritance syntax means that accessible members inherited from the base become private members of derived clas

Lab Task

Submission Instructions:

1. Now create a new folder with name *ROLLNO_SEC_LAB11* **e.g. i23XXXX_SEC_LAB11**
2. Move all of your **.cpp and .h files** to this newly created directory and compress itinto a **.zip file**.
3. Now you have to submit this zipped file on Google Classroom.
4. If you don't follow the above-mentioned submission instruction, you will be marked **zero.**
5. Plagiarism in the Lab Task will result in **zero** marks in the whole category.

Q1.

Design a class named **Employee** derived from class Person.Person class has following data members

- Name(string)
- Address(string)

The Employee class should keep the following information in member variables:

- Employee number(int)
- Hours_worked(int)

Write one or more constructors and the appropriate setters and getter functions for the class. Next, write a class named ProductionWorker that is derived from the Employee class. The ProductionWorker class should have member variables to hold the following information:

- Shift (an integer)
- Hourly pay rate (a double)
- Salary (double)

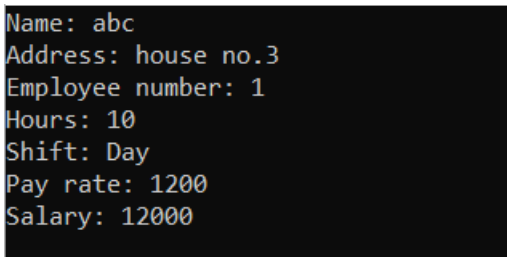
The workday is divided into two shifts: day and night. The shift variable will hold an integer value representing the shift that the employee works. The day shift is shift 1 and the night shift is shift 2. If an employee works more than 5 night shift hours will get a bonus of 1000. Write parameterized constructors and the appropriate setters and getters functions

Add the following member function in production worker

- double calculateSalary()

- void print_details()

Calculate the salary of production worker in by creating objects in main. Print details of the employee



Q2.

In this task, we would implement and learn the usage of protected and public inheritance using one parent class '*Staff*' and two child classes '*Faculty*' and '*NonFaculty*'. **'*Faculty*' would use protected inheritance with '*Staff*' and '*NonFaculty*' would use public inheritance with '*Staff*'**.

Create the staff class and include the following protected data members in it – *name*, *cnic*, *dob*(used for year of birth only) and *contact*. Define the required constructor(s), getters and setter(s) – define a separate getter for each of the data member.

- **string** getName();
- **string** getCnic();
- **int** getDob();
- **int** getAge();
- **unsigned int** getContact();

Create the child class '*Faculty*' first and include the following data members as its protected data members – *payScale*, *roomNumber*, *subject* and *rank*. Define the necessary constructor(s), getters and setter(s) – define a separate getter for each of the data member.

- **int** getPayScale();
- **int** getRoomNumber();
- **string** getSubject();
- **string** getRank():

Create the second child class '*NonFaculty*' and include the following data members as its protected members – *duty*, *startTiming* and *endTiming*. Do the same, define constructors, getters and setters – define a separate getter for each of the data member.

- **string** getDuty();
- **string** getStartTiming();
- **string** getEndTiming();

You need to create two simple classes '*HOD*' and '*Teacher*' inheriting '*Faculty*'. Considering including the following attributes as private and public respectively – HOD(*DeptID*, *Department*), Teacher(*TeachProgram*). Define the required constructor(s), getters and setter(s).

class HOD

- **int** getDeptID();
- **string** getDepartment();

class Teacher

- **string** getProgram();

Now, create '*Security'* and '*GateKeeper'* on '*NonFaculty*'. Consider including the following attributes as private and public respectively – Security(*GunMade*, *LisenceNumber*), GateKeeper(*GateNumber*). Define the necessary constructor(s), getters and setter(s).

class Security

- **string** getGunMade();
- **string** getLisenceNumber();

class GateKeeper

- **int** getGateNumber();

Your main task is to implement print_detail() function in all the classes and, each class will print only the attributes defined in it

```
Printing HOD details:
Name: John Doe
CNIC: 12345-67890
DOB: 1980
Age: 42
Contact: 1234567
Pay Scale: 3
Room Number: 1
Subject: Computer Science
Rank: Professor
Department ID: 1
Department: Department of Computer Science

Printing Teacher details:
Name: Jane Smith
CNIC: 98765-43210
DOB: 1990
Age: 32
Contact: 2345678
Pay Scale: 4
Room Number: 2
Subject: Physics
Rank: lecturer
Teaching Program: Bachelor of Science in Physics

Printing Security details:
Name: Bob Jones
CNIC: 11111-1111
DOB: 1975
Age: 47
Contact: 3456789
Duty: Security Guard
Start Timing: 8:00 AM
End Timing: 5:00 PM
Gun Made: abc
License Number: 123

Printing GateKeeper details:
Name: Tom Smith
CNIC: 22222-2222
DOB: 1985
Age: 37
Contact: 4567890
Duty: guard
Start Timing: 9:00:00
End Timing: 17:00:00
Gate Number: 2
```