



# **OBJECT ORIENTED PROGRAMMING**

## **LAB # 12**

## **ASSIGNMENT**

**NAME: S.M.IRTIZA**

**ROLL NO. : 22K-4638**

**CLASS: 2-F**

## QUESTION#01:

The program defines three classes: Person, Student, and Faculty. The Person class is an abstract base class that has two data members: a string name\_ and an int age\_. The class also declares a pure virtual function display(), which has no implementation. The Student class is derived from Person and adds a data member id\_ of type int. The class overrides the display() function and prints out the name\_, age\_, and id\_ of the student. The Faculty class is also derived from Person and adds a data member department\_ of type string. The class overrides the display() function and prints out the name\_, age\_, and department\_ of the faculty member. The University class manages a collection of Person objects. It has an array people\_ of size MAX\_PEOPLE (a constant integer defined in the program) that holds pointers to Person objects. The class also has an integer num\_people\_ that keeps track of the number of Person objects currently in the array. The University class defines two member functions: addPerson() and displayPeople(). The addPerson() function takes a pointer to a Person object as its parameter and adds it to the next available slot in the people\_ array. If the array is already full, the function prints an error message. The displayPeople() function iterates through the people\_ array and calls the display() function on each Person object to print out their information. In the main() function, the program creates three objects: s1 and s2 of class Student, and f1 of class Faculty. The program then adds these objects to an instance of the University class using the addPerson() function, and displays all the people using the displayPeople() function. Finally, the program deallocates the memory for the objects using delete.

## CODE:

```
//NAME: S.M.IRTIZA | NU ID: 22k-4638
```

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
const int MAX_PEOPLE=10;
```

```
class Person{
```

```
    string name_;
```

```
    int age_;
```

```
public:
```

```
    Person(){}
```

```
    Person(string a, int b){
```

```
        name_=a;
```

```
        age_=b;
```

```
    }
```

```
    string getName_() {
```

```
        return this->name_;
```

```
    }
```

```
    void setName_(string name_) {
```

```
        this->name_ = name_;
```

```
    }
```

```
    int getAge_() {
```

```
        return this->age_;
```

```
    }
```

```
    void setAge_(int age_) {
```

```
        this->age_ = age_;
    }

    virtual void display()=0;
};

class Student:public Person{
    int id_;

public:
    Student(){}
    Student(string a,int b, int c):Person(a,b){
        id_=c;
    }

    int getId_()
    {
        return this->id_;
    }
    void setId_(int id_)
    {
        this->id_ = id_;
    }

    void display(){
        cout<<"name: "<<getName_()<<endl;
        cout<<"age: "<<getAge_()<<endl;
        cout<<"id: "<<getId_()<<endl;
    }
};

class Faculty: public Person{
    string departement_;

public:
    Faculty(){}
    Faculty(string a,int b,string c):Person(a,b){
        departement_=c;
    }
    string getDepartement_()
    {
        return this->departement_;
    }
    void setDepartement_(string departement_)
    {
        this->departement_ = departement_;
    }

    void display(){
```

```

        cout<<"name: "<<getName_()<<endl;
        cout<<"age: "<<getAge_()<<endl;
        cout<<"Departement: "<<getDepartement_()<<endl;
    }
};

class University{
    int num_people_;

    Person* people_[MAX_PEOPLE];
public:
    University(){
        num_people_=0;
    }

    void addPerson(Person* p){
        if( num_people_<MAX_PEOPLE){
            people_[num_people_]=p;
            num_people_++;
        }
        else{
            cout<<"No vacant space left"<<endl;
        }
    }

    void displayPeople(){
        for(int i=0;i<num_people_;i++){
            people_[i]->display();
        }
    }

    int getNum_people_()
    {
        return this->num_people_;
    }
    void setNum_people_(int num_people_)
    {
        this->num_people_ = num_people_;
    }
};

int main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22k-4638"<<endl;
    Student *s1= new Student("irtiza",18,1);
    Student *s2= new Student("sadiq",20,2);
    Faculty *f1= new Faculty("Shahzad",45,"CS");
    University U;
    U.addPerson(s1);
    U.addPerson(s2);
    U.addPerson(f1);
}

```

```
U.displayPeople();
```

```
    delete s1;  
    delete s2;  
    delete f1;  
    return 0;  
}
```

## OUTPUT :

```
NAME: S.M.IRTIZA | NU ID: 22k-4638  
name: irtiza  
age: 18  
id: 1  
name: sadiq  
age: 20  
id: 2  
name: Shahzad  
age: 45  
Departement: CS
```

**QUESTION # 02:**

There are four classes: Order, Starter, MainDish, and Meal. The Order class is the base class, and the Starter and MainDish classes both inherit from it using virtual inheritance. The Order class has a function serve(), which is overridden in the Starter and MainDish classes with their specific implementations of serving the food. The Meal class overrides the serve() function as well, but it calls the serve() functions of both Starter and MainDish classes, and adds a message to let the customer know that they can enjoy their meal. In the main() function, an object of the Meal class is created, and its serve() function is called, which triggers the overridden function in the Meal class to print out the messages of serving the starter, main dish, and enjoying the meal.

**CODE:**

```
//NAME: S.M.IRTIZA | NU ID: 22K-4638
```

```
#include<iostream>
```

```
#include<string>
```

```
using namespace std;
```

```
class Order{
```

```
public:
```

```
Order(){}
```

```
virtual void serve(){
```

```
    cout<<"its order time!"<<endl;
```

```
}
```

```
};
```

```
class Starter:virtual public Order{
```

```
public:
```

```
Starter(){}
```

```
void serve(){
```

```
    cout<<"it's starter time!"<<endl;
```

```
}
```

```
};
```

```
class MainDish:virtual public Order{
```

```
public:
```

```
MainDish(){}
```

```
void serve(){
```

```
    cout<<"it's main dish time!"<<endl;
```

```
}
```

```
};
```

```
class Meal:public Starter,public MainDish{
```

```
public:
```

```
Meal(){}
```

```
void serve(){
```

```
    Starter::serve();
```

```
    MainDish::serve();
```

```
    cout<<"it's meal time!"<<endl;
```

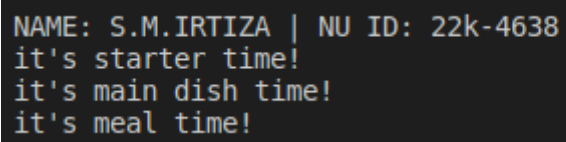
```
}
```

```
};
```

```
int main(){
```

```
cout<<"NAME: S.M.IRTIZA | NU ID: 22k-4638"<<endl;  
Meal m;  
m.serve();  
return 0;  
}
```

## OUTPUT :

A screenshot of a terminal window with a black background and light blue/grey text. It shows the output of a C++ program: "NAME: S.M.IRTIZA | NU ID: 22k-4638", "it's starter time!", "it's main dish time!", and "it's meal time!".

```
NAME: S.M.IRTIZA | NU ID: 22k-4638  
it's starter time!  
it's main dish time!  
it's meal time!
```

**QUESTION # 03:**

There are two classes Car and Bike that both inherit from the abstract base class Vehicle. The Vehicle class has a pure virtual function display(), which is overridden in the Car and Bike classes with their specific implementation to display the type of the vehicle. There are also two classes CarFactory and BikeFactory, both of which inherit from the abstract base class VehicleFactory. The VehicleFactory class has a pure virtual function createVehicle(), which returns a pointer to a Vehicle object. The CarFactory and BikeFactory classes both implement the createVehicle() function and return a pointer to a Car or Bike object respectively. In the main() function, the user is prompted to enter the type of vehicle they want to create, and based on their choice, either a CarFactory or a BikeFactory object is created. Then, the createVehicle() function of the factory object is called to create a new Vehicle object, and its display() function is called to print out the type of the vehicle.

**CODE:**

```
//NAME: S.M.IRTIZA | NU ID: 22k-4638
#include<iostream>
#include<string>
using namespace std;
class Vehicle{
    public:
        virtual void display()=0;
};
class Car:public Vehicle{
    public:
    Car(){}
    void display(){
        cout<<"CIVIC 23 "<<endl;
    }
};
class Bike:public Vehicle{
    public:
    Bike(){}
    void display(){
        cout<<"HONDA 125"<<endl;
    }
};

class VehicleFactory{
    public:
    virtual Vehicle* createVehicle()=0;
};

class CarFactory:public VehicleFactory{
    public:
    Vehicle* createVehicle(){
        cout<<"Car is created!"<<endl;
        return new Car();
    }
};

class BikeFactory:public VehicleFactory{
    public:
    Vehicle* createVehicle(){
        cout<<"Bike is created"<<endl;
        return new Bike();
    }
};

int main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22k-4638"<<endl;
    string choice;
```



```
cout<<"enter what you want to create(BIKE/CAR): "<<endl;
cin>>choice;
VehicleFactory* factory;
if(choice=="car"){
    factory=new CarFactory();
}
else if(choice=="bike"){
    factory=new BikeFactory();
}
else{
    cout<<"invalid choice"<<endl;
}

Vehicle *vehicle= factory->createVehicle();
vehicle->display();
delete factory;
delete vehicle;
return 0;
}
```

## OUTPUT :

```
NAME: S.M.IRTIZA | NU ID: 22k-4638
enter what you want to create(BIKE/CAR):
bike
Bike is created
HONDA 125
```

## QUESTION # 04

There are two classes VeggieBurger and ChickenBurger that both inherit from the base class Burger. The Burger class has a virtual function makeBurger(), which is overridden in the VeggieBurger and ChickenBurger classes with their specific implementation to make a veggie or chicken burger. In the main() function, the user is prompted to enter the type of burger they want to make, and based on their choice, either a VeggieBurger or a ChickenBurger object is created. Then, the makeBurger() function of the burger object is called to make the specific type of burger.

### CODE:

```
//NAME: S.M.IRTIZA | NU ID: 22k-4638
#include<iostream>
#include<string>
using namespace std;

class Burger{
public:
    virtual void makeBurger()=0;
};

class VeggieBurger:public Burger{
public:
    void makeBurger(){
        cout<<"Veggie Burger"<<endl;
    }
};

class ChickenBurger:public Burger{
public:
    void makeBurger(){
        cout<<"Chicken Burger"<<endl;
    }
};

int main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22k-4638"<<endl;
    string choice;
    cout<<"enter the choice of burger(veggie/chicken): "<<endl;
    cin>>choice;

    if(choice=="veggie"){
        VeggieBurger burger;
        burger.makeBurger();
    }
    else if(choice=="chicken"){
        ChickenBurger burger;
        burger.makeBurger();
    }
    else{
        cout<<"invalid choice"<<endl;
    }

    // burger->makeBurger();
    return 0;
}
```

**OUTPUT :**

```
NAME: S.M.IRTIZA | NU ID: 22k-4638  
enter the choice of burger(veggie/chicken):  
veggie  
Veggie Burger
```

**QUESTION# 05:**

There are three classes CityTour, BeachTour, and AdventureTour, that all inherit from the Tour base class. The Tour class has a pure virtual function tourDetails() which is overridden in each of the derived classes with their specific implementation to provide details of the tour package. CityTour::tourDetails() function prompt: This tour package includes a visit to famous landmarks, museums, and local markets. BeachTour::tourDetails() function prompt: This tour package includes beach activities, water sports, and a relaxing stay at the resort. AdventureTour::tourDetails() function prompt: This tour package includes trekking, camping, and other outdoor activities. In the main() function, the user is prompted to enter the type of tour package they want to select, and based on their choice, either a CityTour, BeachTour, or AdventureTour object is created. Then, the tourDetails() function of the tour object is called to display the specific details of the selected tour package.

**CODE:**

```
//NAME: S.M.IRTIZA | NU ID: 22k-4638
#include<iostream>
#include<string>
using namespace std;
class Tour{
    public:
        virtual void tourDetails()=0;
};

class CityTour:public Tour{
    public:
        void tourDetails(){
            cout<<"This tour package includes a visit to famous landmarks, museums, and local
markets."<<endl;
        }
};

class BeachTour: public Tour{
    public:
        void tourDetails(){
            cout<<"This tour package includes beach activities, water sports, and a relaxing stay at
the resort."<<endl;
        }
};

class AdventureTour: public Tour{
    public:
        void tourDetails(){
            cout<<"This tour package includes trekking, camping, and other outdoor
activities."<<endl;
        }
};

int main(){
    cout<<"NAME: S.M.IRTIZA | NU ID: 22k-4638"<<endl;
    string choice;
    cout<<"select the tour package(City/Beach/Adventure): "<<endl;
    cin>>choice;

    Tour* tour;
```

```
if(choice=="city"){
    tour=new CityTour();
}
else if(choice=="beach"){
    tour= new BeachTour();
}
else if(choice=="adventure"){
    tour=new AdventureTour();
}
else{
    cout<<"invalid choice"<<endl;
}

tour->tourDetails();
return 0;
}
```

### **OUTPUT :**

```
NAME: S.M.IRTIZA | NU ID: 22k-4638
select the tour package(City/Beach/Adventure):
city
This tour package includes a visit to famous landmarks, museums, and local markets.
```