Object-Oriented Programming (Lab Exercise)
Spring 2024
Software Engineering Department, NED University of Engineering and Technology

Week #7

Inheritance in C++

Objective:
Understand and implement the concept of inheritance in object-oriented programming (OOCP).

Introduction:
Inheritance is a fundamental principle in OOP that allows a class to inherit properties and methods from another class. The class that is inherited from is called the base class (or parent class), and the class that inherits is called the derived class (or child class).This promotes code reusability, organization, and the creation of a hierarchical relationship between classes.
In this lab, you will:

- Learn the basic principles of inheritance.
- Create a base class with common attributes and methods.
- Create derived classes that inherit from the base class and extend or override its functionality.
- Understand how to use inheritance to write more efficient and maintainable code.

Key Concepts:

Base Class (Parent Class): The class whose properties and methods are inherited.
Derived Class (Child Class): The class that inherits properties and methods from the base class.
Constructor (__init__ method): Initializes the object's state.
Method Overriding: A derived class can provide a specific implementation of a method that is already defined in its base class.

Example Scenario:
We will create a base class called Animal with attributes and methods common to all animals. Then, we will create derived classes Dog and Cat that inherit from the Animal class and add specific behaviors.

Step 1: Define the Base Class (Animal):

```
#include <iostream>
#include <string>
using namespace std;

class Animal {
```

```
protected:
    string name;
    string species;

public:

    // Constructor
    Animal(string n, string s) : name(n), species(s) {}

    // Method to make sound (to be overridden by derived classes)
    void makeSound() {
        cout << "Some generic sound" << endl;
    }

    // Method to display information
    void displayInfo() {
        cout << "Name: " << name << ", Species: " << species << endl;
    }
};
```

Step 2: Define the Derived Classes (Dog and Cat)

```
// Derived class Dog
class Dog : public Animal {
private:
    string breed;

public:
    // Constructor
    Dog(string n, string s, string b) : Animal(n, s), breed(b) {}

    // Overriding the makeSound method
    void makeSound() {
        cout << "Woof!" << endl;
    }

    // Method to display Dog-specific information
    void displayInfo() {
        Animal::displayInfo(); // Call base class method
        cout << "Breed: " << breed << endl;
    }
};

// Derived class Cat
class Cat : public Animal {
```

```
private:
    string color;

public:
    // Constructor
    Cat(string n, string s, string c) : Animal(n, s), color(c) {}

    // Overriding the makeSound method
    void makeSound() {
        cout << "Meow!" << endl;
    }

    // Method to display Cat-specific information
    void displayInfo() {
        Animal::displayInfo(); // Call base class method
        cout << "Color: " << color << endl;
    }
};
```

Step 3: Create Objects and Demonstrate Inheritance

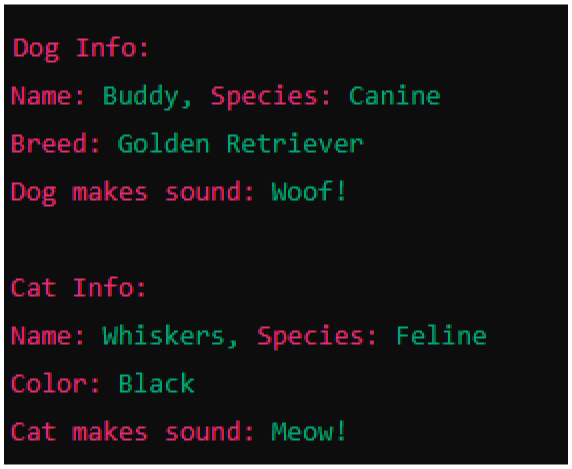
```
int main() {
    // Creating objects of Dog and Cat
    Dog dog("Buddy", "Canine", "Golden Retriever");
    Cat cat("Whiskers", "Feline", "Black");

    // Displaying information and sounds of Dog
    cout << "Dog Info:" << endl;
    dog.displayInfo();
    cout << "Dog makes sound: ";
    dog.makeSound();

    cout << endl;

    // Displaying information and sounds of Cat
    cout << "Cat Info:" << endl;
    cat.displayInfo();
    cout << "Cat makes sound: ";
    cat.makeSound();

    return 0;
}
```



```
Dog Info:
Name: Buddy, Species: Canine
Breed: Golden Retriever
Dog makes sound: Woof!

Cat Info:
Name: Whiskers, Species: Feline
Color: Black
Cat makes sound: Meow!
```

Explanation:
Base Class (Animal):

The Animal class has protected attributes name and species which can be accessed by derived classes It includes a constructor to initialize these attributes and a method makeSound that outputs a generic sound. The displayInfo method shows the animal's name and species.

Derived Classes (Dog and Cat):

Both Dog and Cat classes inherit from the Animal class using public inheritance (: public Animal).Each derived class has its own specific attribute: breed for Dog and color for Cat. The makeSound method is overridden to provide specific sounds for each class.
The displayInfo method is extended to include the specific attributes (breed and color).

Creating Objects and Demonstrating Inheritance:

Objects dog and cat are created with their specific attributes. The displayInfo method from the base class is called to show inherited attributes, and the overridden makeSound method.This example illustrates the basics of inheritance in C++, showcasing how a derived class can extend and customize the behavior of a base class.

Exercise:

Question 1: Basic Inheritance

Create a base class Vehicle with a method move() that prints "Vehicle is moving". Derive a class Car from Vehicle and override the move() method to print "Car is moving".

Question 2: Constructor in Derived Class

Create a base class Person with attributes name and age. Create a derived class Student with an additional attribute studentID. Initialize these attributes using constructors.

Question 3: Method Overriding

Create a base class Shape with a method draw() that prints "Drawing Shape". Create derived classes Circle and Square that override the draw() method to print "Drawing Circle" and "Drawing Square", respectively.

Question 4: Access Specifiers

Create a base class Base with a private attribute privateVar, a protected attribute protectedVar, and a public attribute publicVar. Create a derived class Derived and demonstrate access to these attributes. Write your observations.

Question 5
You are required to create a C++ program that demonstrates the concept of inheritance.Consider the following requirements:

Base Class - Shape:
Attributes:
- color (string)
- Constructor that initializes color.
- A pure virtual function area() which will return the area of the shape.
- A virtual function display() that prints the color of the shape.

Derived Classes:
Rectangle:
- Attributes:width (double),height (double)
- Constructor that initializes color, width, and height.
- Override area() to calculate and return the area of the rectangle.
- Override display() to print the color, width, height, and area of the rectangle.

Circle:

Attributes:

- radius (double)
- Constructor that initializes color and radius.
- Override area() to calculate and return the area of the circle.
- Override display() to print the color, radius, and area of the circle.

Main Function:

- Create instances of Rectangle and Circle with different attributes.
- Use a pointer to Shape to store the addresses of these instances and call their display() method**.**