Object-oriented programming CS10003

Lecturer: Do Nguyen Kha

Contents

Object-Oriented Programming (OOP)

“Object-oriented programming (OOP) is a programming paradigm based on the concept of ***objects****, which can contain* *data* *and* *code**: data in the form of fields (often known as attributes or properties), and code in the form of procedures (often known as methods).”* - Wikipedia

Class and Object

A class is a definition of objects of the same kind. In other words, a class is a blueprint, template, or prototype that defines and describes the attributes and behaviors common to all objects of the same kind.

- Fruit: Orange, Apple, Pineapple, Mango, Kiwi…
- Car: Volvo, Toyota, Tesla, Audi, BMW…
- Animal: Elephant, Cat, Dog, Dolphin…
- Shape: Circle, Rectangle, Square…

Define a class

```
class Car {
public:
  int serial;
  string manufacturer;
  string color;
  bool isEV;
};
```

Create an object

```
Car car1;

Car car2, aSpecialCar;

car1.serial = 123456;

car1.manufacturer = "Tesla";

car1.isEV = true;

car1.color = "black";

cout << car1.color;

cin >> car1.color;
```

Access Modifiers

In C++, there are three access specifiers:

- `public` - members are accessible from outside the class
- `private` - members cannot be accessed (or viewed) from outside the class
- `protected` - members cannot be accessed from outside the class, however, they can be accessed in inherited classes.

Method

Methods are functions that belongs to the class

```
class MyClass {
  public:
    void myMethod() {
      cout << "Hello World!";
    }
};
int main() {
  MyClass myObj;
  myObj.myMethod();
  return 0;
}
```

Encapsulation

Encapsulation is a way to restrict the direct access to some components of an object, so users cannot access state values for all of the variables of a particular object. Encapsulation can be used to hide both data members and data functions or methods associated with an instantiated class or object.

- Data Hiding
- Access control
- Abstraction
- Maintainability and Flexibility

Encapsulation

```
class Employee {
  private:
    int salary;

  public:
    // Setter
    void setSalary(int s) {
      this.salary = s;
    }
    // Getter
    int getSalary() {
      return this.salary;
    }
};
```

The `this` pointer

In C++, `this` is a keyword that refers to the current instance of the class.

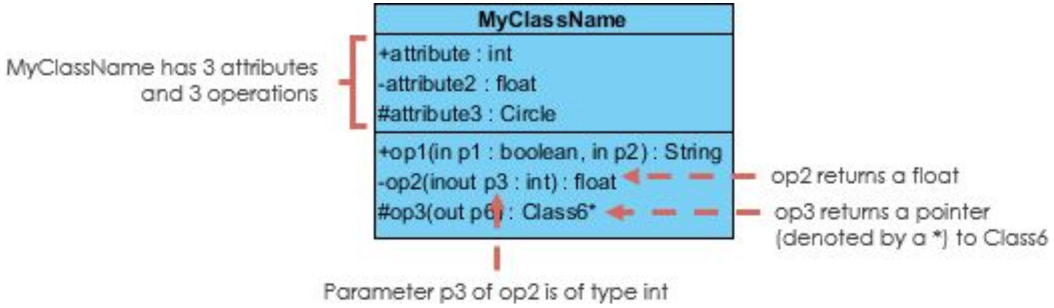
Dynamic Memory Allocation

```
Employee* e1 = new Employee();

e1->setSalary(10000);

cout << e1->getSalary();

delete e1;
```

UML

The UML (**U**nified **M**odeling **L**anguage) Class diagram is a graphical notation used to construct and visualize object oriented systems:

- Classes
- Attributes
- Methods
- and the relationships among objects


MyClassName
+attribute : int
-attribute2 : float
#attribute3 : Circle
+op1(in p1 : boolean, in p2) : String
-op2(inout p3 : int) : float
#op3(out p6) : Class6*
MyClassName has 3 attributes and 3 operations
op2 returns a float
op3 returns a pointer (denoted by a *) to Class6
Parameter p3 of op2 is of type int

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/

Coding Convention

- Each class must be defined in a header file (.h): Employee.h
  - Using #define guard to prevent multiple inclusions

    ```
    #ifndef EMPLOYEE_H_
    #define EMPLOYEE_H_
    …
    #endif  // EMPLOYEE_H_
    ```
- Class method is implemented in a C++ source code file (.cpp): Employee.cpp