Computer Programming (b) - E1124

(Spring 2021-2022)

Lecture 9

C++ Object-Oriented Programming

INSTRUCTOR

Dr / Ayman Soliman

➢ Contents

- What is OOP?
- Object-oriented programming advantages
- What are Classes and Objects?
- Create a Class
- Create an Object
- Class Methods

What is OOP?

- OOP stands for Object-Oriented Programming.

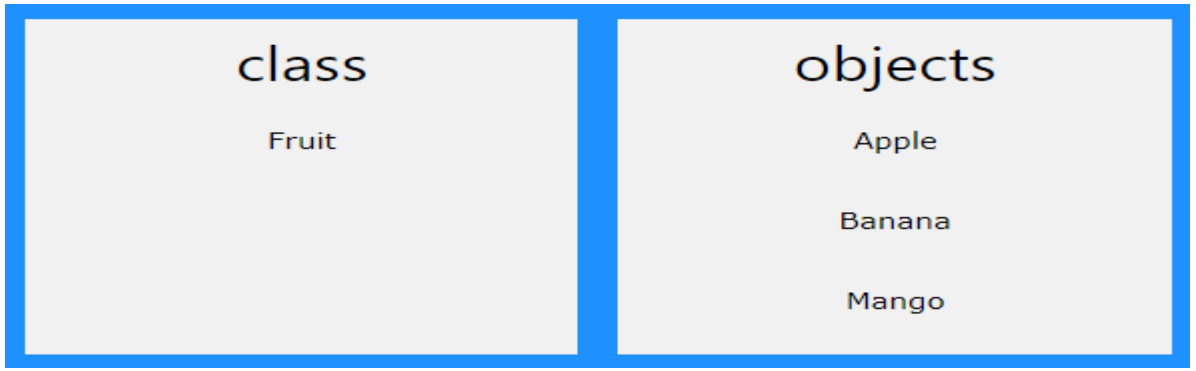
- Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming advantages

- OOP is faster and easier to execute
- OOP provides a clear structure for the programs
- OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
- OOP makes it possible to create full reusable applications with less code and shorter development time

➢ What are Classes and Objects?

- Classes and objects are the two main aspects of object-oriented programming.
- Look at the following illustration to see the difference between class and objects:



So, a class is a template for objects,

and an object is an instance of a class.

C++ Classes/Objects

- C++ is an object-oriented programming language.
- Everything in C++ is associated with classes and objects, along with its attributes and methods.
- For example: in real life, a car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.
- Attributes and methods are basically variables and functions that belongs to the class. These are often referred to as "class members".
- A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.

➢ Create a Class

➢ To create a class, use the class keyword:

Example

➢ Create a class called "MyClass":

```
class MyClass {          // The class
  public:                // Access specifier
    int myNum;           // Attribute (int variable)
    string myString;     // Attribute (string variable)
};
```

➢ Example explained

- The class keyword is used to create a class called MyClass.
- The public keyword is an access specifier, which specifies that members (attributes and methods) of the class are accessible from outside the class. You will learn more about access specifiers later.
- Inside the class, there is an integer variable myNum and a string variable myString. When variables are declared within a class, they are called attributes.
- At last, end the class definition with a semicolon ;

Create an Object

- In C++, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects.

- To create an object of MyClass, specify the class name, followed by the object name.

- To access the class attributes (myNum and myString), use the dot syntax (.) on the object:

➢ Example

➢ Create an object called "myObj" and access the attributes:

```
class MyClass {                 // The class
  public:                       // Access specifier
    int myNum;                  // Attribute (int variable)
    string myString;            // Attribute (string variable)
};

int main() {
  MyClass myObj;                // Create an object of MyClass

  myObj.myNum = 15;             // Access attributes and set values
  myObj.myString = "Some text";

cout << myObj.myNum << "\n";    // Print attribute values
cout << myObj.myString;
  return 0;
}
```

➢ Multiple Objects

```
// Create a Car class with some attributes
class Car {
  public:
    string brand;
    string model;
    int year;
};

int main() {
  // Create an object of Car
  Car carObj1;
  carObj1.brand = "BMW";
  carObj1.model = "X6";
  carObj1.year = 2020;
  // Create another object of Car
  Car carObj2;
  carObj2.brand = “Chevrolet";
  carObj2.model = “Aveo";
  carObj2.year = 2015;

  // Print attribute values
  cout << carObj1.brand << " " <<
carObj1.model << " " << carObj1.year << "\n";
  cout << carObj2.brand << " " <<
carObj2.model << " " << carObj2.year << "\n";
  return 0;
}
```

➢ Class Methods

- Methods are functions that belongs to the class.

- There are two ways to define functions that belongs to a class:
    1. Inside class definition
    2. Outside class definition
- In the following example, we define a function inside the class, and we name it "myMethod".
- Note: You access methods just like you access attributes; by creating an object of the class and using the dot syntax (.):

➢ Inside Example

```
class MyClass {                 // The class
  public:                       // Access specifier
    void myMethod() {           // Method/function defined inside the class
      cout << "Hello World!";
    }
};

int main() {
  MyClass myObj;                // Create an object of MyClass
  myObj.myMethod();             // Call the method
  return 0;
}
```

➢ Outside Example

➢ To define a function outside the class definition, you have to declare it inside the class and then define it outside of the class. This is done by specifying the name of the class, followed the scope resolution :: operator, followed by the name of the function:

```
class MyClass {          // The class
  public:                // Access specifier
    void myMethod();  // Method/function declaration
};
// Method/function definition outside the class
void MyClass::myMethod() {
  cout << "Hello World!";
}
```

```
int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```

Parameters

- You can also add parameters:

```
#include <iostream>
using namespace std;
class Car {
  public:
    int speed(int maxSpeed);
};
int Car::speed(int maxSpeed) {
  return maxSpeed;
}
int main() {
  Car myObj;                    // Create an object of Car
  cout << myObj.speed(200);     // Call the method with an argument
  return 0;
}
```

Thank you