Computer Programming (b) - E1124

(Spring 2021-2022)

Lecture 12

Characteristics of an OOP language

INSTRUCTOR

Dr / Ayman Soliman

Contents

- Introduction
- Characteristics of an Object-Oriented Programming language
- Class
- Object
- Encapsulation
- Abstraction
- Polymorphism
- Why and When to Use "Inheritance" and "Polymorphism"

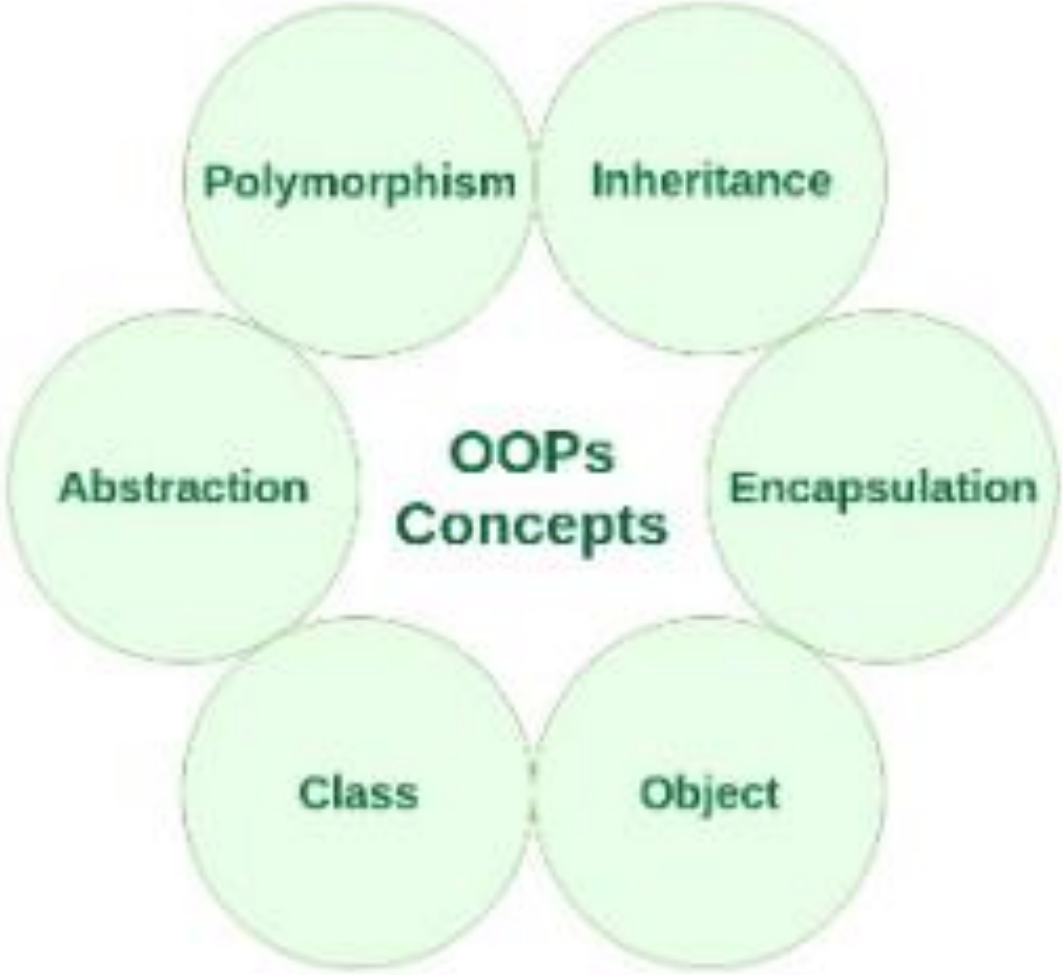
Introduction

- Object-oriented programming – As the name suggests uses objects in programming.

- Object-oriented programming aims to implement real-world entities like inheritance, hiding, polymorphism, etc. in programming.

- The main aim of OOP is to bind together the data and the functions that operate on them so that no other part of the code can access this data except that function.

➢ Characteristics of an Object-Oriented Programming language



➢ Class

- The building block of C++ that leads to Object-Oriented programming is a Class.

- It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class. A class is like a blueprint for an object.

- For Example: Consider the Class of Cars. There may be many cars with different names and brand but all of them will share some common properties like all of them will have 4 wheels, Speed Limit, Mileage range etc. So here, Car is the class and wheels, speed limits, mileage are their properties.

➢ Class (cont.)

- A Class is a user-defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables and together these data members and member functions define the properties and behavior of the objects in a Class.
- In the above example of class Car, the data member will be speed limit, mileage etc. and member functions can apply brakes, increase speed etc.
- We can say that a Class in C++ is a blue-print representing a group of objects which shares some common properties and behaviors.

Object

- An Object is an identifiable entity with some characteristics and behavior.
- An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

```
class person
{
   char name[20];
   int id;
public:
   void getdetails(){}
};
```

```
int main()
{
   person p1;    // p1 is an object
}
```

Object

- Object take up space in memory and have an associated address like a record in pascal or structure in C.

- When a program is executed, the objects interact by sending messages to one another.

- Each object contains data and code to manipulate the data. Objects can interact without having to know details of each other's data or code, it is sufficient to know the type of message accepted and type of response returned by the objects.

➢ Encapsulation

- In normal terms, Encapsulation is defined as wrapping up of data and information under a single unit.
- In Object-Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them.

Abstraction

- Data abstraction is one of the most essential and important features of object-oriented programming in C++.
- Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation.

- Consider a real-life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of the car or applying brakes will stop the car, but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc. in the car.

➢ Abstraction using Classes

- We can implement Abstraction in C++ using classes.
- The class helps us to group data members and member functions using available access specifiers.
- A Class can decide which data member will be visible to the outside world and which is not.

➢ Abstraction in Header files

- One more type of abstraction in C++ can be header files.

- For example, consider the pow() method present in math.h header file.

- Whenever we need to calculate the power of a number, we simply call the function pow() present in the math.h header file and pass the numbers as arguments without knowing the underlying algorithm according to which the function is actually calculating the power of numbers.

Polymorphism

- The word polymorphism means having many forms. In simple words, we can define polymorphism as the ability of a message to be displayed in more than one form.

- A person at the same time can have different characteristic. Like a man at the same time is a father, a husband, an employee. So, the same person posse's different behavior in different situations. This is called polymorphism.

➢ Polymorphism (cont.)

- An operation may exhibit different behaviors in different instances. The behavior depends upon the types of data used in the operation.

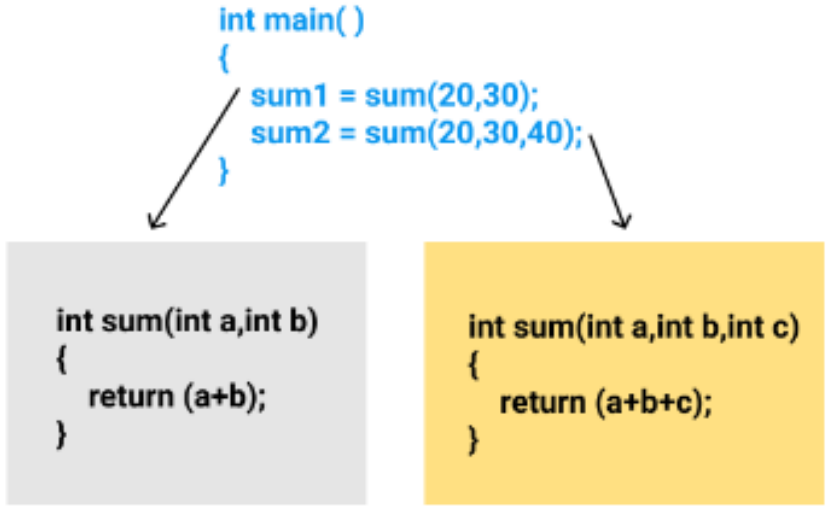
- C++ supports operator overloading and function overloading.

  - Operator Overloading: The process of making an operator to exhibit different behaviors in different instances is known as operator overloading.
  - Function Overloading: Function overloading is using a single function name to perform different types of tasks.

- Polymorphism is extensively used in implementing inheritance.

➢ Example

➢ Suppose we have to write a function to add some integers, sometimes there are 2 integers, sometimes there are 3 integers. We can write the Addition Method with the same name having different parameters, the concerned method will be called according to parameters.



➢ Polymorphism

➢ Like we specified in the previous lecture; Inheritance lets us inherit attributes and methods from another class. Polymorphism uses those methods to perform different tasks. This allows us to perform a single action in different ways.

➢ For example, think of a base class called Animal that has a method called animalSound(). Derived classes of Animals could be Pigs, Cats, Dogs, Birds - and they also have their own implementation of an animal sound (the pig oinks, and the cat meows, etc.):

➢ Example

```
// Base class
class Animal {
  public:
    void animalSound() {
    cout << "The animal makes a sound \n" ;
  }
};
// Derived class
class Pig : public Animal {
  public:
    void animalSound() {
    cout << "The pig says: wee wee \n" ;
  }
};
// Derived class
class Dog : public Animal {
  public:
    void animalSound() {
    cout << "The dog says: bow wow \n" ;
  }
};
int main() {
  Animal myAnimal;
  Pig myPig;
  Dog myDog;

  myAnimal.animalSound();
  myPig.animalSound();
  myDog.animalSound();
  return 0;
}
```

```
The animal makes a sound
The pig says: wee wee
The dog says: bow wow
```

➢ Why and When to Use "Inheritance" and "Polymorphism"

- It is useful for code reusability: reuse attributes and methods of an existing class when you create a new class.

Thank you