Object-Oriented Programming (oop)

By

Dr. Rabih Ali Sbera

Department of Computer Engineering Techniques -2nd

2025-2026

Let's learn programming



lets learn online

Introduction

Classes

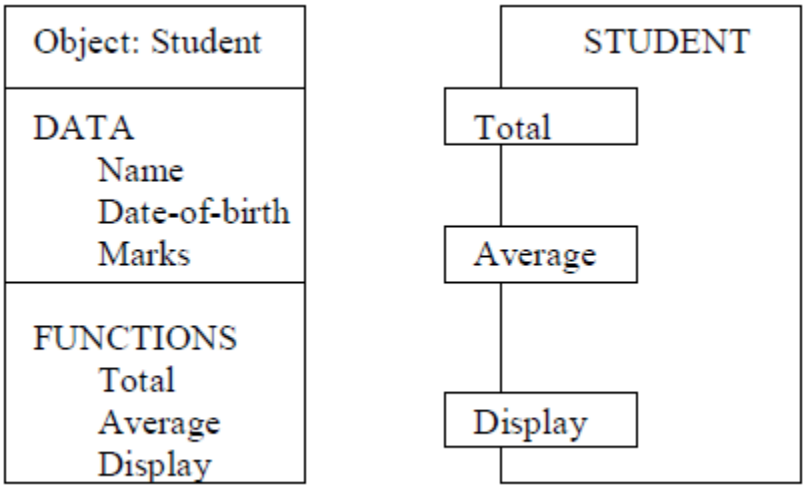
Objects

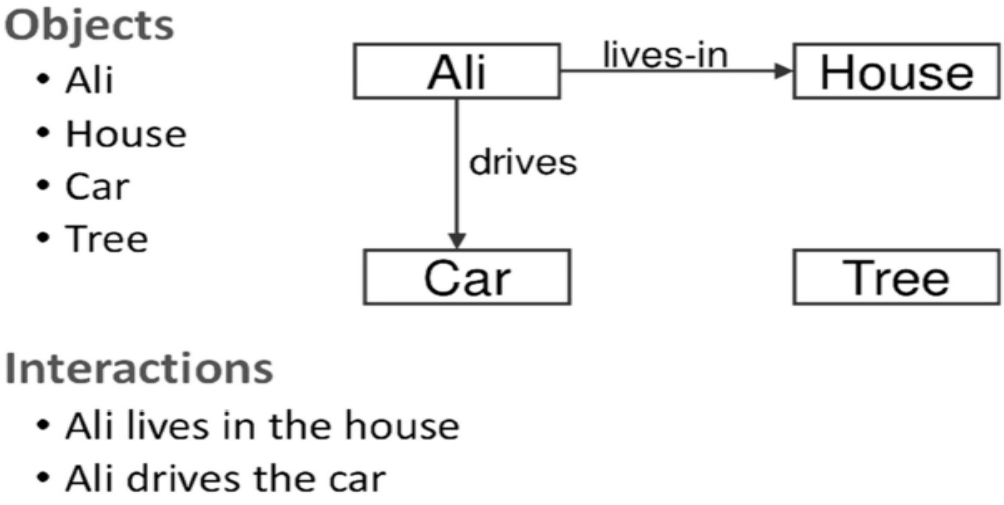
Objects are the basic run-time entities in an object-oriented system. They may represent a person, a place, a bank account, a table of data or any item that the program must handle.

The fundamental idea behind object oriented approach is to combine both data and function into a single unit and these units are called objects.

The term objects means a combination of data and program that represent some real word entity. For example: consider an example named Amit; Amit is 25 years old and his salary is 2500. The Amit may be represented in a computer program as an object. The data part of the object would be (name: Amit, age: 25, salary: 2500)

The program part of the object may be collection of programs (retrive of data, change age, change of salary). In general even any user –defined type-such as employee may be used. In the Amit object the name, age and salary are called attributes of the object.





… What is an Object?

An object has

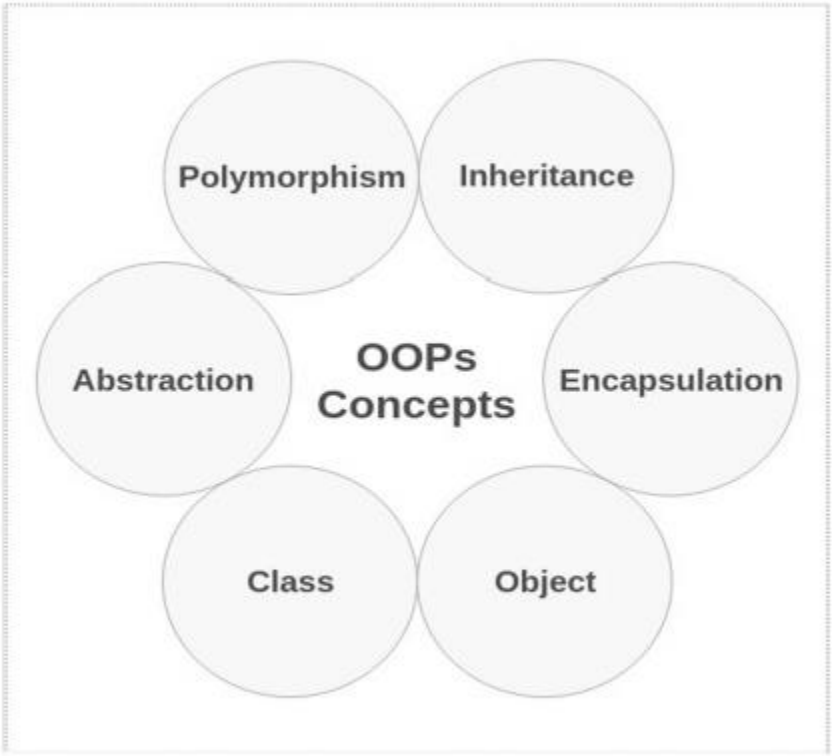
- State (attributes)
- Well-defined behaviour (operations)
- Unique identity

Example – Time is an Object Apprehended Intellectually

- State (attributes)
  - Hours
  - Seconds
  - Minutes
- Behaviour (operations)
  - Set Hours
  - Set Seconds
  - Set Minutes
- Identity
  - Would have a unique ID in the model

Example – Car is a Tangible Object

- State (attributes)
  - Color
  - Model
- Behaviour (operations)
  - Accelerate
  - Start Car
  - Change Gear
- Identity
  - Its registration number

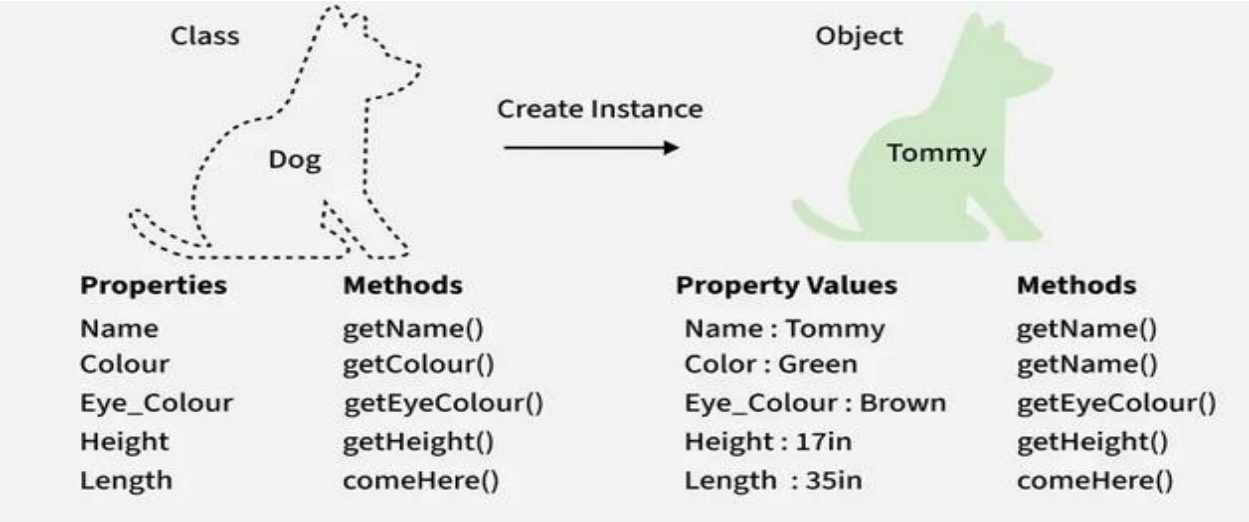


C++ Classes and Objects

Object Oriented Programming, classes and objects are basic concepts of that are used to represent real-world concepts and entities.

A class is a template to create objects having similar properties and behavior, or in other words, we can say that a class is a blueprint for objects.

An object is an instance of a class. For example, the animal type Dog is a class, while a particular dog named Tommy is an object of the Dog class.



C++ Classes

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

Create a Class : A class must be defined before its use. C++ class is defined using the keyword class keyword.

```
#include <iostream>

using namespace std;

// Class definition

class GfG {

public:

    // Data member

    int val;

    // Member function

    void show() {
```

```
        cout << "Value: " << val << endl;
    }
};
int main() {
  // Object will declareed here
    }
```

++ Objects

When a class is defined, only the specification (attributes and behaviour) for the object is defined. No memory is allocated to the class definition. To use the data and access functions defined in the class, we need to create its objects.

Objects are the actual entities that are created as an instance of a class. There can be as many objects of a class as desired. For example, in the above, we discussed the class of Cars. If we create an actual car based on the properties of the Car class, the car we made is the object of that class.

Create Object : Once the class is defined, we can create its object in the same way we declare the variables of any other inbuilt data type.

className objectName;

This statement creates an object of className class.

Member Access : Members of the class can be accessed inside the class itself simply by using their assigned name.

To access them outside, the (.) dot operatoris used with the object of the class.

obj.member1 // For data members

obj.member2(..) // For functions

Difference Between Constructor and Destructor in C++

Constructor

A constructor is a member function of a class that has the same name as the class name. It helps to initialize the object of a class. It can either accept the arguments or not. It is used to allocate the memory to an object of the class. It is called whenever an instance of the class is created. It can be defined manually with arguments or without arguments. There can be many constructors in a class. It can be overloaded but it can not be inherited or virtual. There is a concept of copy constructor which is used to initialize an object from another object.

Syntax:

```
class className{
   // Constructor
   className() {
      // Body of constructor
   }
};
```

Destructor

Like a constructor, Destructor is also a member function of a class that has the same name as the class name preceded by a tilde(~) operator. It helps to deallocate the memory of an object. It is called while the object of the class is freed or deleted. In a class, there is always a single destructor without any parameters so it can't be overloaded. It is always called in the reverse order of the constructor. if a class is inherited by another class and both the classes have a destructor then the destructor of the child class is called first, followed by the destructor of the parent or base class.

Syntax:

```
class className{
   // Destructor
   ~className() {
      // Body of destructor
   }
};
```

Note: If we do not specify any access modifiers for the members inside the class then by default the access modifier for the members will be Private.
Example/Implementation of Constructor and Destructor:

```
#include <iostream>
using namespace std;
class Z
{
public:
   // constructor
   Z()
   {       cout<<"Constructor called"<<endl;    }
```

```
    // destructor
    ~Z()
    { cout<<"Destructor called"<<endl;    }
};
int main()
{
    Z z1;   // Constructor Called
    int a = 1;
    if(a==1)
    {
        Z z2;  // Constructor Called
    }  // Destructor Called for z2
} //  Destructor called for z1
```

Output:

```
Constructor called
Constructor called
Destructor called
Destructor called
```