

## Table of Contents

Table of Contents .....	1
Learning Objectives .....	2
Object-Oriented Approach.....	2
Advantages of Object-Oriented Approach.....	2
Principles of Object Orientation .....	2
Object.....	2
Examples of Objects .....	4
Encapsulation.....	5
Class.....	6
Inheritance.....	6
Polymorphism.....	8
Reference for Further Reading.....	8

## Learning Objectives

After completing this handout, you will be able to:

- Understand object-oriented programming.
- Understand classes and objects.
- Understand principles of object-oriented approach.
- Understand encapsulation, inheritance, and polymorphism.

## Object-Oriented Approach

- Structured programming presented some challenges for which object-oriented programming was designed to solve.
- With object-oriented programming, developers create blocks of code, called objects.
- With the object-oriented approach, you divide an application into many small chunks, or objects, that are fairly independent of one another.
- With the object-oriented approach, we focus on both information and behavior.
- The fundamental idea behind object-oriented languages is to combine into a single unit both data and the functions that operate on that data. Such a unit is called an object.

## Advantages of Object-Oriented Approach

- One of the primary advantages of the object-oriented paradigm is the ability to build components once and then use them over and over again. This is called reusability.

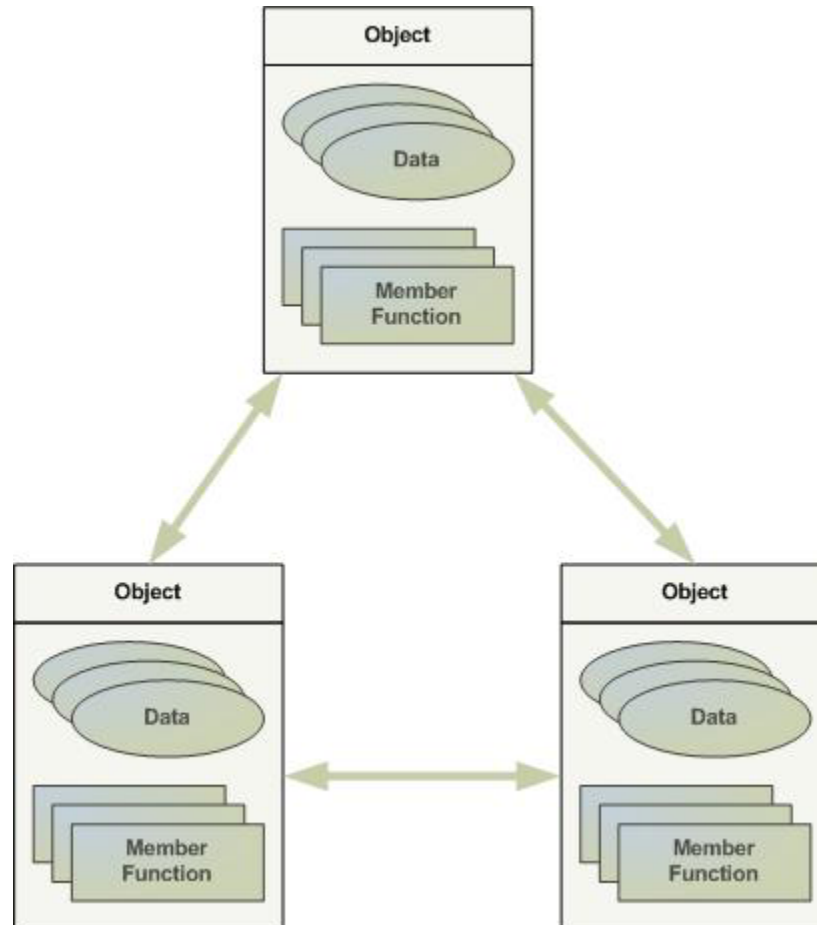
## Principles of Object Orientation

- Encapsulation
- Inheritance
- Polymorphism

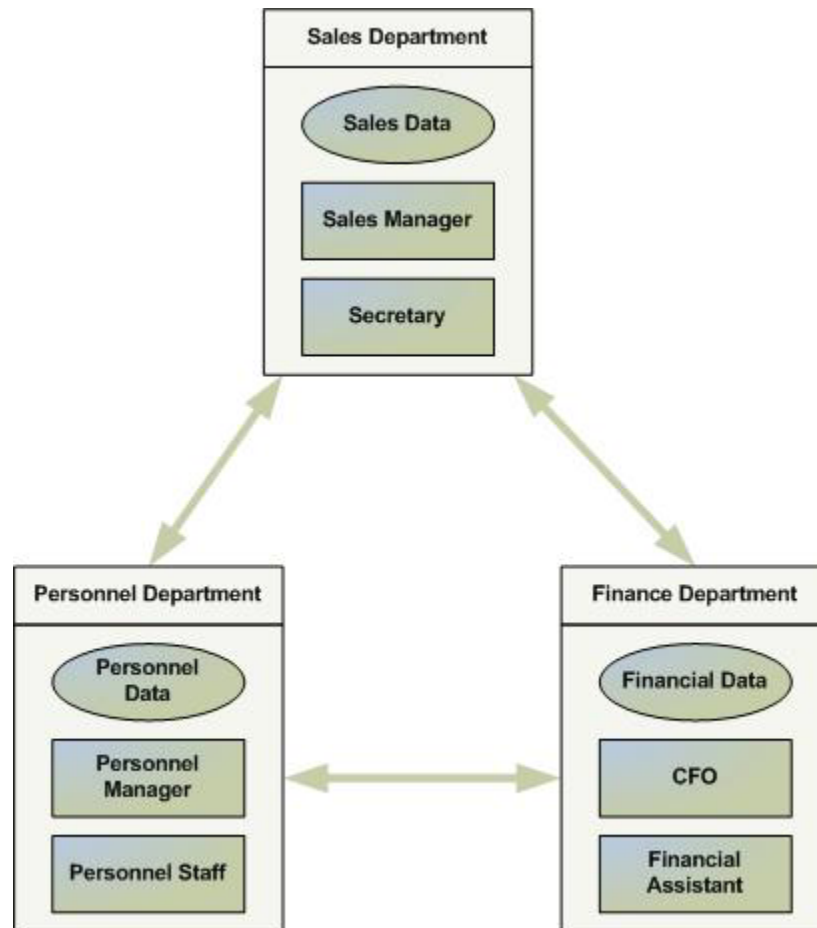
## Object

- An object's functions, called member functions in C++, typically provide the only way to access its data.
- If you want to access a data stored in an object, you call a member function of that object.
- You cannot access the data stored in objects directly.

- The data is said to be hidden inside the object and is safe from accidental alteration.
- A C++ program typically consists of a number of objects, which communicate with each other by calling one another's member functions.



- Think of objects as departments—such as sales, accounting, personnel, and so on—in a company.



- In OOP, the concept of inheritance provides an important extension to the idea of reusability.

### ***Examples of Objects***

- Physical objects
  - Automobiles in a traffic-flow simulation
  - Electrical components in a circuit-design program
  - Countries in an economical model
  - Aircraft in an air-traffic control system
- Elements of the computer-user environment
  - Windows
  - Menus
  - Graphics objects (lines, rectangles, circles)
  - The mouse, keyboard, disk drives, printer
- Data-storage constructs
  - Customized arrays

Stacks

Lined lists

Binary trees

- Human entities

Employees

Students

Customers

Salespeople

- Collections of data

An inventory

A personnel file

A dictionary

A table of the latitudes and longitudes of world cities

- User-defined data types

Time

Angles

Complex numbers

Points on the plane

- Components in computer games

Ghost in a maze game

Positions in a board game (chess, checkers)

Animal in an ecological simulation

Opponents and friends in adventure games

## Encapsulation

- In object-oriented systems, we combine a piece of information with the specific behavior that acts upon that information. Then we package these into an object. This is referred to as encapsulation.
- For example, we have information relating to a bank account, such as the account number, balance, customer name, address, account type, interest rate, and opening date. We also have behavior for a bank account: open, close, deposit, withdraw, change type, change customer, and change address. We encapsulate this information and behavior together into an account object.
- A concept similar to encapsulation is information hiding. Information hiding is the ability to hide the murky details of an object from the outside world.

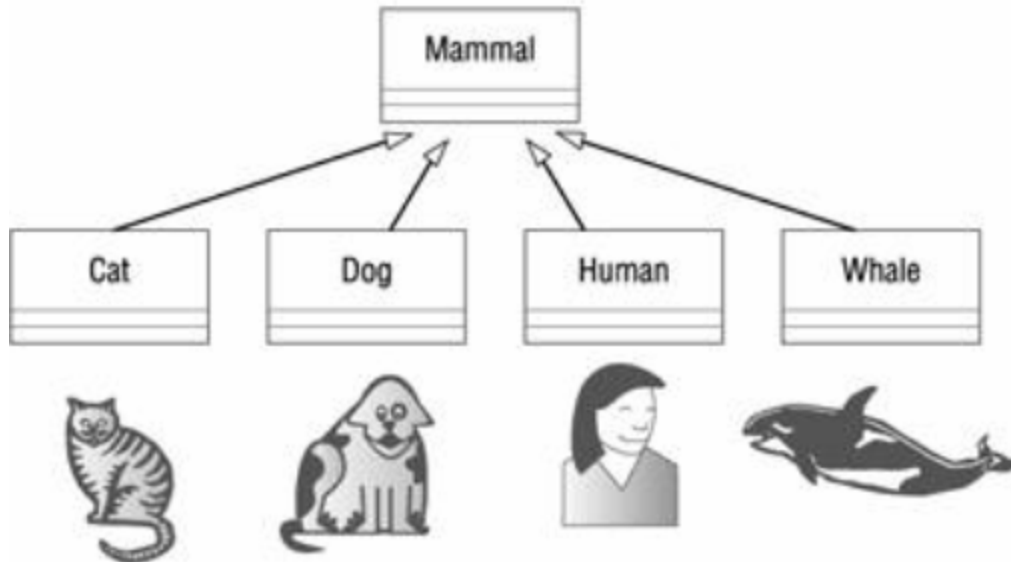
- Data and its functions are said to be encapsulated into a single entity called an object.
- Data encapsulation and data hiding are key terms in the description of object-oriented languages.

## Class

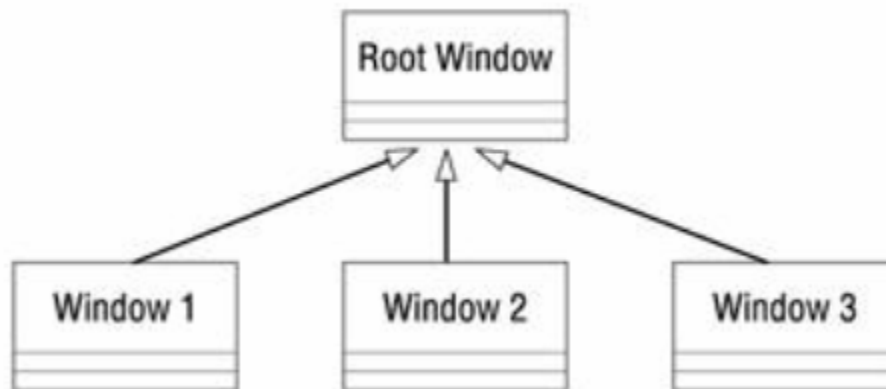
- Objects are members of classes.
- You can define many objects of the same class.
- A class serves as a plan, or template. It specifies what data and what functions will be included in objects of that class.
- Defining a class does not create any object.
- A class is a collection of similar objects.
- Classes lead to inheritance because classes are divided into subclasses. For example class of animals is divided into mammals, amphibians, insects, birds, and so on. Similarly, class of vehicle is divided into cars, trucks, buses, and motorcycles.

## Inheritance

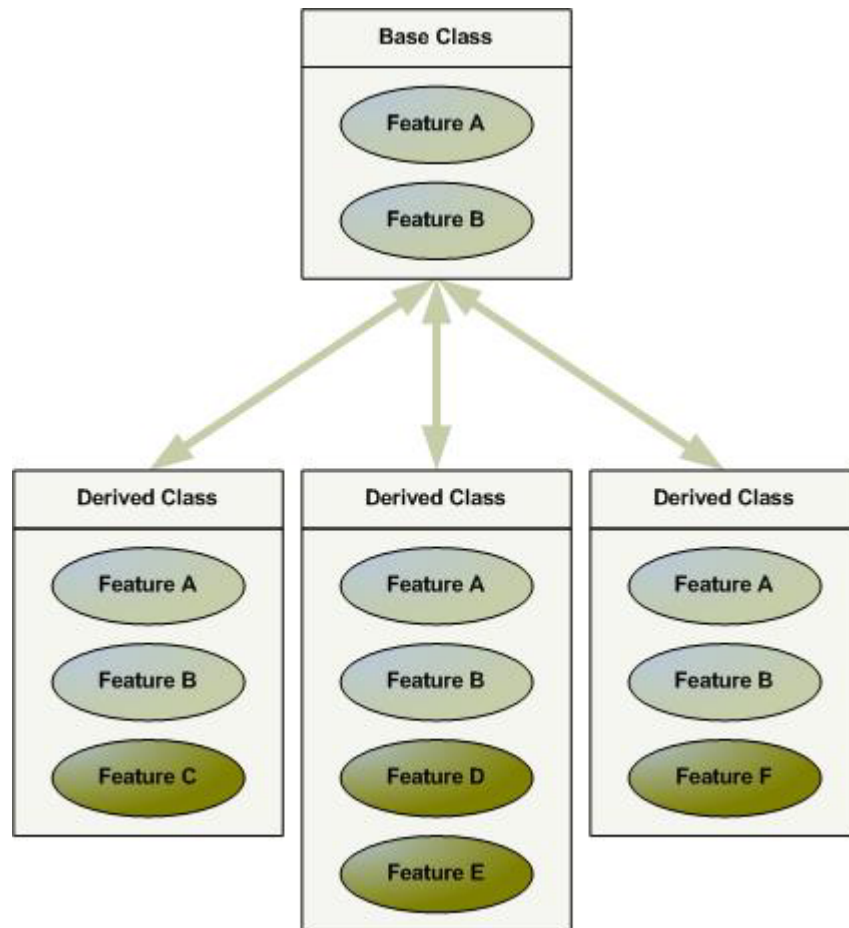
- In object-oriented systems, inheritance is a mechanism that lets you create new objects based on old ones: The child object inherits the qualities of a parent object.
- You can see examples of inheritance in the natural world. There are hundreds of different types of mammals: dogs, cats, humans, whales, and so on. Each of these has certain characteristics that are unique and certain characteristics that are common to the whole group, such as having hair, being warm-blooded, and nurturing their young.
- In object-oriented terms, there is a mammal object that holds the common characteristics. This object is the parent of the child objects cat, dog, human, whale, etc. The dog object inherits the characteristics of the mammal object, and has some additional dog characteristics of its own, such as running in circles and slobbering.



- One of the major benefits of inheritance is ease of maintenance. When something changes that affects all mammals, only the parent object needs to change—the child objects will automatically inherit the changes.
- In an object-oriented system, an example of inheritance might be in the windows.



- In a banking system, we might use inheritance for the different types of accounts we have.
- In C++ the original class is called the base class; other classes can be defined that share its characteristics, but add their own as well. These are called derived classes.
- Each subclass shares common characteristics with the class from which it's derived.
- In addition to the characteristics shared with other members of the class, each subclass also has its own particular characteristics.



## Polymorphism

- The dictionary defines it as the occurrence of different forms, stages, or types.
- Polymorphism means having many forms or implementations of a particular functionality. In terms of an object-oriented system, this means that we can have many implementations of a particular functionality.

## Reference for Further Reading

- Mastering UML with Rational Rose 2002, Wendy Boggs and Michael Boggs, Chapter 1.
- Object-Oriented Programming in C++, Second Edition, Robert Lafore, Chapter 1.