C++ & Object Oriented Programming

Dr. Alekha Kumar Mishra

Prerequisites

- Skill of developing programming logic
- Programming ability in C language.
- Implementation ability of data structures in C language.
- Operating system(commands and editors),

Text books

- Schildt : “C++ : The Complete Reference”, 3rd Ed., Tata McGraw Hill Publication.
- Object Oriented Programming with C++ - E. Balagurusamy, McGraw-Hill Education (India)
- Deitel & Deitel : “C++ : How to Program”, 3rd Ed., Pearson Education Pvt. Ltd.
- Lafore R., “Object Oriented Programming in Turbo C++”, Galgotia Publication, New Delhi, 1995.

Evaluation

- Theory
  - Mid semester exam. : 30 marks
  - End semester exam. : 50 marks
  - Internal : 20 marks
    - Attendance : 5 marks
    - Programming / Development task : 15 marks
- Laboratory
  - End semester lab test : 30
  - End semester viva : 10
  - Regular assignment : 40
  - Attendance : 10
  - Record maintance : 10

About the course

- C++ as the vehicle for illustrating and implementing Object Oriented concepts.
- Object-oriented paradigm is applied in the design of programming using the language C++

Why Object-Oriented Progamming??

- Object-oriented programming was developed because limitations were discovered in earlier approaches to programming

Procedural Languages

- Each statement in this language tells the computer to do something:
  - Get some input,
  - add these numbers,
  - divide by six,
  - display that output.
- A program in a procedural language is a **list of instructions.**
- When programs become larger, a single list of instructions becomes unwieldy.

Procedural Languages(2)

- Function was adopted as a way to make programs more comprehensible
  - also may be referred to as a subroutine, a subprogram, or a procedure in other languages
- A procedural program is **divided into functions**, and (ideally, at least) each function has a clearly defined purpose and a clearly defined interface to the other functions in the program.
- C, Pascal, FORTRAN, and similar languages are procedural languages.

Structured programming

- Grouping a number of functions together into a larger entity called a module (which is often a file)
- Dividing a program into functions and modules is one of the cornerstones of **structured programming**
- Structured programming influenced programming organization for several decades before object-oriented programming.

Problems with Structured Programming

- Increase in complexity as the program grows larger
- Functions have unrestricted access to global data
- Functions and data are unrelated.
- A change made in a global data item may necessitate rewriting all the functions that access that item.
- Procedural paradigm provides a poor model of the real world

Real-world modeling

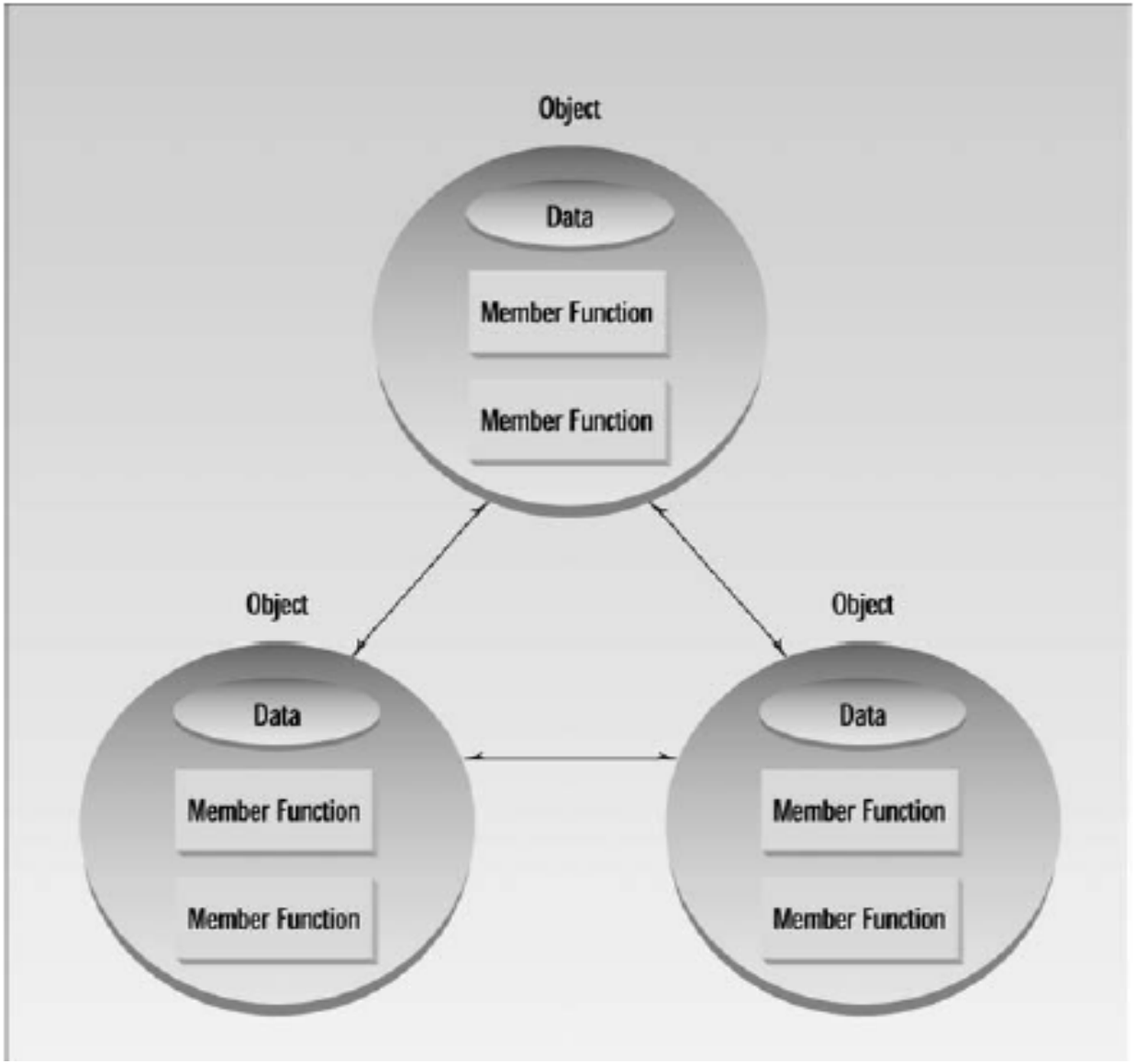
- In the physical world we deal with objects such as people, institution, and cars.
- These are not like data and they aren't like functions too.
- Complex real-world objects have both attributes and behavior.
- Attributes sometimes called characteristics in the real world are equivalent to data in a program: they have a certain specific values
- Behavior is something a real-world object is like a function: you call a function to do something (display the inventory, for example).
- So neither data nor functions, by themselves, model real-world objects effectively.

The Object-Oriented Approach

- The fundamental idea : "**combine both data and the functions into a single unit that operate on its data**".
- Such a unit is called an **object**.
- An object's functions, called member functions in C++, typically provide the only way to access its data.
- The data is hidden, so it is safe from accidental alteration.
- Data and its functions are said to be encapsulated into a single entity.
- Data encapsulation and data hiding are key terms in the description of object-oriented languages.
- This simplifies writing, debugging, and maintaining the program.

Object-oriented program

- A C++ program typically consists of a number of objects, which communicate with each other by calling one another's member functions.



Object

- Things having physical or logical existence
  - Physical objects
  - Elements of the computer-user environment
  - Data-storage constructs
  - Human entities
  - Collections of data
  - User-defined data types
  - And so on...

Class

- Objects are member of classes.
- Defining the class doesn’t create any objects
- A class is thus a description of a number of similar objects.
  - Himesh, Vishal, and Ankit are members of the music composer.
- There is no one person called “music composer,” but specific people with specific names are members of this class if they possess certain characteristics.
- An object is often called an “**instance**” of a class.

Characteristics of Object-Oriented Languages

- Data abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Dynamic binding
- Reusability

Data abstraction

- Abstraction refers to the act of showing essential features without including its implemetation details.
- It focuses on the outside view(interface) of an object. Therefore, it serves to separate an objects essential behaviour from its implementation.
- It is used to reduce complexity by ignoring some aspect of a subject that are not relevant to the current purpose.
- The type that uses data abstraction are known as Abstract Data Type.

Example : Data abstraction

- InstantChatApp
  - Name
  - Version
  - Device Requirement
  - OSversion
  - Updated version
  - ...
  - VoiceCall()
  - TakePhoto()
  - VideoCall()
  - SendMessage()
  - UploadDoc()
  - UploadMedia()
  - ...

Encapsulation

- The binding of data and functions into a single unit is know as encapsulation.
- Encapsulation achieves information hiding ; hiding all the information that is unnecessary to the outside world
- Each abstraction have two parts : an interface and an implementation.
- Encapsulated elements may be termed as the secrets of abstraction

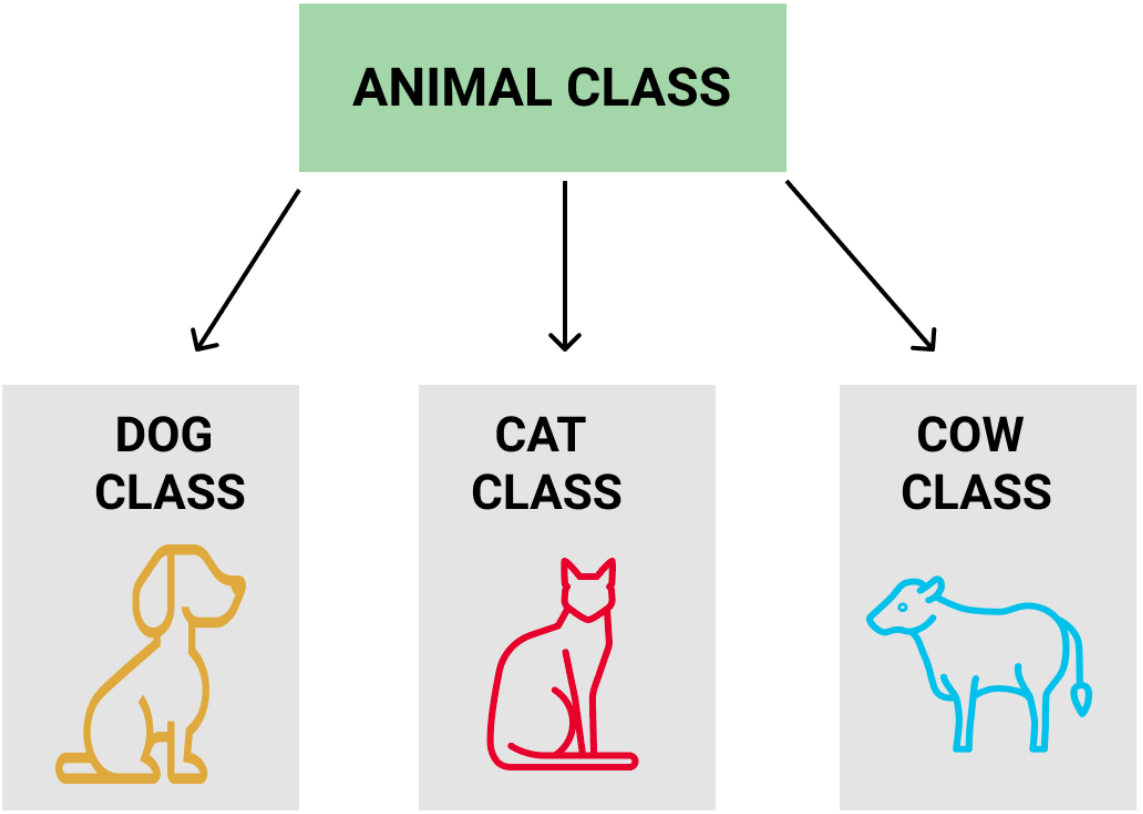
Example : Encapsulation

- Class Triangle {
  side1len;
  side2len;
  side3len;
  ...
  calArea();
  calPermeter();
  isRightAngled();
  ....
  }

Inheritance

- It is the process by which objects of one class acquire the properties of objects of another class.
- It supports the concept of **is-a** kind of hierarchical classification
- Each derived class shares common characteristics with the class from which it is derived.
- Provides the idea of reusability

Example : Inheritance



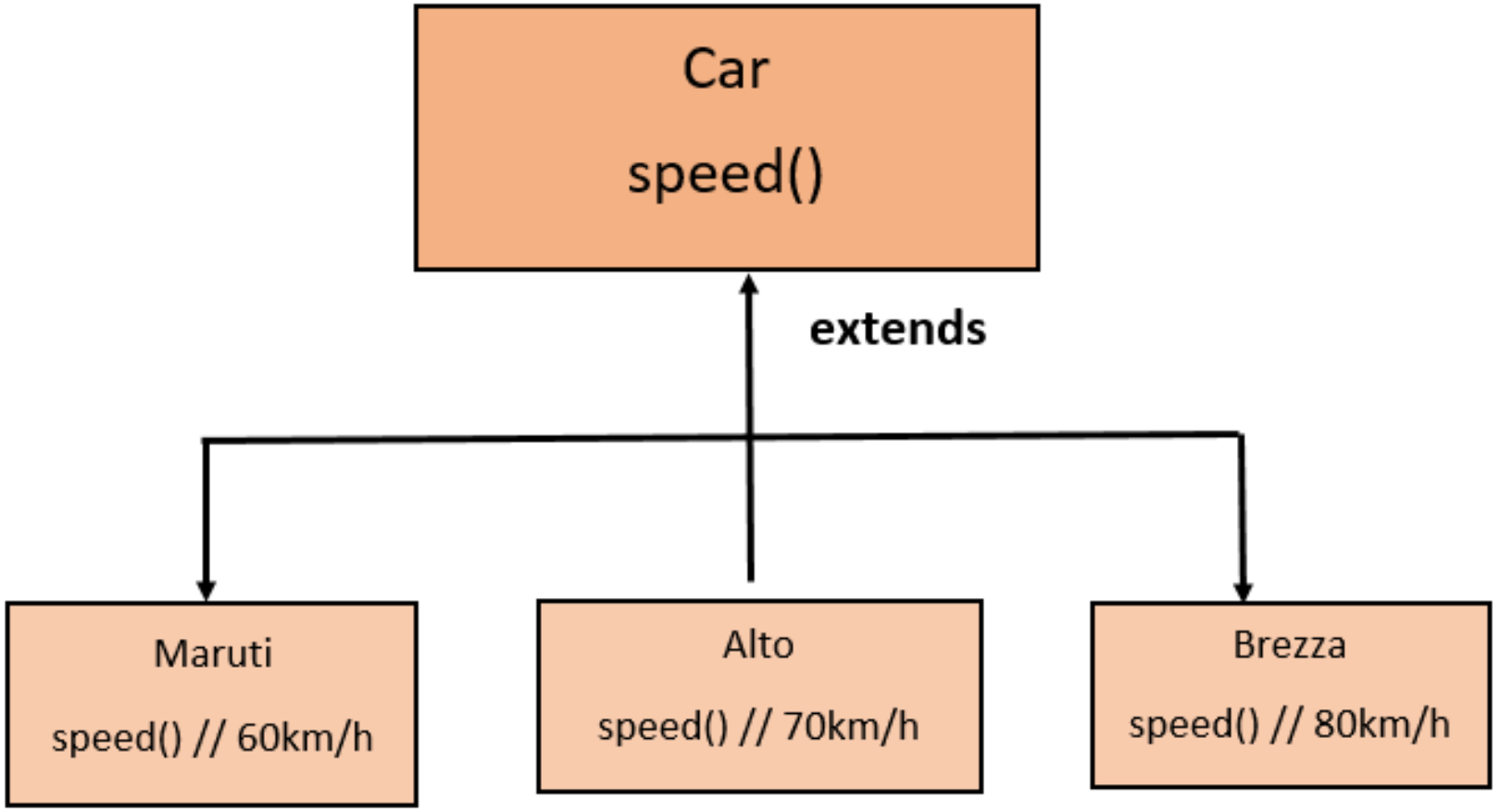
Polymorphism

- “ability to take more than one form”
- An operation may perform different task in different instances.
- The behaviour depends upon the types of data used in the operation.
- Supports operator overloading in c++
- Using single function name to perform different types of tasks is known as function overloading

Dynamic binding

- Here, binding refers to the linking of precedure call to the code to be executed in reponse to the call.
- Code associated with a given message will be decided at run time,i.e., at the time of call
- Helps achieving dynamic polymorphism also called runtime-polymorphism

Polymorphism : Example



Benifits of OOP

- Elimination of redundant code via inheritance
- Saves development time
- Data restricted from unauthorized access
- Easily scalable
- Interface between objects is simpler
- Lesser complexity
- Encourages reusability

OOP languages

- Java, C++, C#, Python, PHP, JavaScript, Ruby, Perl, Object Pascal, Objective-C, Dart, Swift, Scala, Common Lisp, MATLAB, Smalltalk and the list goes on...