Unit 2

Object Oriented Concepts

1. Overview of Procedure Oriented Programming

In Procedure oriented programming, a program is considered as a "sequence of tasks to be done". Hence, the emphasis is on the procedure or algorithm. To manage the complexity of the task, usually a modular approach is used. As a result, not enough attention is given to data. If the same data is needed by many fun ctions, programmers tend to use global data. This affects the security of data. Such programs use the **top-down** approach. Programs written using most high level languages such as COBOL, FORTRAN and C use the **procedure-oriented** approach.

A typical program structure for procedural programming is shown in the following *figure*.

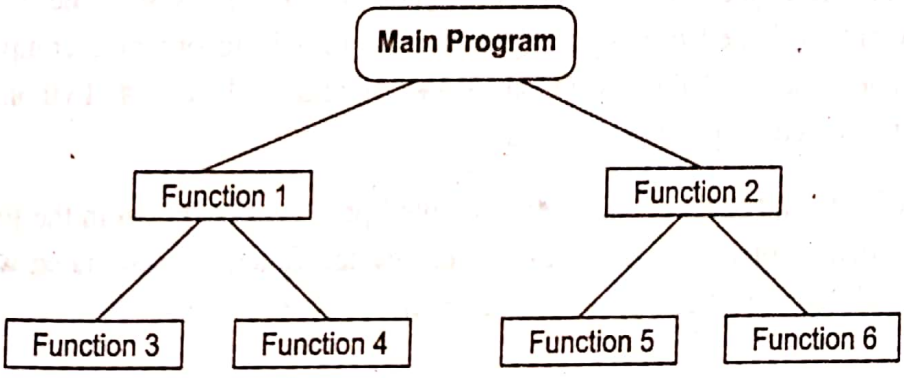

Figure 2.1: Typical Structure of Procedure-Oriented Programs

▶ Features of Procedure-Oriented Programming

i. Focus is on the functions.

ii. It follows Top-Down approach.

iii. The program consists of multiple functions.

iv. Most of the functions share global data.

v. Functions transform data from one form to another.

vi. Data is not hidden and can be easily shared.

▶ Disadvantages of Procedure-Oriented Programming

i. Due to the use of global data, it is very difficult to identify which data is used by which function. Moreover, if the data structure changes, we need to modify all functions that access the data.

ii. The procedure-oriented programming approach does not model real world problems very well. This is because functions are action-oriented and do not correspond to the entities of the problem.

iii. Beyond a certain size, procedure oriented programming cannot manage the complexity of the problem.

iv. Procedure-oriented programming fails to address important issues such as reusability, maintainability, portability, security, integrity, etc.

2. Object Oriented Programming

The term "Object-Oriented" means organizing software as a collection of objects which consist of both, data and behavior. Objects model real world entities in the software. These objects interact with each other. Object oriented programming makes it possible to organize complex code using classes, objects, inheritance and polymorphism. C++, Smalltalk, Java, C#, Python, PHP etc. are examples of object-oriented programming languages.

The organization of data and functions in object-oriented programs is shown in the following *figure*. The data members of an object can be accessed only by the functions associated with that object. However functions of one object can access the functions of other objects.

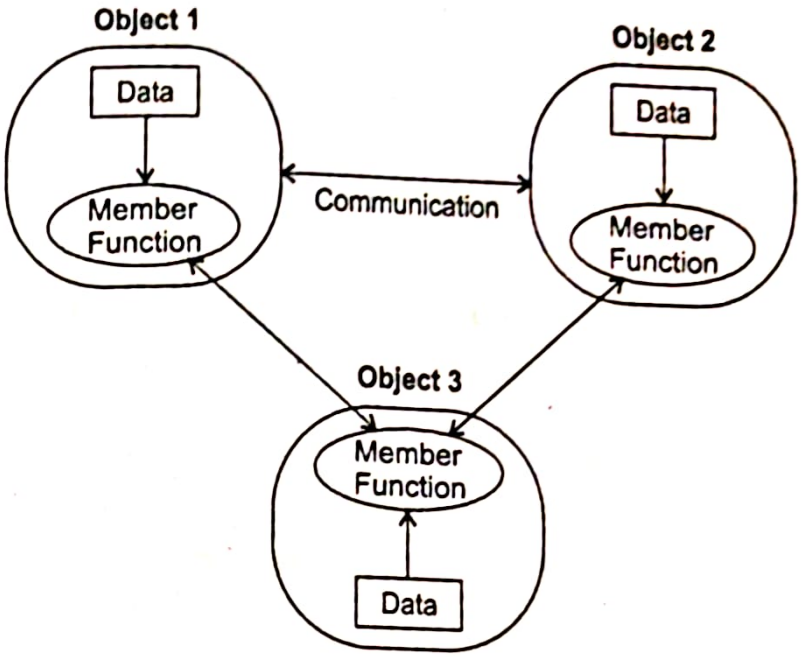

Figure 2.2: Structure of Object-Oriented Programming

▶ Features of Object-Oriented Programming

i. Focus is on data rather than procedures or functions.

ii. A program consists of different objects.

iii. The object's data and functions that operate on the data, are tied together.

iv. Data is hidden and can only be accessed through the object's member functions.

v. Objects can pass messages to each other through functions.

vi. New data and functions can be easily added whenever necessary.

vii. It follows bottom-up approach.

3. Difference between Procedure-oriented Programming and Object-oriented Programming

	Procedure-oriented Programming	Object-oriented Programming
1.	Focus is on procedures or functions rather than data.	Focus is on data rather than procedures or functions.
2.	A program consists of a set of functions.	A program consists of a set of objects.
3.	It follows Top-Down approach.	It follows Bottom-up approach.
4.	All functions can access the global data.	Data is hidden and can only be accessed through the object's member functions.
5.	Less data security.	More data security.
6.	Difficult to make modifications to existing functions or add new features.	New data and functions can be easily added whenever necessary.
7.	Difficult to model real-world, large-scale, complex problems.	Large-scale and complex real-world problems can be easily modeled using object-oriented programming.

4. Object-Oriented Concepts

The following terms are important in object-oriented programming.

i. **Object:** An object is a software entity that models a real life entity in an object-ori program. Each object has some characteristics or attributes and behavior. Objects are the run-time entities in an object oriented system.

For example, an object could represent a person, a student, an account etc.

An object consists of two parts:

a. Data elements, referred to as *attributes.*

b. Processes that may change the attributes, called *functions* or *methods.*

The following diagram shows an object named "Ann" whose attributes are id, name, clas percentage and its functions are accept_details(), display_details() and calculate_grade().

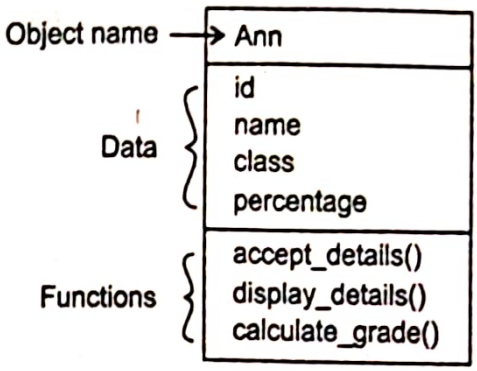

Figure 2.3: Example of an Object

ii. **Class:** A class represents a set of objects that share common characteristics and behavior. A class is a user defined **type** which consists of data members and member functions of an object. Objects are **instances** of the class. This is analogous to data type and variable. Thus, a class is considered as a blue-print for an object.

For example: Consider a class called "Student" having attributes id, name, class and percentage and its functions are accept_details(), display_details() and calculate_grade() and whose objects are Ann, Bob etc.

iii. **Encapsulation and data hiding:** The binding of data members and functions into a single unit is called as encapsulation.

Due to encapsulation, the data members of the class cannot be accessible directly outside the class. Only member functions have the right to access the data members. This insulation of the data from direct access by the program is called as **data hiding** or information hiding. This improves data security.

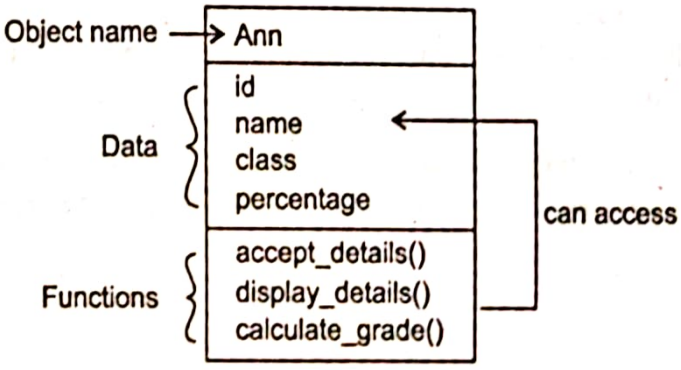

Figure 2.4: Encapsulation and Data Hiding

iv. **Data abstraction:** *Abstraction* refers to the act of representing essential features without including the background details or explanations. Data abstraction separates the properties and behavior of a class and its implementation.

For example, we can define an Abstract Data type called Dictionary which has words and its corresponding meaning. The operations are Add(), Remove() and Search(). Such a dictionary

may be implemented in many ways – using a simple list, hash table or binary search tree. Th end user need not know the implementation details.

v. **Inheritance:** Inheritance is the ability to create a new class from an existing one. It allow classes to be reused. The new class inherits properties from the old one. The parent class i called the **base class** while the child class is called the **derived class.**

Example: In the *figure* given below, Employee is the base i.e. parent class. Two classes namely Full-Time and Part-Time are derived from Employee class. Further, Permanent an Temporary are derived from Full-Time class.

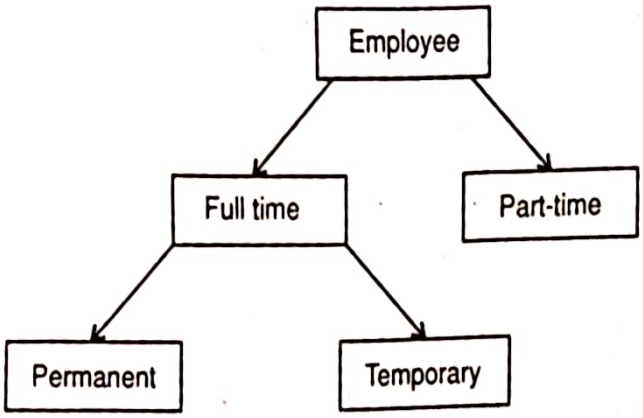

Figure 2.5: Inheritance

The concept of inheritance provides the idea of reusability and extensibility i.e. we can ad additional features to an existing class without modifying it. This is possible by deriving a new class from the existing one and add more features to it.

vi. **Polymorphism (Poly - many and morph - Form):** Polymorphism is the ability to tak more than one form. *For example*, consider the operation of addition. If we add two numbers the operation will generate a sum. If the operands are strings, then the result can b concatenation of the two strings. Thus, the operation exhibits different behavior at differen times depending on the context.

The following *figure* illustrates that a function named "draw" can be used to draw differen shapes. Depending on the object which calls the function, the corresponding draw functio will be called.

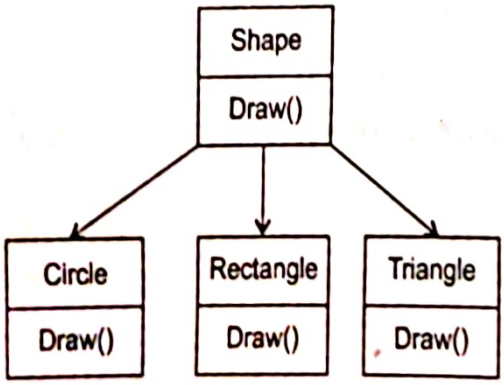

Figure 2.6: Polymorphism

Types of Polymorphism

1. *Compile Time Polymorphism:* Linking the function call to its code during compilation time is called as *early binding* or *static binding* or *static linkage*. During compilation time, the C++ compiler determines which function is to be used on the basis of parameters passed to the function. The compiler then substitutes the correct function for each call. Such compiler based substitutions are called as static linkage. By default, C++ follows early binding. With early binding, one can achieve greater efficiency. Function calls are faster in this case because all the information necessary to call the function is known beforehand.

   Compile-time polymorphism is achieved in two ways:

   a. Operator Overloading: Using the same operator to perform different tasks is known as *operator overloading*. *For example*, the operator + can be overloaded for adding strings and objects like matrices, fractions etc.

   b. Function Overloading and Overriding: Using a single function name to perform different types of tasks is known as *function overloading*. *For example*, the function area() can be overloaded to calculate the area of circle, rectangle etc.

      Function Overriding means redefining a base class function in the derived class. *For example*, a function calculate_wages() of the Employee class can be redefined in the Full-Time and Part-Time class.

2. *Run-time Polymorphism:* Choosing functions during execution time is called as *late binding* or *dynamic binding* or *dynamic linkage*. The linking of the function call to the function code is done during runtime. Which function to call is decided on the basis of the object which is created during runtime. Late binding is implemented through virtual functions. These functions are called using pointers.

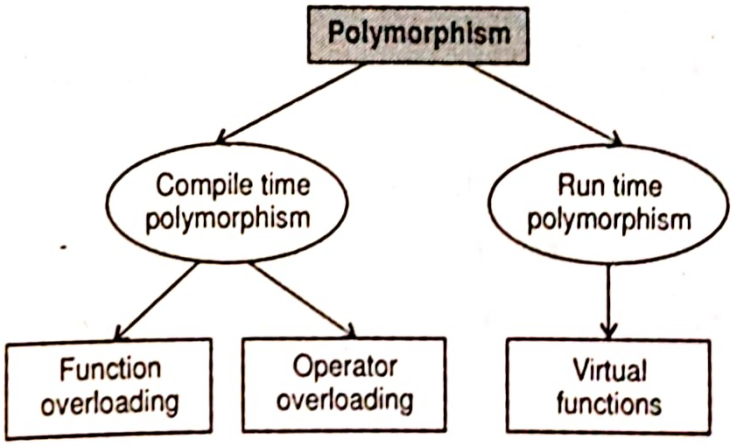

Figure 2.7: Types of polymorphism

vii. **Message passing**: When an object oriented program is executed, the objects communicate with each other by sending messages to one another. Objects can interact without knowing the details of each other's data or code. It is sufficient to know the type of message accepted and the type of response returned by the objects. Message passing consists of calling a method of an object, giving it the information it requires, and receiving a reply from the method. Message passing can be something as simple as a function call. The syntax is given below:

Syntax: `object-name.function-name(information);`

Example: `dictionary.search(word);`

where, *dictionary* is the object name, *search* is the message and *word* is the information.

5. Advantages of Object-Oriented Programming

i. **Simplicity:** Objects in an object oriented program model real world entities, so it is easier to solve real world problems.

ii. **Modularity:** Each object forms a separate entity whose internal structure is hidden from other parts of the system.

iii. **Modifiability:** It is easy to make changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the external world can access the class only through the use of functions which is its public interface.

iv. **Extensibility:** Adding new features or responding to changing requirements can be solved by introducing new classes using existing ones. Object-oriented systems can be easily upgraded from small to large systems.

v. **Maintainability:** Objects can be maintained separately which makes it easy to locate and correct problems.

vi. **Re-usability:** Objects can be reused in different programs.

vii. **Flexibility**: Polymorphism enables an object to behave differently according to the context.

viii. **Security:** Due to encapsulation, data is insulated from external access. This avoids misuse of data.

6. Applications of OOP

OOP is beginning to gain importance in many areas. Due to the advantages that OOP offers, it has become the most commonly used programming methodology for large and complex applications. Some of these applications areas are listed below.

1. Developing Graphical User Interfaces
2. System Applications
3. Object Oriented Database Systems
4. Expert Systems
5. Simulation and Modeling software
6. Artificial Intelligence
7. Compiler Construction.
8. Communication and Networking software
9. CAD/CAM Software
10. Operating Systems

EXERCISES

A. Multiple Choice Questions.

1. The binding of data and functions into a single unit is called as
   a. Class b. Dynamic binding
   c. Encapsulation d. Inheritance
2. The act of communicating with an object to get something done is called as
   a. Message passing. b. Polymorphism
   c. Inheritance d. Dynamic binding
3. In case of OOP the focus is on
   a. Data b. Procedures
   c. Inheritance d. Functions
4. In case of Procedure Oriented Programming, the focus is on

a. Data b. Functions

c. Objects d. Messages

5. Exhibiting different behavior at different times is called

a. Inheritance b. Encapsulation

c. Polymorphism d. Data hiding

6. The process of building a new class from existing ones is called

a. Inheritance b. Encapsulation

c. Polymorphism d. Data hiding

7. The insulation of data from direct access by unauthorized functions is called

a. Inheritance b. Data hiding

c. Polymorphism d. Message Passing

B. Review Questions.

1. What is procedure oriented programming? State its features.
2. What is object-oriented programming?
3. Explain the important properties of Object Oriented Programming.
4. List the differences between OOP and Procedure oriented programming.
5. What are the advantages and applications of OOP?
6. Write a short note on polymorphism,