Object-oriented programming (OOP) is a programming paradigm that uses objects, which are instances of classes, to organize code. The OOP paradigm is based on several key features that facilitate the design, implementation, and maintenance of software systems.

The main features of the Object-Oriented Paradigm:

1. Objects:

- Definition: Objects are instances of classes and represent real-world entities or concepts.
- Characteristics: Each object has attributes (data) and behaviors (methods/ functions).

2. Classes:

- Definition: Classes are blueprints or templates that define the structure and behavior of objects.
- Characteristics: A class encapsulates data and methods that operate on that data.

3. Encapsulation:

- Definition: Encapsulation is the bundling of data (attributes) and methods that operate on the data into a single unit (class).
- Purpose: It hides the internal details of an object and restricts access to only what is necessary.

4. Abstraction:

- Definition: Abstraction involves simplifying complex systems by modeling classes based on essential properties and behaviors, ignoring unnecessary details.

- Purpose: It focuses on what an object does rather than how it does it, providing a clear and abstract representation.

5. Inheritance:

- Definition: Inheritance allows a new class (subclass or derived class) to inherit attributes and behaviors from an existing class (base class or parent class).
- Purpose: It promotes code reuse, extensibility, and the creation of a hierarchy of classes.

6. Polymorphism:

- Definition: Polymorphism allows objects of different types to be treated as objects of a common type.
- Types: Compile-time polymorphism (method overloading) and runtime polymorphism (method overriding).
- Purpose: It enhances flexibility, allowing a single interface to represent multiple underlying forms.

7. Message Passing:

- Definition: Objects communicate and interact by sending and receiving messages.
- Purpose: Enables collaboration between objects, facilitating the execution of methods and the exchange of information.

8. Dynamic Binding:

- Definition: Dynamic binding allows the association between a method call and the method implementation to be resolved at runtime.

- Purpose: Enhances flexibility and enables late binding, where the specific method to be executed is determined during program execution.

9. Association:

- Definition: Association represents a relationship between two or more objects.
- Types: Aggregation (weaker association) and Composition (stronger association).

Related Posts:

1. Abstraction and encapsulation
2. Object Oriented Programming & Methodolog Viva Voce
3. How to install compiler for code blocks
4. Object Oriented Programming
5. Differences between Procedural and Object Oriented Programming
6. Features of Object Oriented Paradigm
7. Inheritance in Object Oriented Programming
8. Object Oriented Programming
9. Introduction to Object Oriented Thinking & Object Oriented Programming
10. Difference Between Object-Oriented Programming (OOP) and Procedural Programming
11. Merits and demerits of Object Oriented methodology
12. Concept of Objects: State, Behavior & Identity of an object
13. Access modifiers
14. Static members of a Class
15. Instances in OOP
16. Message Passing in OOP
17. Construction and destruction of Objects