Object Oriented Programming

1. **Object:**
   * Objects are instances of classes and represent real-world entities.
   * They encapsulate data (attributes) and behavior (methods) that operate on the data.
2. **Class:**
   * A class is a blueprint or template for creating objects.
   * It defines the structure (attributes) and behavior (methods) that its objects will have.
3. **Encapsulation:**
   * Encapsulation is the bundling of data and methods that operate on that data into a single unit known as a class.
   * It hides the internal details of an object and exposes only what is necessary.
4. **Inheritance:**
   * Inheritance is a mechanism that allows a class (subclass/derived class) to inherit properties and behaviors from another class (superclass/base class).
   * It promotes code reuse and establishes a hierarchy of classes.
5. **Polymorphism:**
   * Polymorphism means "many forms" and is achieved through method overloading and method overriding.
   * Method overloading: Same method name with different parameters.
   * Method overriding: Providing a specific implementation of a method in a subclass.
6. **Abstraction:**
   * Abstraction involves simplifying complex systems by modeling classes based on the essential features.
   * It hides the implementation details and only exposes what is necessary for the outside world.
7. **Instance:**
   * An instance is a specific realization of an object created from a class.
   * Each instance has its own set of data values, but shares the structure and behavior defined by the class.
8. **Constructor:**
   * A constructor is a special method in a class used for initializing the object.
   * It is called when an object is created and typically sets initial values for the object's attributes.
9. **Destructor:**
   * In OOP languages like Java, the concept of a destructor is not as explicit as in some other languages.
   * Automatic garbage collection is used to reclaim memory, and there is no explicit destructor.
10. **Method:**
    * A method is a function associated with a class and is used to define the behavior of the objects.
    * Methods operate on the data within the class and can be called by instances of that class.
11. **Composition:**
    * Composition is a way to combine objects to create more complex ones.
    * It involves creating instances of other classes within a class and using them to achieve complex behaviors.
12. **Association:**
    * Association represents a relationship between two or more classes.
    * It can be a one-to-one, one-to-many, or many-to-many relationship.
13. **Aggregation:**
    * Aggregation is a special form of association where one class is a part of another class.
    * It represents a "whole-part" relationship.
14. **Message Passing:**
    * Objects communicate with each other by sending messages.
    * Messages consist of method calls, and objects interact by invoking each other's methods.
15. **Static and Instance Members:**
    * Static members (variables or methods) belong to the class and are shared among all instances.
    * Instance members are specific to each instance of the class.

Understanding and applying these OOP concepts allows developers to design and implement software in a modular, reusable, and maintainable way.