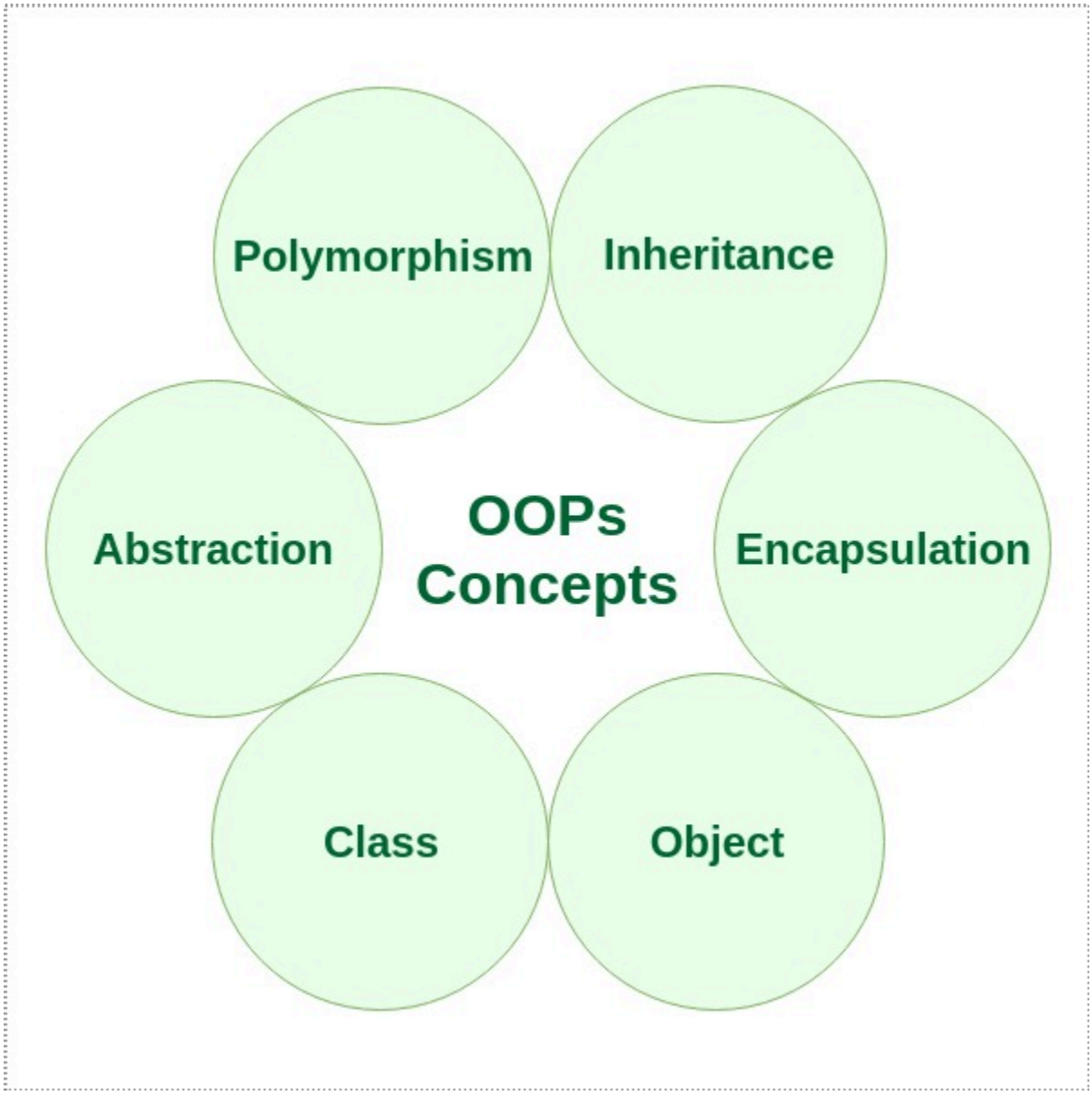
OOPS

Object Oriented Programming (OOP) is a style of programming that uses objects to model real-world things like data and behavior. It focuses on key ideas like **inheritance**, **encapsulation**, and **polymorphism**. The main goal of OOP is to group data and the functions that work on it together, so that the data is protected and can only be changed in controlled ways.

Characteristics of an Object-Oriented Programming Language

There are some basic concepts that act as the building blocks of OOPs i.e.



Class

In C++, the basic building block of Object-Oriented Programming is the class.
A class is a user-defined type that acts like a blueprint to create objects that share similar properties (data) and behaviors (functions).
For example, you can define an **Animal** class with properties like **name**, **age**, and **species**, and behaviors like **eat()**, **sleep()**, and **makeSound()**.
A class is defined using the **class** keyword in C++.

Object

An object is a real, usable instance of a class that has specific properties and behaviors. In C++, an object is created from a class.
For example, Animal is just an idea or blueprint, but a **cat** is a real object based on that class. So, classes are concepts, and objects are the actual things created from those concepts.
When you define a class, no memory is used. But when you create an object from that class, memory is allocated for it

Objects take up space in memory and have an associated address like a record in pascal or structure or union. When a program is executed, the objects interact by sending messages to one another. Each object contains data and code to manipulate the data. Objects can interact without having to know 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

Encapsulation

In simple terms, encapsulation is defined as wrapping up data and information under a single unit. In Object-Oriented Programming, encapsulation is defined as binding together the data and the functions that manipulate them together in a class. Consider an example of the Animal class, the data members species, age and name are encapsulated

with the member functions like eat(), sleep, etc. They can be protected by the access specifier which hides the data of the class from outside.

Abstraction

Abstraction is one of the most essential and important features of object-oriented programming in C++. Abstraction means displaying only essential information and ignoring the other details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. In our case, when we call the `makeSound()` method, we don't need to know how the sound is produced internally, only that the method makes the animal sound

Polymorphism

The word polymorphism means having **many forms**. In simple words, we can define polymorphism as the ability of an entity to behave different in different scenarios. person at the same time can have different characteristics.

For example, the same **makeSound()** method, the output will vary depending on the type of animal. So, this is an example of polymorphism where the makeSound() method behaves differently depending on the Animal type (Dog or Cat).

Inheritance

The capability of a class to derive properties and characteristics from another class is called Inheritance. Inheritance is one of the most important features of Object-Oriented Programming.

- **Sub Class**: The class that inherits properties from another class is called Sub class or Derived Class.
- **Super Class**: The class whose properties are inherited by a sub-class is called Base Class or Superclass.

Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

Advantage of OOPs

Here are key advantages of Object-Oriented Programming (OOP) over Procedure-Oriented Programming (POP):

- **Modularity and Reusability**: OOP promotes modularity through classes and objects, allowing for code reusability.
- **Data Encapsulation**: OOP encapsulates data within objects, enhancing data security and integrity.
- **Inheritance**: OOP supports inheritance, reducing redundancy by reusing existing code.
- **Polymorphism**: OOP allows polymorphism, enabling flexible and dynamic code through method overriding.

- **Abstraction**: OOP enables abstraction, hiding complex details and exposing only essential features