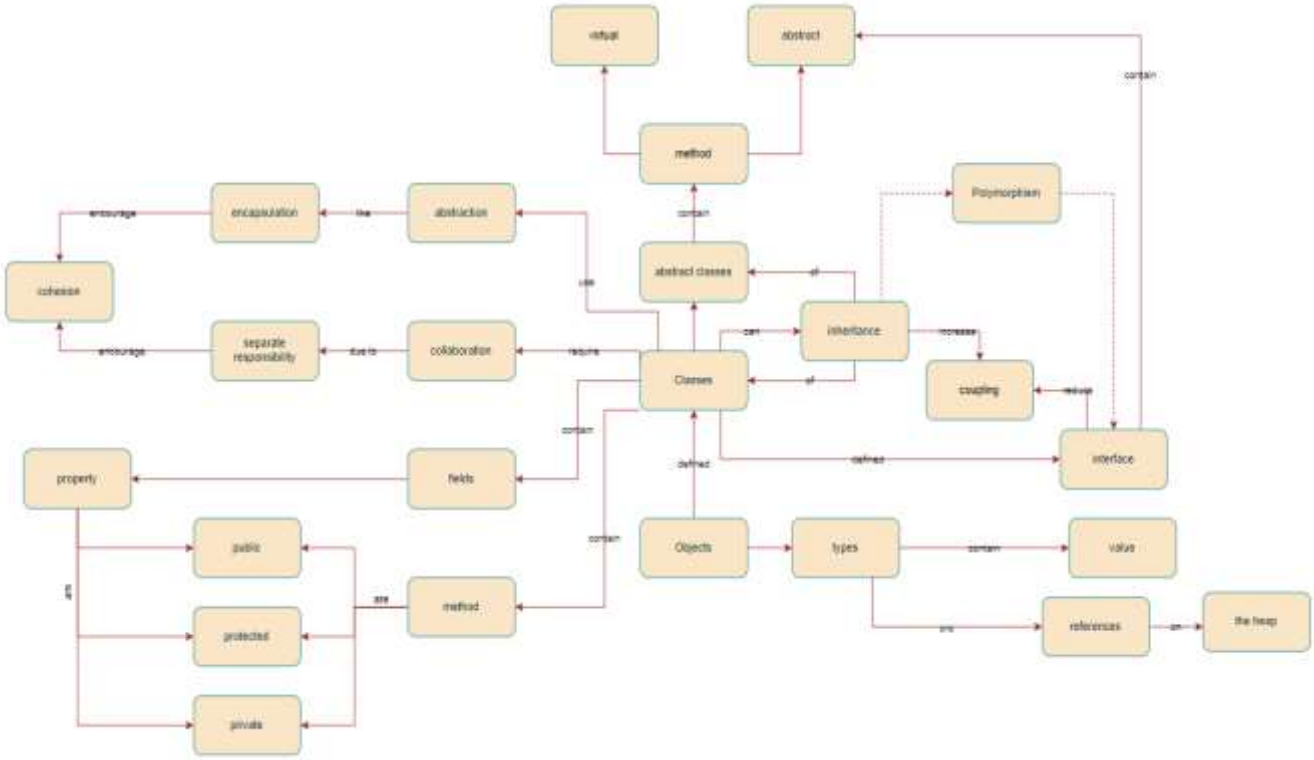
7.1P: Key Object Oriented Concepts

Full Name: Dao Khanh Nga Thi

Student ID: 104177393

The basic idea of "object-oriented programming" (OOP), a programming paradigm, is that objects can store both data and the code required to modify that data.. An object in OOP is a specific instance of a class that has state and behavior. The attributes and operations of objects are specified by classes. A unique object that may communicate with other objects through an interface or within the same program is called an instance of a class. Encapsulation, polymorphism, inheritance, and abstraction are all possible with OOP, giving it a strong and adaptable approach to programming.

The terms used, as well as the relationships between the ideas and the programming artifacts, are provided and illustrated in the concept map below.



In object-oriented programming (OOP), objects are the primary considerations while building a program and the final units of code that result from the procedure.

To deal with this situation, we must comprehend the meaning of the concepts of roles, responsibility, connection, coherence and collaborations of objects in OOP.

1. Roles

The concept of an object’s role in OOP is to define the properties and methods of an object. Properties are the data that an object holds while methods are the functions that an object can perform.

2. Responsibility

The concept of an object’s responsibility in OOP is to define what an object does and how it interacts with other objects. An object’s responsibility is defined by its methods and properties. An object's responsibility should be well-defined and focused, with a clear purpose or intention. This makes it easier to understand, use, and maintain.

3. Coupling

The concept of an object’s coupling in OOP is how much an object is dependent on other objects. Coupling is a measure of how closely connected two classes or modules are. High coupling means that changes in one module will affect many other modules, while low coupling means that changes in one module will have little effect on other modules. There are different forms of coupling such as object, data, control, ...

4. Cohesion

The concept of an object’s cohesion in OOP is how closely related the methods and properties of an object are. High cohesion means that the methods and properties of an object are closely related and work together to achieve a common goal. Low cohesion means that the methods and properties of an object are not closely related and do not work together to achieve a common goal.

5. Collaboration

The concept of an object’s collaboration in OOP is how objects work together to achieve a common goal. Objects can collaborate with each other by sending messages to each other. A message is a request for an object to perform one of its methods. When an object receives a message, it performs the requested method and sends a message back to the original object with the result.

Besides, OOP also includes the different rules:

1. Abstraction

Abstraction is the process of identifying essential characteristics of an object, ignoring those that are irrelevant. It is the act of representing only the necessary features of an object and hiding any unnecessary details. By focusing on what is essential, abstraction helps to simplify complex systems and enable better organization and management of code.

2. Encapsulation

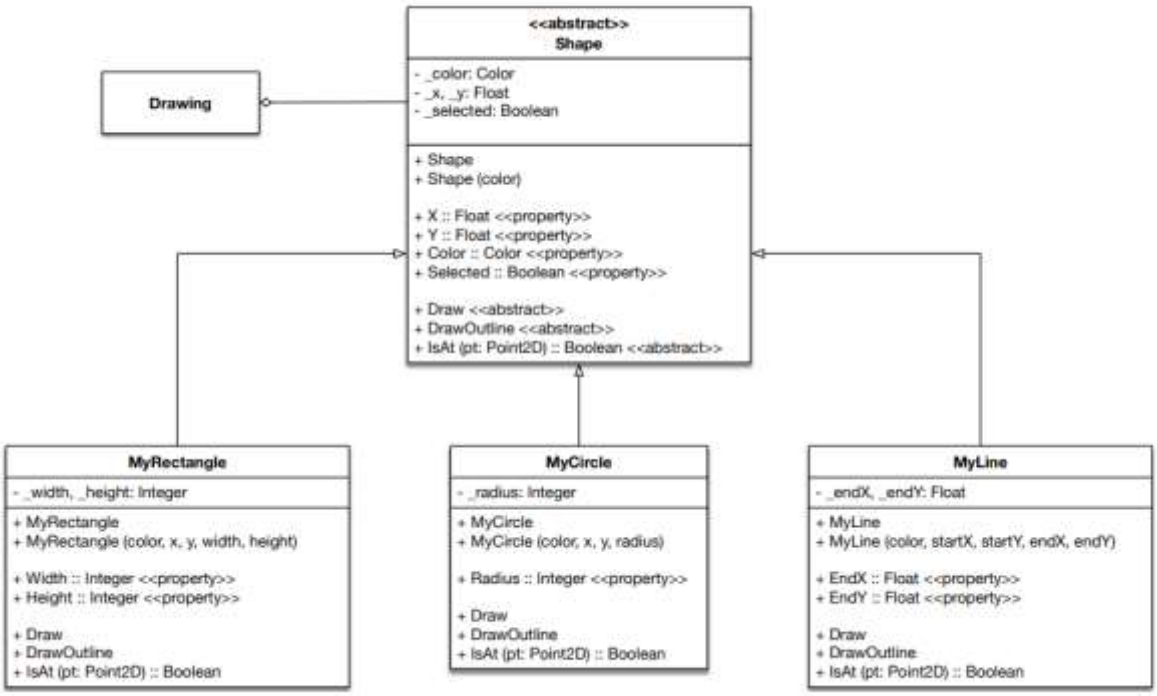
Encapsulation is the practice of hiding the internal workings of an object and only exposing a public interface. This is achieved by using access modifiers such as private, public, and protected to control access to an object's properties and methods. This helps to prevent unintended or unauthorized changes to an object's state and improves code maintainability by reducing dependencies.

3. Inheritance

Inheritance is a way of creating a new class from an existing class by inheriting its properties and methods. The new class can then be customized or extended to suit specific needs. Inheritance facilitates code reuse, simplifies implementation and maintenance, and enables polymorphism.

4. Polymorphism

Polymorphism is the ability of an object to take on many forms. This means that objects can be treated as if they are of a common type, even if they have different properties or behaviors. Polymorphism is achieved through inheritance, interfaces, and overloading. Polymorphism allows for more flexible and dynamic code that can adapt to changing requirements.



During the unit of Object-Oriented Programming, all four programming principles have been employed in this subject.

By restricting field access and utilizing properties, I leverage abstraction and encapsulation in the Shape Drawer program. Encapsulation prevents another object from accessing or altering the data. Avoiding duplication and cutting down on code writing time is made possible by inheriting MyRectangle, MyCircle, and MyLine child classes from the parent class Shape. At the same time, inheritance is also a way to produce polymorphism during programming. When defining an object's class, such as a circle or rectangle, we can also define it as a shape class to add versatility.