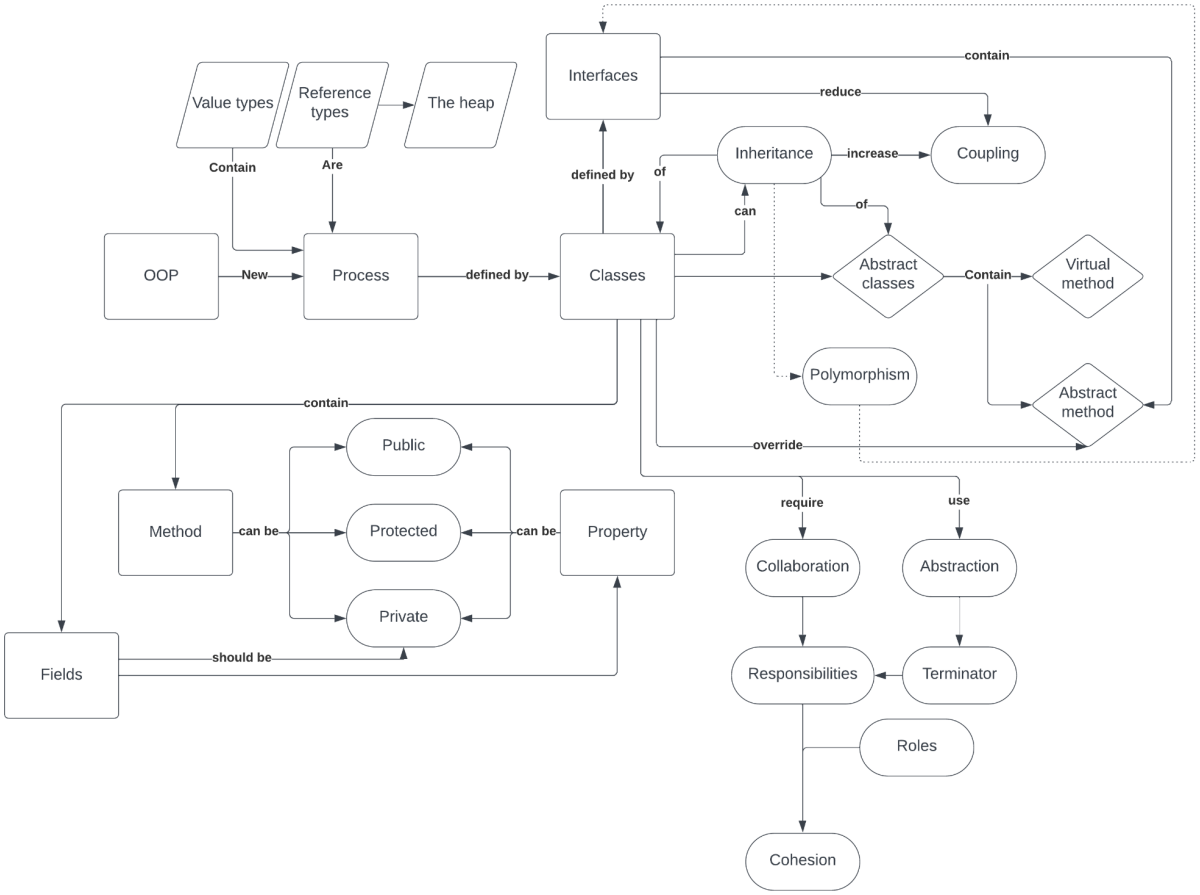
Object-Oriented Programming

An approach to programming known as object-oriented programming is created on the idea that objects can hold both data and code. The code is presented as procedures, whereas the data is shown as fields. Objects frequently have procedures connected to them that have access to and control over the object's data fields. Below are four main concepts of object-oriented programming:

- Encapsulation: Encapsulation enables the combination of data and methods within objects (Raut 2020). It is the fundamental idea behind OOP by ensuring that an object's internal state can only be accessed through specific interfaces. Therefore, encapsulation protects against direct manipulation and promotes data integrity. We can hide the complexity and give other objects a clear interface to interact with by encasing data.
- Inheritance: Inheritance makes it possible to create new classes, referred to as subclasses or derived classes, that take on traits and characteristics from pre-existing classes, referred to as base classes or superclasses (Savinov 2014). As common attributes and methods can be defined in a base class and inherited by numerous subclasses, inheritance makes it easier to reuse code. This encourages modularity and minimizes code duplication, resulting in software systems that are more effective and easy to maintain.
- Abstraction: Abstraction entails the development of streamlined models or representations of actual entities. It enables developers to concentrate on the fundamental characteristics and actions of objects while concealing unnecessary data. Large-scale software projects are simpler to design and maintain because abstraction facilitates the management of complexity and offers a high-level view of the system.
- Polymorphism: Polymorphism enables objects of various classes to be considered as instances of a single base class. It makes it possible to use abstract types, interfaces, and inheritance hierarchies to give various objects a unified interface (Savinov 2014). By allowing code to operate on objects of different types without needing to know their specific implementations, polymorphism improves flexibility and extensibility.

Developers can create software systems that are scalable, maintainable, and robust by comprehending and using these OOP principles. OOP offers a disciplined and explicit method for developing software, boosting collaboration, reuse, and code organization. OOP facilitates the development of modular and flexible software that can expand and adapt to suit the needs of many domains and industries thanks to its emphasis on encapsulation, inheritance, polymorphism and abstraction.

OOP Concept Map



Reference list

Raut, R 2020, *Research Paper on Object-Oriented Programming (OOP)*, IRJET, International Research Journal of Engineering and Technology (IRJET).

Savinov, A 2014, *Concept-Oriented Programming: References, Classes and Inheritance Revisited*, viewed 25 June 2023, <https://arxiv.org/pdf/1409.3947.pdf>.