Swinburne University of Technology

COS20007 Object Oriented Programming

Key Object Oriented Concepts

PDF generated at 22:59 on Sunday 15th October, 2023

Produced by OOP Submission Formatter

Name: Le Ba Tung
ID: 104175915

Object-Oriented Programming Key Concepts

Abstraction: refers to focusing on the essential features of an object or concept while ignoring the irrelevant details. In object-oriented programming, abstraction allows us to define classes that represent abstract concepts and only include the attributes and behaviors that are relevant to that concept. For example, in the Car class, we created, we abstracted the key features of a car like make, model, year, etc. while ignoring irrelevant details like the manufacturing process. Abstraction helps manage complexity by reducing real-world objects to simpler representative forms.

Encapsulation: Encapsulation refers to bundling data and methods within a class and controlling access to them via access modifiers like public and private. For example, in our BankAccount class, we encapsulated the account balance variable and set it as private so it could only be accessed via the deposit() and withdraw() methods. Encapsulation enables modularity by allowing objects to be self-contained and also improves security by preventing direct access to an object's data.

Inheritance: Inheritance enables new classes to be derived from existing classes. The derived or child class inherits the attributes and behaviors of the parent class, allowing code reuse. For example, our SavingsAccount class was inherited from the BankAccount class, so it automatically had the account balance variable and deposit/withdraw methods. Inheritance represents "is-a" relationships, like a SavingsAccount is a type of BankAccount. This enables polymorphism, where a derived class object can stand in place of a parent class object.

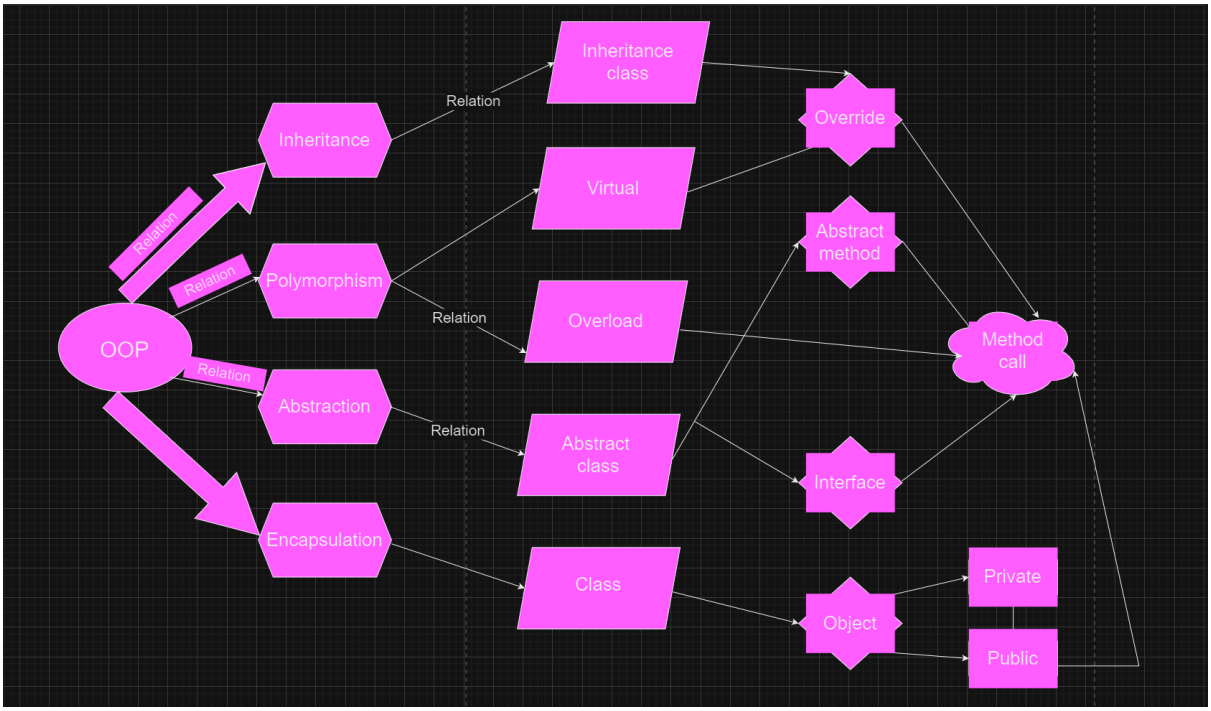
Polymorphism: Polymorphism means "many forms", and allows objects of different derived classes to be treated as objects of their parent class. Because the derived classes inherit the parent's interface, they can be used interchangeably. For example, our code worked on BankAccount objects, but could equally work with SavingsAccount and CheckingAccount objects because they inherit from BankAccount. This allows the same code to work on different kinds of objects, enabling greater flexibility and code reuse.

Interface: This is the ability of the class to know only what it should do, not how to accomplish it. It is the most common way to communicate with different objects. For

example, we just know about the Visa and Mastercard payment methods for the billing system, but we don't know how they work.

- Roles are the things that an object does and the attributes of the object in the context.
- Responsibilities: This term describes the roles that the object plays.
- Collaborations: concern how the objects interact with one another.
- Coupling: mention the interdependence of several classes.
- Cohesion: this is the interconnectedness of many classes.
- Class: describes the properties of the objects.
- Object: pertains to an instance of a class and is defined by the class.
- Approach: consult the use object function.
- Value type: the type contains its data, which is kept in the location of memory.
- Reference type: refers to the data's storage place.
- Abstract class: it contains abstract methods and acts as a base for additional classes.
- An abstract method is a bodyless method specified in an abstract class.
- Private: refers to the limited entry to the classroom members
- Overload: permits many methods to be used in the same class under different names.
- Override: refers to the implementation method that grants the derived class access to it. Found in the basic class definition.
- Virtual: The method can be overridden by the derived class.

Concept map



- OOP contains 4 concepts: inheritance, polymorphism, abstraction, encapsulation.
- The inheritance class can be in relations with the override class.
- Class contains the properties and behaviors of the objects.
- Objects can be private and public.
- The public object can be used within the method call.
- Interface is part of the abstract which the class can implement.
- Abstract class contain abstract methods.