Principles of Object-Oriented Programming

Name: Vishnu Kishankumar Hathiwala

Student ID: 104541189

Abstraction

Abstraction is one of the key principles in object-oriented programming as it enables the functionality to only show the relevant and essential information and keep the confidential or irrelevant information hidden. It also allows the user to avoid facing all the complexity in the internal implementations of the code and achieve the result easily. Abstraction also allows individual classes to have its own custom implementation by overriding. Abstraction was used in the task 4.1P where Shape class was labelled as an abstract class along with class Draw.

Encapsulation

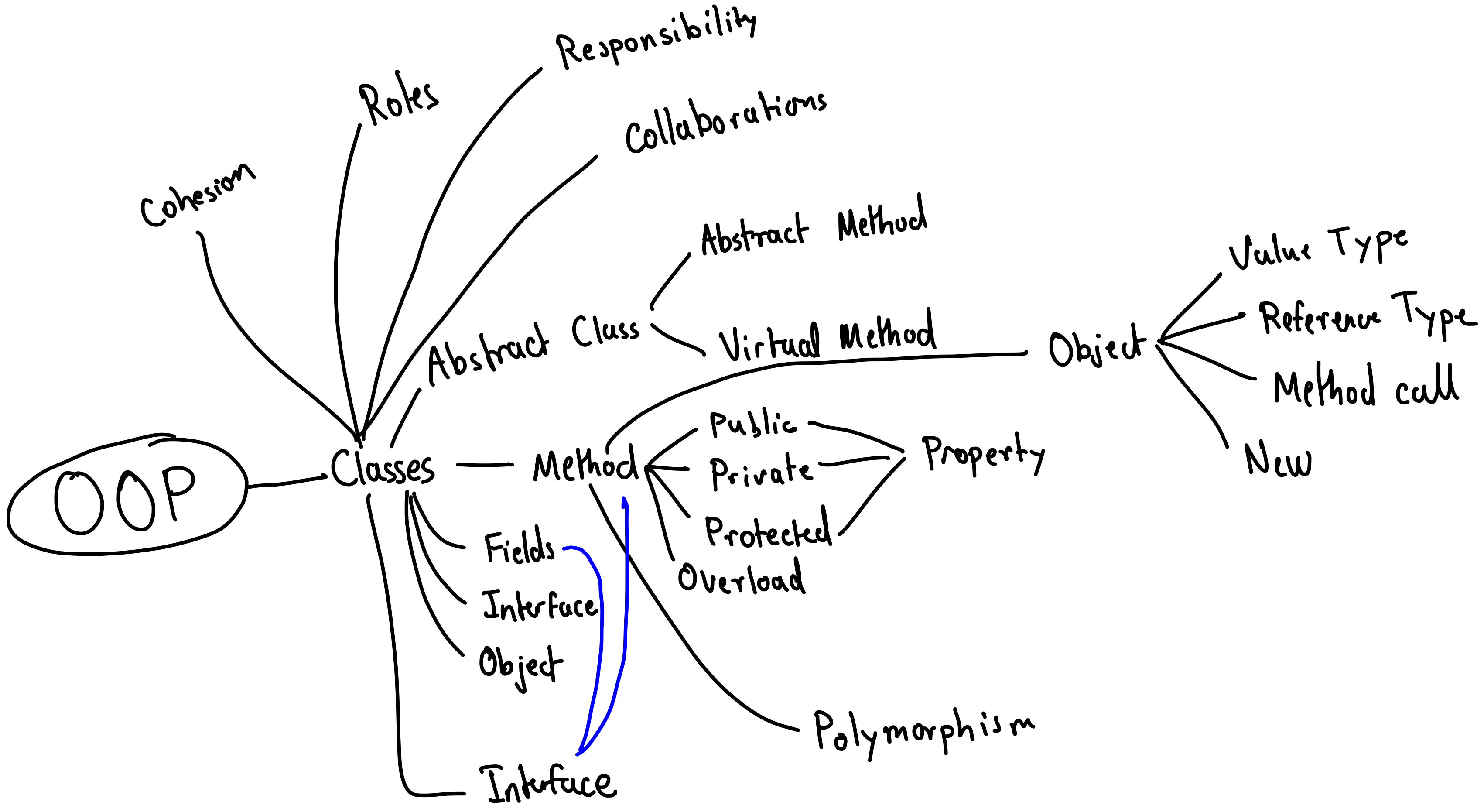
Encapsulation is one of the key principles in object-oriented programming as it gives us the functionality to prevent direct change in the internal data. Supposing we define a private field, that field is only accessible in that class and cannot be accessed outside of the class however properties or methods can be used to change or modify the data in the fields. This concept of saving data from direct modification yet allowing us to change it in a certain way, is encapsulation. The concept of encapsulation is used in almost every task however was introduced in the task 3.3P. In the Counter Class, private fields _count and _name were created along with methods Counter(), Increment() and Reset() which can use those variables. Since the fields are private, their values cannot be changed directly however can be modified with those public method. This is an example of encapsulation.

Polymorphism

Polymorphism is very much like inheritance in a way how classes can be divided into parent and child classes and children class can use all the fields and attributes of parent class. However, in polymorphism, the children class will implement all the methods from parent class with its own unique implementation. Again, polymorphism is clearly seen in task 4.1P where Shape's Draw() and DrawOutline() method is implemented in MyCircle and MyRectangle with their own unique modifications. When we want to draw a rectangle, Draw() and DrawOutline methods of Rectangle will be called and same for circle. This is an example of polymorphism.

Inheritance

Inheritance can be treated as a relationship between a parent and child. Just like how a child inherit all their traits from their parents with some additional qualities of their own, inheritance in programming can do the same! The classes can be split into parent and children classes. Inheritance can vividly be seen in the task 4.1P where Shape is the parent abstract class and MyCircle, MyRectangle are children classes. The abstract shape class provides all the properties like X,Y, Width and Height and MyCircle and MyRectangle uses those properties by overriding them and implementing their own unique features. This example demonstrates Inheritance.


OOP
Classes
Cohesion
Roles
Responsibility
Collaborations
Abstract Class
Abstract Method
Virtual Method
Object
Value Type
Reference Type
Method call
New
Method
Public
Private
Protected
Overload
Property
Fields
Interface
Object
Interface
Polymorphism