Concepts of OOP

The main concepts of OOP consists of Inheritance, Polymorphism, Encapsulation, Abstraction.

In inheritance, classes can inherit characteristics from other classes such as class methods and class properties, which are known as the parent classes. There are 4 type of inheritance, namely single inheritance, multilevel inheritance, hierarchical inheritance and multiple inheritance. Single inheritance is where a class inherits properties from a single class. Multilevel inheritance is where a class inherits from a parent class which inherits properties from another parent class, known as a super class. Hierarchical inheritance is where two or more classes inherit properties from the same parent class. Multiple inheritance is where one class inherit properties from multiple parent class. Inheritance is important when you require objects or functions that have similar working properties or to use existing functionalities provided in frameworks/packages. In my code for example, I inherited the BaseCommand class from the Django framework for my command handler, allowing me to call the commands from the terminal without having to code the logic myself. The inheritance handles all the backend logic for me.

In Polymorphism, the most common use is where child classes can override parent class methods by defining the same method names. For example, to change the account identifier for which the jwt access token can be obtained, we can overwrite the rest framework JWT serialiser and view methods. This will let us customise classes from frameworks/packages specially for our own project/work.

Encapsulation serves security purpose, where class variables cannot be accessed unless you know the class method to call in order to access them. This is known as the setter method. For example, in a class view, unless you know the setter method's endpoint, you are unable to edit or check this value, hence it would only be accessible by people you give the endpoint to, such as clients.

Abstraction is a process of handling complexity by hiding unnecessary information from the user. Abstract classes can only used for inheriting its functionalities. An example of abstract classes is the classes we can use in the Django Framework, such as view classes and pagination classes.