COS20007 – Object Oriented Programming

Key Object-Oriented Concepts

Introduction:

OOP (Object-oriented programming) provides numerous benefits that changed the way of programming for many developers. For example, OOP allows developers to reuse parts of their code from one part and apply it to another area in their program and maximising productivity by decreasing the amount of code that needs to be written, since some of them can be reused. Additionally, developers can debug their code at a much faster rate, as OOP will show the location where the error was outputted.

Key Concepts of Object-oriented programming:

1. **Inheritance**

- Inheritance is the method where a child class that shares the characteristics of the parent class. This allows the program to reuse functions and properties from the parent class in the child class. The parent class can have multiple child classes that shares their parent's data and properties, saving times for developers from writing repetitive code and better code organisation.
- **Example:** This concept was used in the Swin-Adventure Maze Game project where the IdentifiedObject class is used as the base for containing the IDs in the game, which was utilised by the GameObject class to identify whether the in-game object is a player or an item by IDs.

2. **Polymorphism**

- Polymorphism is the method of overriding the initial instructions for a function to repurpose them with new instructions and code. This allows the program to create new variants of the same function and return unique outputs for the same function, depending on which variant of that function is being called. The program can also be extended and maintained without changing any code that was already written.
- **Example:** This concept was used in the Shape Drawing project where Rectangle, Circle, and Line classes share the same Draw function as the Shape class since the Draw function originated from that class. However, the other child classes were able to use that function due to it being overridden. The Draw function can now be called through any child class and draw a shape according to the new instructions.

3. **Abstraction**

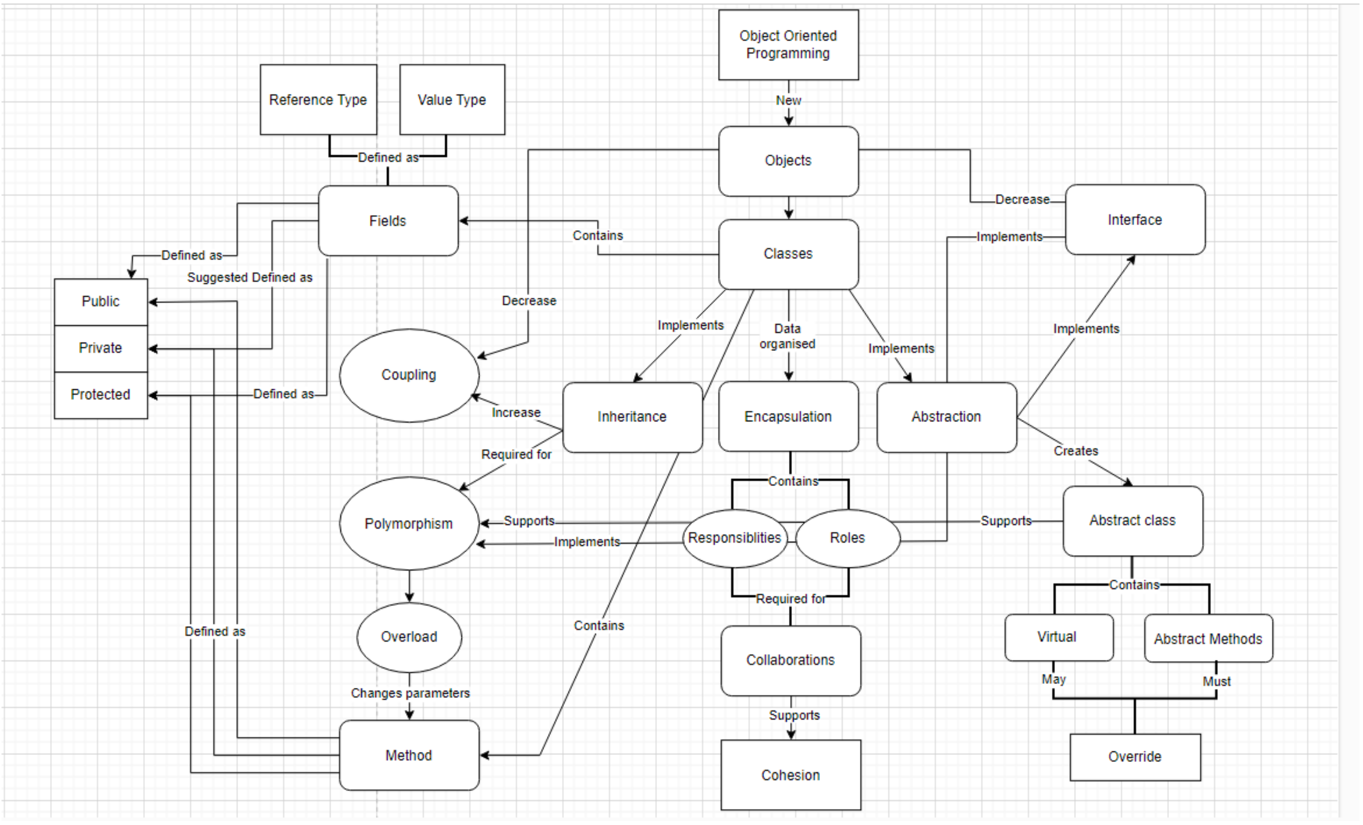
- Abstraction is the method that involves abstract classes and interface. Any child class that originates from an abstract parent class, is forced to override functions that they want to borrow from the parent class or there would be syntax error. An abstract class acts like the baseline with attributes that its child classes must follow. On the other hand, an interface is a framework that requires a class to use a pre-written set of functions. Abstraction aims to simplify the code structure of the program while allowing the code to be reused and reduce duplicate code.

- **Example:** This concept was used in Swin-Adventure Maze Game project where the GameObject class was initialised as an abstract class while being a child class of IdentifiedObject class. GameObject then have a few more child classes, most notably Player class where it overrides the FullDescription property from GameObject. Player class also uses the interface IHaveInventory that contains the Locate function.

4. Encapsulation

- Encapsulation is a method of limiting access to certain parts of code or/and data within the program. When creating a class, encapsulation happens due to the creation of functions, properties, and data that are stored within that class, as well as setting security measures by adjusting each component's access level with only a few areas that can be accessed globally. The purpose of encapsulation is to maximise security to prevent unauthorised access to the core functions and data within the program and make it the program easier to maintain and read.
- **Example:** This concept was used in Shape Drawing projects where classes were created and stored data with some areas restricted to local access only. For example, the Shape class contains private variables for storing data such as "_color", "_x, _y", and "selected". These variables can only be modified through accessing the public properties of the Shape class like "Color", "X, Y", and "Selected".

Concept Maps


Object Oriented Programming
New
Objects
Reference Type
Value Type
Defined as
Fields
Contains
Classes
Decrease
Interface
Implements
Defined as
Suggested Defined as
Public
Private
Protected
Defined as
Decrease
Coupling
Increase
Implements
Data organised
Implements
Implements
Inheritance
Encapsulation
Abstraction
Creates
Required for
Polymorphism
Supports
Implements
Contains
Responsiblities
Roles
Supports
Abstract class
Required for
Contains
Defined as
Overload
Contains
Collaborations
Virtual
Abstract Methods
May
Must
Changes parameters
Method
Supports
Cohesion
Override