Swinburne University of Technology

COS20007 Object Oriented Programming

6.2P - Key Object Oriented Concepts

PDF generated at 15:54 on Monday 1st May, 2023

Key Object Orientated Concepts

The four key principles of object orientated programming is Abstraction, Encapsulation, Inheritance and Polymorphism

Encapsulation:

Encapsulation is the concept of defining what parts of data and functionality is accessible. Encapsulation allows for a specific objects functionalities and information to be private and only used within the class through access modifiers. Other details of the object that is allowed to be modified and accessed is left public. This concept is crucial since it prevents unintentional change or interference to occur to the object and its functionalities which could ruin the intended purpose and workings of the object.

When creating a class, we assign methods and fields with public or private. These access modifiers restrict or allow access to this information from being used outside the class.

Abstraction:

Abstraction is the process of mapping the essential characteristics of an object where the specific implementations of these features are ignored in order to create a digital representation of how the object behaves and its core features. Abstraction indicates the roles the objects need to have and how it may interact and collaborate with other objects in order to behave as it is intended.

Abstraction can be used through UML diagrams and sequence diagrams when designing code. it gives the outline of the features and collaborations classes may have.

Inheritance:

inheritance is the idea of creating a child class from a current parent class. Child classes or derived classes have all the information from the base or parent class but can expand on the parent classes properties and behaviours. Inheritance allows for better cohesion as code duplication is reduced and also reusability of existing code which in turn improved code management. Inheritance can be implemented through abstract classes.

An example of inheritance from the iteration tasks as it uses abstract classes where other classes can inherit from that abstract class reducing the need to rewrite code.

Polymorphism:

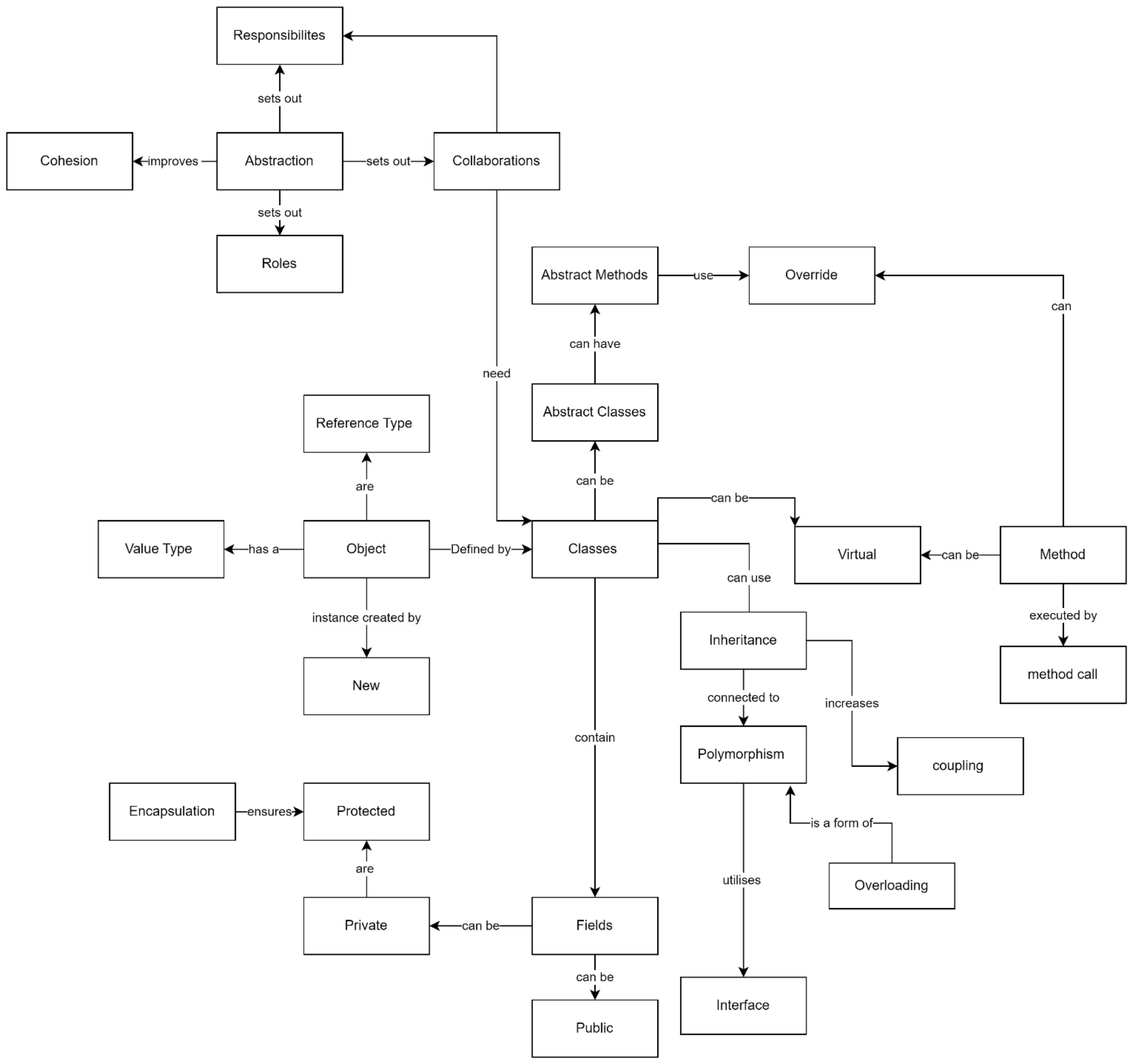
Polymorphism is referring to an object where it can take in many forms. Polymorphism allows for objects of different object to be implemented as if they were the of the same type of class.

Through interfaces and abstract classes are able to override methods from the parent class so that it can behave differently and be implemented as if the where the same object type.

Polymorphism is implemented through families of related objects; this is shown in the shapes tasks. Where we overload methods so that we can call it in different ways and creating subtypes to allow us to use the shape objects child classes such as rect, circ and line.

Shared interfaces also is used to define a functionality so that it can be implemented by many classes and can be treated as if they were the same type.

OOP Conecept Map


Responsibilites
sets out
Cohesion
improves
Abstraction
sets out
Collaborations
sets out
Roles
Abstract Methods
use
Override
can
can have
need
Abstract Classes
Reference Type
can be
are
can be
Value Type
has a
Object
Defined by
Classes
Virtual
can be
Method
can use
instance created by
executed by
Inheritance
method call
New
connected to
increases
contain
Polymorphism
coupling
Encapsulation
ensures
Protected
is a form of
are
utilises
Overloading
Private
can be
Fields
can be
Interface
Public