COS20007

7.1P Object Oriented Principles

Lab: LA1-11, Friday, 10:30 AM, ATC621

Tutor: Agus Hartoyo

Contents

Concepts and Principles

Inheritance

Inheritance is where one class can inherit a parent class to essentially borrow its method and variables which can be extended upon in the child class.

Abstraction

When using inheritance, the parent class can be set to an abstract class. This way, when each method is inherited, they can be manipulated to suit the exact need of the child class.

Polymorphism

You may be wondering, “why would I need to use abstraction if I can just declare the method in the child class themselves?” well it is a valid question, and it is all to do with poly morphism. Essentially, polymorphism is the idea that an object can be used as varying types. When we use abstraction, this allow the child object to be treated a parent object.

For example, in our shape drawing program, the abstract parent shape class is inherited by the child classes: rectangle, line and circle. Abstraction is demonstrated whereby there is an abstract draw method, which is overridden within each child class. The reason it is overwritten is because a different draw method is called for each different shape, otherwise each child class would draw the same shape. By doing so, each child class can be identified a shape and its object specific shape. This is useful because say we have a list of multiple different shapes, the can all be called under one foreach loop in the game loop because they all or identified as a shape object.

Encapsulation

Encapsulation is all to do with the level of accessibility that a piece of data has. This tends to intertwine with abstraction as it is essentially implementing the desired level of abstraction. In C#, modifiers are used such as public, private, protected, internal and protected internal to control these degrees of encapsulation.

Roles

Often to explain and understand code functions, analogies can be used that apply to code to a real-life scenario. This helps as it is referencing something that everyone is familiar with.

Collaborations

When we work on a project, very often in the real world we will work collaboratively. Task will be delegated through iteration and product backlogs.

Cohesion

Cohesion aims to have each class have an exact purpose, and each method and variable declared within the class is very focused at contributing this purpose. Every function should only be written at maximum once so when cohesion works efficiently, the entire code works efficiently and is easier to understand.

Responsibilities

Each person involved in a project is responsible for their part. What is delegated to them is for them to complete. These responsibilities are determined by the expected time taken and the prioritization in relation to other tasks.

Coupling

Coupling is all to do with the relationship between a child class and its parent class. In all situations, we should avoid what is known as tight coupling. This is where a ripple effect is created by modifying one piece of code means another piece of code needs to be modified. Instead, loosely coupling means that one piece of code can be modified without anything else needing to be changed. This is ideal as it is more efficient for both the developer and memory usage itself.

Programming Artefacts

Class

A class serves as a new type. It contains its own variables and method that can be initialized, altered and called within the object reference.

Object

An object one implementation of a class. This object contains its own values for each variable that is declared within the class.

Interface

Interfaces function similar to abstract classes whereby they are inherited, but instead implemented and are used as a template for other classes. There is no actual code and modifiers on the methods and variables. The interface should then contain a corresponding class which implements all of the features. The reason this should be done

Method

A method has its own type and perform specific tasks. They can be called from the object reference. If the type is non-void, it is known as a function as it always returns some kind of value. However, if it is void, it is a method as it runs lines of code but returns void.

Fields

Fields are often known as properties. They allow a private variable to be accessed from other classes through get and set. If there is no set command in place, the variable is constant, meaning it cannot be changed hence it is a read-only property.

Podcast Link

https://youtu.be/Td-F6M1UTrg