COS2007 - Object Oriented Programming Task 7.1
Son Nguyen - 1032324103

Object Oriented Programming - Letter to a friend

Dear James,

I am writing to you to talk about my wonderful learning experience and wanted to share with you about a fascinating topic that I have discovered during my study. The topic I'm referring to would be Object Oriented Programming (OOP). It's designed which focuses on data rather than action and logic , using literal components to build up into a larger system. It's great I tell you.

You must be asking yourself my dear friends , how does it all start? Well in object-oriented, we first have to define the building block of it , in this case it's object and class. Class is defined as the core of modern OOP languages as it allows for decomposition of a problem into several entities called objects. It's like a blueprint of an object that includes variables for storing data and method to perform an operation on the data. Picture a car, a car has many things like model/plate/brand/colour - thats a class.

An object is like a representation of a real life object - let's say for a simple example, the car I mentioned earlier. An object has fields which are like tooltips and methods which allow it to do things. An object being the car itself which can behave by itself , but can be changed. Several objects can be created in one program like how one brand can create many cars which allows for one company to create many different set of car like how in OOP especially C# - a program can consist of multiple objects that interact dynamically.

Much like a car , for it to be optimal ; you need certain parts to work together cohesively ; I don't know much about real life car mechanisms but in OOP , it's about the classifications , roles , responsibilities and collaborations of objects to be able to make a good design software.

Man , let me tell you about the cohesion of them together. A role is a set of responsibilities and responsibility much like IRL is a task we should do , but in this case its object , which in turn means that an object has many responsibilities such as knowing its fields and has its own method.

Every part working together in order to complete a common goal , much like you & I , and much like in real life , this would be called collaboration. Much like coding, if we don't communicate and devise tasks equally , nothing can be achieved. Like us , they collaborate by sending each other a message.

Remember how I mentioned field earlier , well they are variables of type that are declared in a class which can be used to store data about the class , like our student ID get stored in the system at Swinburne. Since our IDs are numbers , they must be given an assigned operator when they are declared and these fields are generally only used for variables that have protected or private accessibility.

Now that we have all the above , we must ask the machine to perform a task for us using these variables which is what I like to call a method. Methods are blocks of code that contain statements and usually perform a task together. Like how we communicate but this is us communicating to our computer to perform a task. Every program consists of at least one method called Main. Input value can be given from arguments or parameters.

Codes are like us , they have certain requirements to be sensible or non hostile towards each other. I'll start off with cohesion - degree in which elements of modules belong together ; a high cohesion mean seperate class for all methods to execute a specific job,

which is easier for maintenance whilst low cohesion mean only one individual responsible to execute lot of task which can be hard since it's just one man as well as maintenance. Coupling is computer words for relationship between objects (well individuals). Tight coupling means classes and objects are codependent on each other which is not good since it creates no flexibility like your current dating situation (i joke) mean while loose coupling means they require fewer dependencies since they are not dependent on each other.

Last but not least is interface. The interface is a contract that defines a set of methods , properties , events and indexers that a class or a structure must implement. In Layman's term , they are a blueprint instructing what a class must do to achieve abstraction.

All of the above is the foundation of your typical OOP set of code but to master the craft , we require the 4 elements like The Last Airbender (just not natural) - We will require to understand the four principles of Abstraction , Encapsulation , Inheritance and Polymorphsm. Don't worry i'll explain to you what they are , I will try to make it interesting so that it would not bore you.

We start with Abstraction , which is like a janitor - they clean up the unwanted details to show you shiny clean floors but in OOP case , they would show you the essential information. They help to reduce the complexity of the design and implementation process of software and solve issues at the design level and allow focus on what information objects must contain. Basically like a janitor , they clean up the mess behind the scene so that the floor is fresh in the morning.

I recently had the pleasure to practise these in one of my Lab tasks , in 4.1P task in Line Class. The SplashKit.Drawline method from the SplashKit library is called to draw the line. It is essential to provide DrawLine option to draw the line.

Encapsulation means wrapping code and data together into a single unit, much like a capsule which is the mixture with several medications. We can use the getter and setter method to set and get the data in it. It also hides the value and state of the data object that prevent unauthorised parties direct access to them.Encapsulation is a method to protect information from the outside.

Inheritance is one of the most important concepts in object-oriented programming that inherits the parent class from the child. This also provides an opportunity to re-use the code functionality. It is used to achieve polymorphism which I will also tell you about but the main reason to use inheritance is still to reuse the code. Inheritance mean an "is-a" relationship between classes

Last but not least in the principles - polymorphism , basically the features that enable to perform one thing multiple ways - a jack of all trades if you will. Polymorphism occurs when we have many classes that are related to each other by inheritance. Every child class inherits methods and properties from the base class.Polymorphism uses this method to do different tasks. In this way , a single action can be performed in many ways.

I tell you James - I have learnt so much about Object Oriented Programming, much like us they are created the same way , structured the same way. Anyway thanks for listening rambling on about what I have learnt this year - hope we can catch up some other times so I can share to you more about my interest. With love - Son

Reference

C++ Home Sept 17th 2020, Object-Oriented Programming, CPP, viewed 21 January 2024, < http://www.cpp-home.com/archives/206.html>.
Microsoft 2017, C# Programming Guide, Microsoft, viewed 22 January 2024, https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/.