ELI5 OOP BLOG

Concepts and Principles

Abstraction

Abstraction is used to handle complexity by hiding unnecessary details from the user. It enables the user to implement more complex logic without understanding or even thinking about all the hidden complexity. It is accomplished using an abstract class or interface (TutorialsPoint 2019). In 5.3D the 'my.Drawing.Draw()' line draws the drawing, hiding all the details of the drawing process.

Inheritance

Inheritance allows you to define a child class that reuses, extends, or modifies the behaviour of a parent class. The class whose members are inherited is called the base class. The class that inherits the members of the base class is the derived class (Microsoft 2019). In 5.3D, the 'Rectangle' class was one of the classes that inherited from the 'Shape' class. With the use of inheritance, the 'Rectangle' class looked and felt like a Shape. It could do everything a Shape could and its override functions was used to change the 'Draw' and 'Draw Outline' methods so that it could draw rectangles.

Roles

The role is used to describe an object classification, whether it should be implemented as a class, abstract class or interface. It is essentially a set of responsibilities that can be used interchangeably (SwinBrain 2009). The class 'Drawing' in 5.3D is used to create an object, while the 'Shape' is an abstract class that provides partial implementation, requiring derived class such as 'Rectangle' to provide the missing functionality such 'Draw'.

Collaborations

Collaborations are requests from one object to another. One object calls on or collaborates with, another because it needs something. The two objects work together to fulfil larger responsibilities (SwinBrain 2009). In 5.3D, we have drawing and the drawing holds shapes. There are three main types of collaboration ones are: "uses", "has-a", and "is-a-kind of", being Dependency, Association, and Inheritance respectively.

Cohesion

Cohesion is used to indicate the strength of the relationship between the methods and data of a class and some concept served by that class. High cohesion is preferred because it is much easier to maintain and reusable (Kanjilal 2015). In 6.1C, the 'Player' class has no stray methods, a method does one thing. The 'Locate' method returns a GameObject or null, it is not expected to return the 'Inventory' or 'Location.

Encapsulation

Encapsulation is when you make some parts of your program inaccessible to access other parts of the program. It is the ability to hide data and behaviour that are not relevant for the users. Encapsulation allows a group of properties, methods and, other members to be considered an object (TutorialsPoint 2019). In 5.3D, the 'Drawing' class is where the shapes get drawn to, but the 'Drawing' class does not know how to draw the shapes. It is the 'Shape' class that draws itself.

Polymorphism

Polymorphism allows different classes to be treated the same as its base class. It occurs when a parent class reference is used to refer to a child class object (TutorialsPoint 2019). In 5.3D, we wanted each shape to able to draw itself differently using the same inputs (color, x, y). We got the 'Rectangle' class to override the 'Draw' and 'DrawOutline' methods in the 'Shape' class.

Responsibilities

Responsibilities refers to the object been able to do something or know something (SwinBrain 2009). In 6.1C, a Location object is responsible for knowing its description.

Coupling

Coupling refers to how related or dependent two classes are towards each other. For low coupling class, changing something major in one class should not affect the other. High coupling would make it difficult to change and maintain since everything is dependent on another. Good software should be low coupling (Kanjilal 2015). In 6.1C, the 'Player' class could continue to work without 'Inventory' class existing, vice versa.

Programming Artefacts

Class

A class is like a blueprint for a specific object. It defines properties, fields, events, methods, etc. it also defines the kinds of data and the functionality their objects will have. A class will not occupy any memory space but must create a variable for the class, that is known as an object (TutorialsTeacher 2019). In 1.1P we created a class called 'Message' which only had one method, Print. When the method Print with parameters was called, you passed the parameters to the Print method and it would print.

Object

Objects are the run-time entities of an object-oriented program. They can represent a person, a place or any items the program must handle. When an object is created the new operator, memory is allocated for the class in the heap, the object is called an instance and its starting address will be stored in the object in the stack. If an object is created without the new operator, the memory will not be allocated in the heap and the object in the stack contains the value of null (GeeksforGeeks 2019). In 2.1P, we had the main class and the 'Shape' class. Whenever the new operator was executed in the main, a new object was created.

Interfaces

An interface defines what an object must do (methods) and cannot define attributes in the way that a class definition does. They are used to build loosely-coupled applications, an application with components that are not tightly related to each other. Therefore, changing in one of its components is easier and has less or no impact on other components (TutorialsPoint 2019). In 5.1P, we created an interface called 'IHaveInventory' which was used to allow the 'Player' and 'Bag' classes to be treated the same way, as objects that contain other items.

Method

A method is a block of code that contains a series of arguments. A program causes the statements to be executed by calling the method and specifying any required method arguments (TutorialsPoint 2019). In 5.3D, we had a method called 'Draw' in the 'Drawing' class. When executed, the 'Draw' method would clear the screen and draw each shape in the list.

Fields

The field is a class level that holds a value. They generally have a private access modifier and are used with a property (Techopedia 2019). In 5.3D, the 'Shape' class contained fields such as colour, x-coordinate, y-coordinate, width and height. They either got their value from either the parameters passed through the constructor or from the default constructor when no parameters were passed.

References

- GeeksforGeeks 2019, C# | Object Class, viewed 26 April, 2019, <https://www.geeksforgeeks.org/c-sharp-object-class/>.
- Kanjilal, J 2015, Design for change: Coupling and cohesion in object oriented systems, viewed 26 April, 2019, <https://www.infoworld.com/article/2949579/design-for-change-coupling-and-cohesion-in-object-oriented-systems.html>.
- Microsoft 2019, Inheritance in C#, viewed 26 April, 2019, <https://docs.microsoft.com/en-us/dotnet/csharp/tutorials/inheritance>.
- SwinBrain 2009, Object Oriented Programming - Collaborations and Relationships, viewed 26 April, 2019, <https://swinbrain.ict.swin.edu.au/wiki/Object_Oriented_Programming_-_Collaborations_and_Relationships>.
- SwinBrain 2009, Object Oriented Programming - Responsibilities, viewed 26 April, 2019, <https://swinbrain.ict.swin.edu.au/wiki/Object_Oriented_Programming_-_Responsibilities>.
- SwinBrain 2009, Object Oriented Programming - Roles, viewed 26 April, 2019, <https://swinbrain.ict.swin.edu.au/wiki/Object_Oriented_Programming_-_Roles>.
- Techopedia 2019, What is a Field? - Definition from Techopedia, viewed 26 April, 2019, <https://www.techopedia.com/definition/1203/field-c>.
- TutorialsPoint 2019, C# Encapsulation, viewed 26 April, 2019, <https://www.tutorialspoint.com/csharp/csharp_encapsulation.htm>.
- TutorialsPoint 2019, C# Interfaces, viewed 26 April, 2019, <https://www.tutorialspoint.com/csharp/csharp_interfaces.htm>.
- TutorialsPoint 2019, C# Methods, viewed 26 April, 2019, <https://www.tutorialspoint.com/csharp/csharp_methods.htm>.
- TutorialsPoint 2019, C# Polymorphism, viewed 26 April, 2019, <https://www.tutorialspoint.com/csharp/csharp_polymorphism.htm>.
- TutorialsPoint 2019, What is abstraction in C#, viewed 26 April, 2019, <https://www.tutorialspoint.com/What-is-abstraction-in-Chash>.
- TutorialsTeacher 2019, C# Class, viewed 26 April, 2019, <https://www.tutorialsteacher.com/csharp/csharp-class>.