Mark Biswell

The Characteristics of an Object Oriented Paradigm

Encapsulation

Encapsulation is a core concept in object oriented programming. It refers to the idea of placing different methods and other data together in one place, such as a class. Encapsulation can also be used to hide what state an object is in from elsewhere in the program, this is refered to as 'information hiding'.

Polymorphism

Polymorphism in object oriented programming refers to the ability of a program to process objects in different ways. How the object is handled can depend on what data it holds. An example of this could be a class called 'shape', this can be processed by the program to create circles, squares or triangles, but they are all defined as shapes.

```
gumballMachine(int qty) { //a contstrutor that sets the ammount of gumballs in the machine
  std::cout << "Welcome to the Gumball Machine!\n"; //Prints message
  std::cout << "There Are 3 Gumballs in the Machine\n"; //Prints message
  setGumballsLeft(qty); //sets the ammount of gumballs left
  setState(NoCoinInserted); //sets the initial state to NoCoinInserted
}
```

Constructors & Destructors

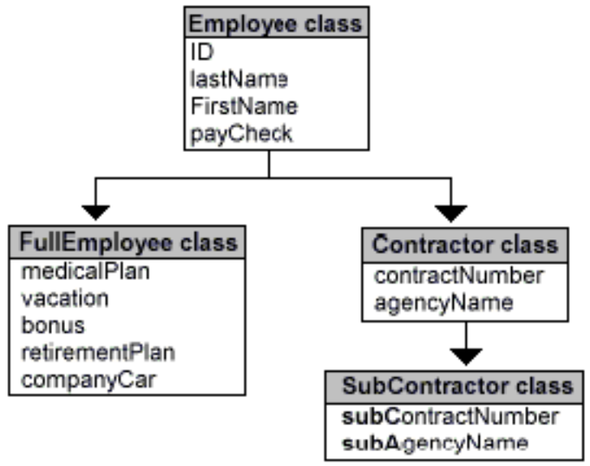
Constructors are a part of a class that is run automatically when an object is created using that class. They define basic functions and properties that the object should be created with. Destructors are the opposite, they are part of a class that is run when the object is deleted. They are used to clean up the application to make sure that no processes are left running that are no longer needed, they can also be used to make sure no data is lost by automatically saving data when the object is deleted.

Abstract & Concrete Classes

Abstract classes are designed to create the blue-prints that concrete classes can use. They contain key properties and methods that the other classes need. Concrete refers to a class which properties and methods are fixed, they cannot be changed and form the base for other classes to draw from

Base & Derived Classes

A base class referes to a type of class that others can be derived from, they inherit the property and methods of the base class by essentially reusing the code in the base class. A derived class is a class that inherits the properties and functions of a base class, Base classes can also be refered to as a superclass.



Objects, Sub-Objects

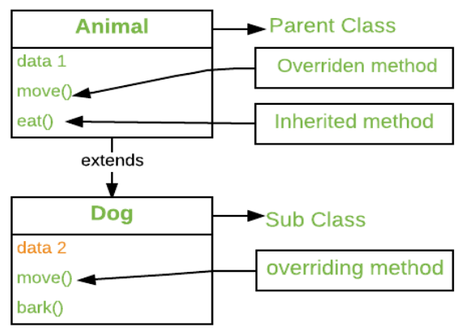
Objects can be defined as a specific instance of a class. All objects have properties and most will have methods, Sub-objects can be considered to be objects that are created using properties and/or methods derrived from a different object. Container refers to a data structure or a class that are instances of multiple other objects or classes, storing the objects in an organized way according to the containers own set of rules.

Object Oriented Interfaces

These are interfaces that are created by following the object oriented paradigm. The interface is structured around multiple interactive objects that combine to for a user interface. Most modern operating systems use some form of object oriented interfaces to create their user interface

Method Redefinition, Overriding & Overloading

Method redefinition or overriding refers to the process of a sub class taking a method that it has inherited and changing it's process in some way. Method Overloading is the process of having two or more methods that have the same name in the same class, the only difference being that they have different parameters.



Generics & Templates

Generics refers to the process of creating source code templates. A template in OOP is essentially a blueprint that can be drawn from to create generic functions and classes that can then be modified further to solve a specific problem,

Career Ride. 2018. OOP Charecteristics. [ONLINE] Available at: https://www.careerride.com/oops-characteristics.aspx. [Accessed 25 November 2018].

Mind Meister. 2018. Characteristics of OOP. [ONLINE] Available at: https://www.mindmeister.com/1103601222/characteristics-of-object-oriented-programming. [Accessed 25 November 2018].