The Characteristics of an Object Orientated Paradigm

Object oriented programming

Object-oriented programming (OOP)refers to a paradigm of programming organized around objects rather than "actions" and data rather than logic.Object-oriented programming is based on the three concepts encapsulation, inheritance, and polymorphism.

Encapsulation

A fundemental concept in OOP refers to the wrapping up of data (Variables and functions) under a single unit. Thus creating self contained modules that bind proccessing functions to data. It ensures code modularity, which keeps routines seperate and less prone to conflict with each other. This concept is often used to hide the internal representation, or state, of an object from the outside (Private, Public, Protected).

Polymorphism

Refers to an objects ability to take multiple forms, inaddition to a programming language's ability to process objects differently depending on their data type or class. More specifically, it is the ability to redefine methods for derived classes.

Class | Objects|Sub Object|Containers

Class can be defined as a blueprint pertaining to an object, classess usually contain methods and variables of an object (pre-defined properties). Thus, it can be defined as a specific instance of a class. Whilst a Sub object is an object that sits inside another object in the same category. A container uses a dynamic structure and is class,data structure or abstract data type, whose instances are collections of other objects. In other words a container stores objects in an organised way that follows a specific set of rules pertaining to access.

Types Of Classes

Base & Derived Classes

Base is the parent class of a derived class, a class that facilitates the creation of other classes that can reuse the code implicitly inherited from the base class with the exception of constructors and destructors.

Derived is a class created or derived from another existing class, concieved through the process of inheritance, which entails passing and using existing properties of that class/object.

Abstract & Concrete Classes

Abstract class is a template definition of methods and variables of a class that contains one or more abstracted methods. Similiarly to derived classes, abstract essentially inherit properties of other classes, hence why abstracts methods contain no implementation.

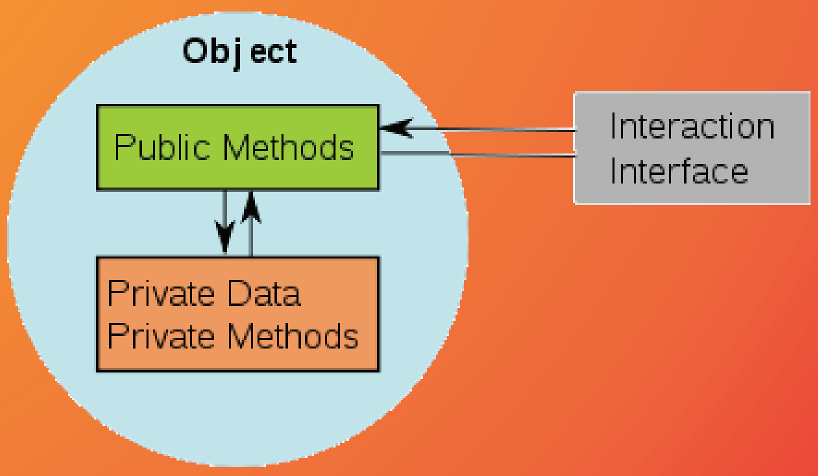
Concrete is a class that has an implementation for all of it's methods that were inherited from abstract or implemented via interfaces.

An instance of a derived class (creature) and base class (persona):

```
class persona {

public:
        void setname(string i) { name = i; };
        string getname() { return name; };

private:
        string name = "";
};
class creature : public stats, public persona {
public:
        int getWorldx() { return worldx; };

        int getWorldy() { return worldy; };

        int getDMG() { return dmgtaken; }

        bool isAlive() {
                bool alive = true;
                if (dmgtaken >= maxHP()) {
                        alive = false;
                }
                return alive;
```

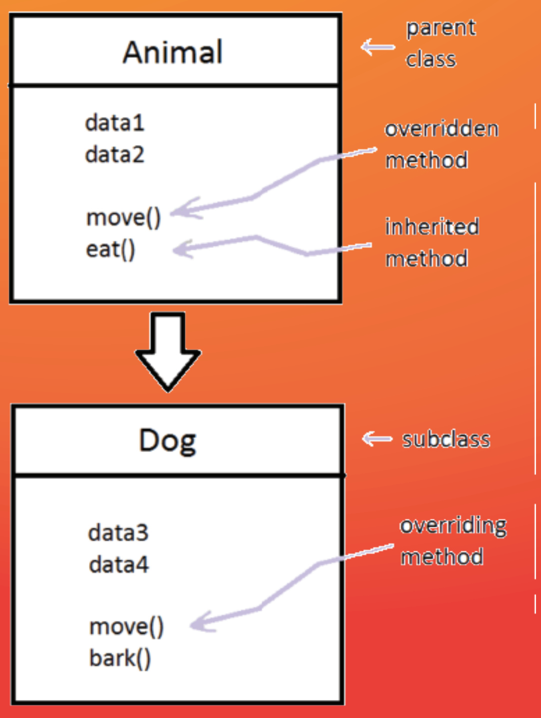
Object Oriented Interfaces

In OPP, an interface is essentially a programming structure (syntax) and paradigm or way to interact with methods , stating the enforcment of certain properties on an object (class) that includes that interface and what it could respond to. For instance a “box” is an object. For a given “box”, interface declares the different “inputs” and corresponding “outputs” of that “box”. Interface is a concept of abstraction and encapsulation.



Method Redefinition, Overriding & Overloading

Method overloading in OPP is when there are two methods of the same class with the same name, but different parameter types. Method overriding (also known as redefinition) is a procedure that allows a (child) subclass to provide specific implementation of a method that is provided by one of its super (parent) classes . In otherwords, is when a child class redfines a method inherited from its parent class .



Generics & Templates

Relates to writing “templates” of source codes that the compiler in the process of compilation transforms into source codes, rather than source code that is compiled as is, An example of generic programming can be seen through container classes like arrays, lists or maps that contain a collection of other objects.

A template is the foundation of genric programming, essenitally a blue print for creating a generic class or a function with code that can be modified to fit a specific situation.