OOP (Object Oriented Programming: 객체지향개발)

1) 추상화(Abstraciton)

- 공통의 속성이나 기능을 묶어 이름을 붙이는 것
- 객체 지향적 관점에서 클래스를 정의하는 것을 바로 추상화라고 정의 내릴 수 있겠다.
- 좀 더 살펴보면 물고기, 사자, 토끼, 뱀이 있을 때 우리는 이것들을 각각의 객체라 하며 이 객체들을 하나로 묶으려 할 때, 만약 동물 또는 생물이라는 어떤 추상적인 객체로 크게 정의한다고 하자. 이때 동물 또는 생물이라고 묶는 것을 추상화라고 한다. 객체지향에서 추상화라는 개념은 '객체에서 공통된 속성과 행위를 추출하는 것'을 의미한다. 예를 들어 홍길동 교수, 이순신 교수, 강감찬 교수가 있다고 하자. 이 교수들을 객체로 보고 공통된 속성과 오퍼레이션으로 '교수'라는 클래스를 정의한다고 해보자. 공통된 속성으로 이름, 주민등록번호, 강의 분야, 주소, 전화번호를 추출하고, 공통된 행위로써 '강의하다','채점하다','과정 등록하다'를 추출했다고 하면, 그를 포함한 클래스가 생성된다.

추상화는 다른 객체들과 구분되는 핵심적인 특징들에만 집중함으로써, 복잡도를 관리할 수 있도록 한다. 주의할 점은 추상화는 문제 영역과 관점에 의존적이라는 것이다. 그래서 어떤 영역에서 중요한 것이 다른 영역에서는 그렇지 않을 수도 있다. 예를 들어 학생이라는 클래스와 멀티캠퍼스와 같은 단기 과정을 수강하는 곳의 시스템 내부에서 모델링되는 학생이란 클래스는 내부의 속성과 오퍼레이션이 서로 많이 다를 것이다. 따라서, 하나의 대상에 대하여 목적이나 원하는 기능에 따라 여러 추상화 모델이 생성될 수 있다.

하나의 개념적 덩어리 안에서 여러 메서드들이 다루게 될 재료(recipe) 에 대해 공유하는 부분 또한, 캡슐화가 가지는 팀웍과 우아함이다. 대중음식점이나, 고급레스토랑의 구조와 처리흐름을(work flow) 생각해보면 어떤 잇점들이 있는지 잘 알 수 있다. 주문을 받는 웨이터, 웨이트레스들과 요리사는 주문서를 공유하고, 요리의 결과물을 주고 받으며, 재료의 재고 상태를 공유한다. 필요시 웨이터들은 재료가 떨어졌음을 (메모리고갈 상태를) 고객에게 디버깅 메시지로 알려주기도 한다.

2) 캡슐화(Encapsulation)

- 실제로 구현되는 부분을 외부에 드러나지 않도록 캡슐로 감싸 이용방법만을 알려주는것
- 데이터 구조와 데이터를 다루는 방법들을 결합 시켜 묶는 것. 다시 한번 말하자면 변수와 함수를 하나로 묶는것을 말한다.
- 하지만 무작정 한대 묶으면 되는 것이 아니라 객체가 맡은 역할을 수행하기 위한 하나의 목적을 한데 묶는다고 생각해야한다.
- 또한 데이터를 절대로 외부에서 직접 접근을 하면 안되고 오로지 함수를 통해서만 접근해야하는데 이를 가능하게 해주는 것이 바로 캡슐화이다.

캡슐화를 통해 묶고 숨김을 생각해 보았다. 그런데 왜 묶어야 하고 숨겨야 하는 것일까? 먼저, 묶음으로 인해 프로그램을 바라보는 단위가 커진다. 이전의 프로그래밍 언어인 C언어는 프로그램을 함수 단위로 구조화할 수 있으나, 프로그램 소스가 커지면 이해하기 어렵고 관리가 힘들어 질 수 있었다. 그러나 객체지향 프로그램에서는 프로그램 소스를 클래스 단위로 바라보게 됨으로써 좀더 복잡하고 커다란 소스코드도 쉽게 이해하게 되었다. 왜냐하면 클래스 내부에 여러 함수를 내포할 수 있기 때문에 프로그램 소스 코드를 바라보는 단위가 커졌으며, 그로 인해 프로그램 관리가 좀 더 수월해진 것이다.

두번째, 내부를 숨김으로써 내부를 좀더 자유롭게 변경할 수 있게 되었다. 이전의 함수 중심적인 구조적 프로그래밍 언어에

서는 프로그램 내부에서 데이터가 어디서 어떻게 변경되는지 파악하기 어려웠고, 그로 인해 유지 보수가 힘들었기 때문에 자료를 중심으로 함수가 종속되는 구조가 되기도 하였다. 객체 지향에서는 클래스 내부의 데이터를 외부에서 참조하지 못하도록 차단하여 이러한 폐단을 없앨 수 있다. 이렇게 내부의 데이터나 함수를 외부에서 참조하지 못하도록 차단하는 개념을 정보 은닉Information Hiding)이라고 하며 이것이 바로 캡슐화라는 개념이다.

3) 은닉화 (hiding)

- 은닉이란 내부 데이터, 내부 연산을 외부에서 접근하지 못하도록 은닉(hiding) 혹은 격리(isolation)시키는 것
- 변수에 접근지정자를 private 로 지정한다
- setter , getter 를 사용해 변수의 접근, 제어한다

은닉화는 캡슐화에 비해 비교적 구체적인 개념이다. 은닉화는 캡슐화의 한 개념으로 객체 외부에서 객체내의 자료로의 접근을 제한하고 데이터를 수정,조작하는 동작은 내부에 두고 접근(getter),설정(setter)하는 메소드로 결과만 받는것이다. 이렇게 되면 외부에서는 내부적인 움직임을 알수가 없으며 데이터에 어떤값이 있는지 또는 어떤 변화가 일어나는지 알수없다. 단지 데이터의 접근을 메서드(setter , getter)를 통해 결과만 받을뿐이다. 이런한 것을 은닉화라 한다

1.은닉화는 중요사항이(변수든 메소드든 간에) 밖으로 드러나지 않도록 꼭꼭 감추는것 2.캡슐화는 중요사항을 감춘 상태에서 외부에 그것을 사용할수 있는 방법을 설정하고 외부와 직접적으로 의사소통을 의미 한다

흔히 캡슐화와 혼용된다. 캡슐화의 과정에서, public 으로 공개할지, private 로 숨길지, protected 로 자손들에게만 알려줄지를 서술하게 마련이기 때문이지만, 개념은 캡슐화와 다르다. 고객이 식당에 와서 메뉴판을 보며 고르는 행위를 돕기 위해서 굳이 식당에서 들여오는 계란의 유통구조와 삶는 방법을 메뉴판에 열거할 필요는 없다. 알아서 나아질 것이 없다면 모르는게 약이다란 뜻이다. 너무 많은 인터페이스의 노출은 결정적으로 사용자 불편을 초래한다. 그래서 APPLE 의 미니멀리즘이 성공한거지. (ipod shuffle 같은 mp3 player 등이 좋은 예) 구글 검색에서 쓸데없는 검색결과들이 당신의 시간을 얼마나 좀먹는가? 심지어 인터넷에 당신에게 필요한 정보가 있을지라도, 불필요한 정보의 노출에 의해 필요한 내용을 평생 얻지 못할 수도 있다. 이렇듯 은닉화는 정말 중요한 개념이고, UI 나 인터페이스 설계에서 핵심적인 부분이다.

4) 상속성, 재사용(Inheritance)

- 상위 개념의 특징을 하위 개념이 물려받는 것
- 객체지향의 하이라이트 부분이라고 생각한다. 상속이란 개념이 없으면 객체지향이나 절차지향이나 도진개진
- 간단히 예를 들자면, 자동차라는 부모클래스가 있다. 기름을 먹거나 달리는 기능을 하는 자동차인데, 만약 지붕뚜껑이 열리는 특수한 기능을 추가하고 싶다면 기존의 자동차에서 스포차카를 생성한다. 그러면 스포츠카는 기름도 먹고 달리면서 지붕두껑이 열리는 기능도 갖춘 자동차가 되는 것

상속을 사용하여 좋아지는 것은?

- 먼저, 재사용으로 인해 코드가 줄어든다. 하위 클래스에서 속성이나 오퍼레이션을 다시 정의하지 않고 상속받아서 사용함으로써 코드가 줄어든다. 그리고 좀 더 범용성있게 사용할 수 있다. 참고로 하위 클래스는 상위 클래스가 가지고 있는 모든 자료와 메소드를 물려받아 자유롭게 사용할 수 있지만, 또한 자신만의 자료와 메소드를 추가적으로 덧붙임으로써 새로운 형태의 클래스로 발전하게 된다는 것도 잊지 말자.

5) 다형성(Polymorphism)

- 부모클레스에서 물려받은 가상 함수를 자식 클래스 내에서 오버라이딩 되어 사용되는 것
- 웹사전에 따르면, 다형성의 일반적인 의미는 '다양한 형태로 나타날 수 있는 능력'이라고 한다. 또 다른 웹 사전을 보면, 객체지향에서의 다형성은 '여러 클래스들이 동일한 이름의 오퍼레이션을 서비스하도록 하는 것'이라고 한다.

이러한 다형성은 실제의 코드에서는 '하나의 클래스 내부에 같은 이름의 오퍼레이션을 여럿 정의하거나, 상위 클래스의 오퍼레이션을 하위 클래스에서 다시 정의함'으로써 구현한다.

예를 들어 상위클래스인 Object클래스에 toString(), equals() 메소드가 존재하지만, 하위 클래스인 String과 Date 클래스에도 toString(), equals() 메소드가 존재함으로 볼 수 있다. 이렇게 상위 클래스에 있고 상속받았으나 하위 클래스에서 다시 정의하는 것을 메소드 오버라이딩(Method Overriding)이라고 하며, 메소드 오버라이딩이 다형성이다.

그리고 동일한 이름의 오퍼레이션이 여러개 정의되어 있는데 단지, 매개변수의 타입에 따라 서로 구분될 수 있다. 이렇게 클래스 내부에 동일한 이름의 오퍼레이션을 여럿 정의하는 것이 바로 메소드 오버로딩(Method Overloading)이라고 하는 다형성이다.