2. Object와 Class 이해하기

1. 객체(object)

객체는 사람, 집, 계좌, 차 등 현실 세계에서 우리가 접하는 **사물**들이다.
객체지향 프로그램에서 객체는 자료와 이를 취급할 수 있는 연산을 하나의 **모듈**로 구성한 것이다.

객체 = 자료(속성) + 연산(메서드, 멤버함수)

- 속성((property) : 객체가 가지는 특성 또는 상태이다.
- 메서드(method) : 객체가 실행해야 할 구체적인 연산이다.

2. 추상화(abstraction)

실세계의 객체를 프로그래밍에 반영하려면,
객체를 프로그래밍에 적용할 수 있는 형태로 변형시켜야 한다. 이런 작업을 **추상화**라 한다.
추상화는 사물의 주된 특징은 부각시키고, 나머지는 과감하게 생략하는 방식으로 진행된다.
추상화는 '**명사**와 **동사**'를 이용해서 사물의 특징을 **추출**하는 것이다.

예제 1 : **'집'이라는 Object**를 명사와 동사로 추출해 보자.

명사 : 주인이름, 층수, 평수, 건축년도, 창문수, ...
동사 : 구입하다, 팔다, 청소하다, 고치다, 세를 놓다, ...

예제 2 : 은행 업무에서 **'계좌'**라는 사물을 추상화하면,

명사 : 계좌명, 계좌번호, 비밀번호, 잔고, 이율, 이체한도, ...
동사 : 입금하다, 출금하다, 조회하다, 이체하다, ...

명사는 사물의 **속성**, **동사**는 **메서드**를 나타내는데 사용되었음을 알 수 있다.

그 다음은 추상화 결과를 어떻게 프로그래밍에 접목하느냐? 이다.
추상화를 바탕으로 프로그래밍에서 사용자가 구현한 자료구조가 **'클래스'**이다.
클래스는 **캡슐화**된 구조이며, 객체의 외부 모습은 구체적이 아닌 **추상적**인 형태이다.

3. 클래스(class)

다음은 **추상화**한 '**계좌**'를 Java code 형태처럼 구현한 것이다.

```
public class 계좌
{
      private String 계좌번호 = "389-02-123456";     //속성
      private String 비밀번호 = "1234";
      private long 잔고 = 1000;
      private float 이율 = 6.02F;

      public void 입금하다(long 금액) {               //메서드
                  잔고 = 잔고 + 금액;
      }
      public void 출금하다(long 금액) {
                  잔고 = 잔고 - 금액;
      }
      public long 조회하다() {
                  return 잔고;
      }
      public void 이율변경(float 새이율) {
                  이율 = 새이율;
      }
}
```

클래스는 같은 형태의 속성과 메서드를 갖는 객체를 생성할 수 있는 **틀(templet)**이다.
즉, 클래스는 사용자가 정의한 새로운 자료형으로 객체 타입(object type)이다.

```
계좌 하나 = new 계좌(계좌번호, 비밀번호, .....)      //객체 생성
```

여기서, 계좌 : 클래스 이름
하나 : 생성되는 객체 이름
new : 객체를 생성하기 위한 명령어

결론적으로 객체지향언어에서 객체를 생성하여 사용하려면 먼저 클래스를 만들어야 한다.
모든 객체는 클래스로부터 생성되기 때문이다.

객체가 생성되는 과정을 실체화(instantiation)라 하고,
생성된 객체를 **인스턴스(instance, 사례)**라 한다. 객체와 인스턴스는 같은 뜻으로 사용된다.

4. 캡슐화(encapsulation)

〈데이터 캡슐화〉

데이터 **캡슐화**는 외부 세계로부터 데이터 객체의 자세한 **구현**을 **은폐**하는 것이다.**(정보은닉)**

〈이석호 자료구조론〉

- **캡슐화**는 객체의 **속성**과 **메서드**를 하나로 묶는다.(객체지향언어)
- 캡슐화된 정보를 **접근**하기 위한 문법으로 "private, protected, public"이 있다.

〈캡슐화와 외부에서 접근 가능 여부〉

```
class 사람                                  //캡슐화
{
   private String irum = "홍재연";          //전용(외부에서 접근 불가)
   protected int don = 10000000;            //보호(외부에서 자식은 접근 가능)
   public int get() { return don; }         //공용(외부에서 접근 가능)
}
```

// 데이터 추상화와 캡슐화

- 데이터 추상화와 캡슐화 개념은 **기계-인간** 상호작용에서 많이 사용되고 있다.
- 데이터 추상화와 캡슐화 개념은 기술 지향적 세계에서 널리 보급된 개념이다.
- 데이터 추상화와 캡슐화 개념은 컴퓨터 **프로그램**에서 데이터를 **묶고 구조화**하는데 사용된다.
- 데이터 추상화와 캡슐화를 이용하면 **고품질**의 **프로그램**을 개발할 수 있다.(인터페이스 단순화)
- 데이터 추상화와 캡슐화를 이용하면 프로그램 개발 **시간**과 **인력**을 줄여준다.**(개발 비용 절감)**
- 데이터 추상화와 캡슐화를 이용하면 프로그램 **수정**이 **용이**하다.(**낮은 결합도 높은 응집도**)
- 데이터 추상화와 캡슐화를 이용하면 **재사용성**을 높여준다.

추상화	• 객체의 **명세**(specification)와 **구현**을 **분리**한다. • 어떤 **사물(객체)**에서 **핵심 부분**만을 **추출**하여 **이해**하기 쉽도록 표현하는 기술이다.
캡슐화	• 외부 세계로부터 데이터 객체의 자세한 구현을 **은폐**하는 원리이다.**(정보은닉)** • 사용자는 객체의 **기능**과 **사용법**만 알면 쉽게 사용할 수 있다.**(인터페이스)**
추상 자료형	• 객체의 **명세**가 **구현**으로부터 **분리**된 **데이터 타입**이다.(추상 데이터 타입, ADT) • 객체의 내부 **표현**이나 연산의 **구현**은 기술하지 않은 것이다. • 객체가 **무엇**(what)을 하는지? **사용설명서**와 같다. • 객체가 내부적으로 **어떻게**(how) **동작**하는지?는 알려주지 않는다. • **이름**만 있고 내용은 없는 **빈껍데기**와 비슷하다.

- **추상화, 캡슐화, 정보은닉**은 서로 다른 개념이지만 **밀접**하게 연관되어 있다.
- **추상화**는 사용자에게 중요한 것에만 집중할 수 있도록 한다.
- **캡슐화**는 연관성 있는 것을 묶는 것이고, 사용자에게 중요하지 않은 것을 숨길 수 있다.
- **정보은닉**은 사용자에게는 중요하지 않는 것(원시코드)을 숨기는 것이다.
- **정보은닉**은 타인이 나의 원시코드 파일을 이용만 하고 내용은 보지 못하게 하는 은닉이다.

추상자료형(abstract data type, ADT)

〈추상자료형 정의〉

객체의 **명세**와 이들 **객체에 대한 연산의 명세**가 **객체의 표현**과 **연산의 구현**으로부터 **분리**된 방식으로 구성된 **데이터 타입**이다.

〈이석호 자료구조론〉

- 추상자료형 정의는 참 **복잡**한 느낌을 받는다. 이해가 잘되지 않을 수도 있다.
- 해서, 추상자료형 예를 들어서 설명한다.

// 예제 1 : 스택(stack) S에 대한 추상자료형

객체	0개 이상의 원소를 가진 유한의 순서리스트 (여기서, 객체는 **데이터**를 의미)
연산 (함수)	create(**s**) ::= 스택 **s** 생성 is_empty(**s**) ::= 스택 **s**가 비어 있는지 검사 is_full(**s**) ::= 스택 **s**가 가득 찼는지 검사 push(**s, e**) ::= 스택 **s**의 맨 위에 요소 **e**를 삽입 pop(**s**) ::= 스택의 맨 위에 있는 요소를 삭제

- **추상자료형 = 객체 + 연산** (**연산**에서는 **세부사항** 무시하고 **기능**을 강조)
- **추상자료형**은 사람마다 조금씩 다르게 정의할 수 있다.

// 예제 2 : 객체지향언어 C++에서 추상자료형

```
class Circle                    //클래스 선언부(추상자료형)
{  public:                      //C++에서 추상자료형은 키워드 class를 이용하여 정의함
      int radius;               //원의 반지름
      double getArea();         //원의 넓이를 구하는 함수
}
double Circle::getArea()        //클래스 구현부 (외부로부터 숨겨짐, 정보은닉)
{
  return radius * radius * 3.14; //원의 넓이 구하기 (구체적으로 구현)
}
```

- 클래스 **선언부**가 **추상자료형**에 해당한다.
- 추상자료형의 큰 특징 중 하나는 **사용자**와 **구현자**의 **분리**이다.
- 어떤 사용자가 추상자료형을 **선언**하고, 다른 사람이 선언된 것을 **구현**할 수 있는 것이다.
- 추상자료형은 **사용 관점**과 **구현 관점**을 명확히 **분리**하여 프로그램 개발을 유용하게 한다.
- 클래스와 추상자료형은 완전히 **같은 것**은 **아니다.**(클래스는 추상자료형 이외 다른 것이 포함됨)
- Java의 **interface**는 추상자료형의 한 종류이다. (명세와 구현을 분리)

먼저, 자바 응용프로그램을 하나 살펴본다.

```
import java.lang.*;                  //java.lang 패키지의 모든 클래스를 사용할 수 있다.
class A{
  protected static int kor;          //국어 점수
  protected static int eng;          //영어 점수
  public void set(int k, int e){     //(국어, 영어) = (70, 90)으로 대입
    kor = k;
    eng = e;
  }
}
class B extends A{                   //상위클래스 : A, 하위클래스 : B
  private int sum;
  public int add(){                  //합을 구한다.
    sum = kor + eng;
    return sum;
  }
}
class C extends A{
  private float avg;
  public float average(){            //평균을 구한다.
    avg = (kor + eng) / 2f;
    return avg;
  }
}
public class Test{
  public static void main(String args[]){
    A a = new A();
    B b = new B();
    C c = new C();

    a.set(70, 90);                   [실행결과]
    System.out.println(b.add());      160
    System.out.println(c.average());  80.0
  }
}
```

- 관례적으로 클래스 이름의 첫 문자는 대문자로, 메서드는 소문자로 정의한다.
- 메서드 main()은 C의 함수 main()과 같은 역할을 한다.
- 매개변수 args[]는 프로그램 실행시 명령어 행에서 String형으로 자료를 전달받는다.
- 사용자가 명령어 행에 입력한 인자의 순서에 따라 배열에 저장된다.

기출문제 분석

1. 객체지향 개념에서 캡슐화에 대한 설명으로 옳지 않은 것은? [2008년 국가 7급]

① 캡슐화를 하면 객체들 사이의 결합도가 높아지고, 객체의 응집도가 향상된다.
② 객체의 명세와 실체를 분리하는 기법으로 캡슐화된 객체들의 재사용성을 향상시킬 수 있다.
③ 객체의 상세한 내용을 객체 외부로부터 숨기고, 단순히 메시지만으로 객체와의 상호작용을 하게 하는 것을 캡슐화라고 한다.
④ 인터페이스를 단순화시킬 수 있고, 프로그램 변경에 대한 오류의 파급 효과가 적다.

♤ **객체지향 개념에서 캡슐화**

• 캡슐화를 하면 객체들 사이의 결합도가 높아지고, 객체의 응집도가 향상된다.(×)
→ 객체지향 개념에서 캡슐화는 객체들 사이의 결합도는 낮아지고 응집도는 높아진다.

〈캡슐화 정리〉

㉠ 캡슐화는 클래스, 인스턴스, 생성자, 소멸자 같은 객체지향 방식의 기초를 형성한다.
㉡ 캡슐화를 정확하게 표현하지 못하면 상속성과 다형성 표현도 잘못하게 되어 있다.
㉢ 캡슐화는 프로그램 변경에 대한 오류 파급 효과를 줄인다.
㉣ 캡슐화는 인터페이스를 단순하게 만든다.
㉤ 캡슐화는 실제 구현 내용 일부를 외부에 감추어 은닉한다.
㉥ 캡슐화된 객체들은 재사용이 용이하다.
㉦ 캡슐화는 객체들 사이의 **결합도는 낮아**지고 **응집도는 높아**진다.

〈캡슐화와 정보은폐〉

```
class 사람                              //캡슐화
{
    private String irum = "홍재연";      //전용(정보은폐)
    protected int don = 10000000;       //보호(자식은 접근 가능, 일부 정보은폐)
    public int get() { return don; }    //공용(외부에서 접근 가능)
}
```

• 정보은폐가 곧 데이터 캡슐화는 아니다.
• 정보은폐의 목적은 변경에 대한 유연성을 제공하는 것이다.
• 정보은폐는 복잡하거나 변경 가능한 설계는 뒤로 숨기는 기본 설계 원리이다.
• 정보은폐 원리에 따라 설계된 시스템은 변경에 대한 파급효과가 지역적이 된다.
• 해서, 정보은폐 원리에 따라 설계된 시스템은 변경에 따르는 비용이 상대적으로 적다.

정답 : ①

2. 다음 중 사용자에게 해당 객체의 기능(서비스)과 사용법만 제공해 사용하기 쉽게 하고 내부는 함부로 변경할 수 없게 감추는 내용에 해당하는 가장 적절한 객체지향 개념은? [2022년 군무원 7급]

① 무결성
② 캡슐화
③ 상속성
④ 다형성

♤ **캡슐화**

- 캡슐화는 객체의 **속성**과 **메서드**를 하나로 묶고, 내부 표현은 사용자로부터 **감추는 원리**이다.
- 사용자는 객체의 기능과 사용법만 알면 쉽게 사용할 수 있다.
- 캡슐화는 프로그램 **변경**에 대한 오류 파급 효과를 줄인다.

정답 : ②

3. 다음에서 설명하는 객체지향 프로그래밍의 특징은? [2019년 지방 9급]

○ **객체를 구성하는 속성과 메서드가 하나로 묶여 있다.**
○ **객체의 외부와 내부를 분리하여 외부 모습은 추상적인 내용으로 보여준다.**
○ **객체 내의 정보를 외부로부터 숨길 수도 있고, 외부에 보이게 할 수도 있다.**
○ **객체 내부의 세부 동작을 모르더라도 객체의 메서드를 통해 객체의 기능을 활용할 수 있다.**

---

① 구조성
② 다형성
③ 상속성
④ 캡슐화

♤ **객체지향 프로그래밍 – 캡슐화**

- 캡슐화의 주 목적은 **외부**에 **인터페이스**는 **공개**하고, **구현**은 **숨기는** 것이다.**(정보은닉, 블랙박스)**
- **인터페이스(interface)**는 서로 다른 두 시스템 사이에서 정보를 주고받는 **상호작용 접점**이다.
- **구현(implement)**은 자세하게 기술된 **원시코드**이다.(**원시코드**는 외부에 숨긴다)
- **외부**에 정보를 숨기면, 외부에 영향을 주지 않고 객체 내부의 **구현**을 **변경**할 수 있다.

정답 : ④

4. 다음 중 객체지향 언어의 특징으로 옳지 않은 것은? [2008년 계리]

① 구조화　　② 추상화
③ 상속성　　④ 다형성

♤ **객체지향 언어**

- 구조화는 객체지향 언어가 아닌 언어도 가질 수 있는 일반적인 특징이다.
- 예 : C는 객체지향 언어가 아니지만 구조화를 할 수 있다.
- 프로그래밍 언어에서 구조화는 구조적 프로그래밍을 의미한다.
- 구조적 프로그래밍은 goto문을 없애거나 goto문에 대한 의존성을 줄여주는 것이다.

정답 : ①