OOP(Object Oriented Programming) 개념 정리

객체

- 의사나 행위가 미치는 대상 (사전적 의미)
- 구체적, 추상적 데이터 단위

→ 쉽게 접근하면, 현실 세계에서 우리를 둘러싸고 있는 사물, 사람, 동물, 식물 등과 같이 어떠한 **특정 행동을 하거나 당하는 대상!**

객체 지향 프로그래밍 vs 절차 지향 프로그래밍

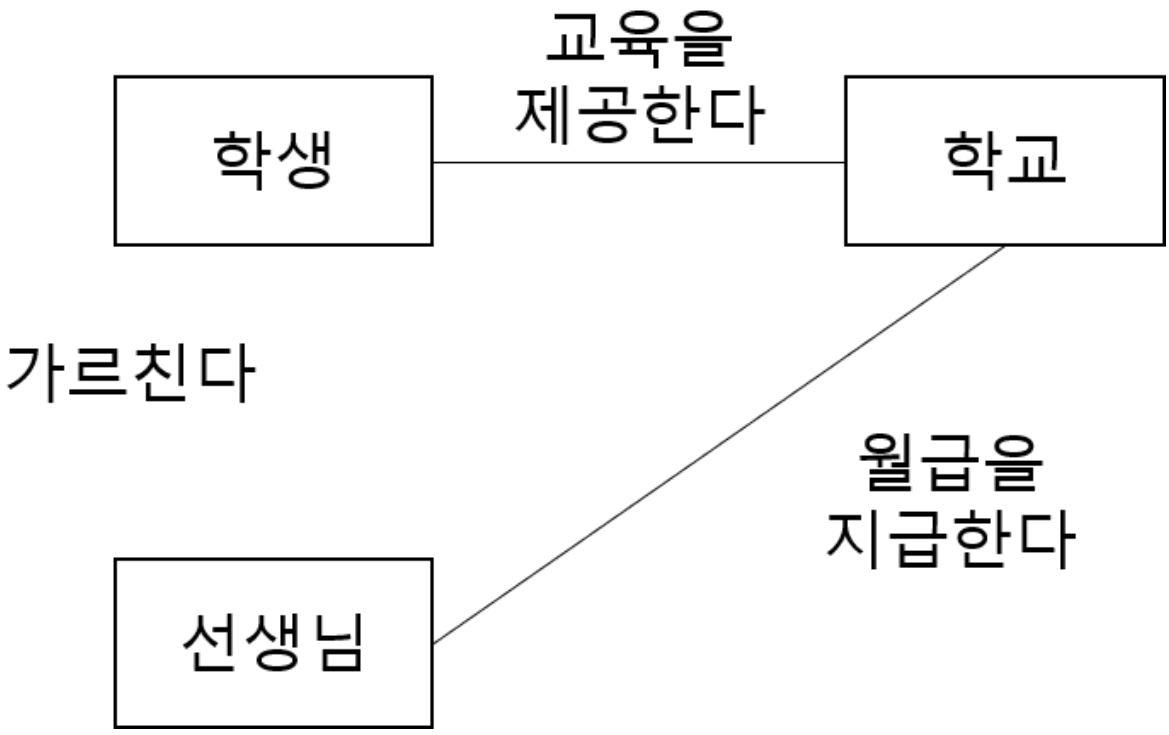
1. 절차 지향 프로그래밍

- 어떠한 행위에 대해서 순차적으로 접근하는 방식

2. 객체 지향 프로그래밍

- 일종의 '모듈' 처럼 행위를 접근하여 해석한다고 생각!

- 어떠한 객체 간의 협력(Collaboration) 관계가 존재
- 각 객체의 기능(역할, 책임, 객체가 해야 하는 일)을 구현함으로써 기능 개별적으로도 접근이 가능해지며, 다른 객체와의 관계를 통한 확장성이 가능해짐



실생활에서 객체를 찾아 클래스로 구현해보기

0. 자바에서 주의할 점, 참고할 점!

- 한 자바 파일 내에 public 클래스를 포함한 두 개 이상의 클래스가 있다면, public 클래스 이름이 파일명과 같아야 함
- 클래스명이 파일 명과 같아야 함
- 한 자바 파일 내에 public 클래스가 없는 경우, 그 중 한 클래스 명이 파일 명과 같아야 함
- 변수나 메서드 명이 여러 단어로 구성된 경우, 마치 낙타의 등과 같이 구불구불한 **camel notation**(카멜 표기법; 자바에서 시작; 암묵적인 규약) - **단, 소문자로 시작하여 적용하기!** 💓
- 패키지명은 소문자로, 클래스명은 대문자로!
- 한 자바 파일 내에 존재할 수 있는 public 클래스는 단 한 개!

https://parfum.tistory.com/593

1. 구현해볼 객체 찾기

- 온라인 쇼핑몰에 회원 로그인을 하고 여러 판매자가 판매하고 있는 제품 중 하나를 골라서 주문
- 아침에 회사에 가는 길에 별다방 커피숍에 들러서 아이스 카페라떼를 주문
- 성적확인을 위해 학사 관리 시스템에 로그인하여 수강한 과목들의 성적을 확인

2. 구현해볼 객체 선정

- 학생 정보를 구현
  1) 학생 번호
  2) 학생 이름
  3) 전공 코드
  4) 전공명
  5) 학년
  6) 학점

```
package com.java.ch2;

public class Student {

  private int sNum;//학생 번호
  private String sName;//학생 이름
  private int majorCode;//전공 코드
  private String majorName;//전공명
  private int grade;//학년
  private char credit;//학점
  public int getsNum() {
    return sNum;
  }
  public void setsNum(int sNum) {
    this.sNum = sNum;
  }
  public String getsName() {
    return sName;
  }
  public void setsName(String sName) {
    this.sName = sName;
  }
  public int getMajorCode() {
```

```
    return majorCode;
  }
  public void setMajorCode(int majorCode) {
    this.majorCode = majorCode;
  }
  public String getMajorName() {
    return majorName;
  }
  public void setMajorName(String majorName) {
    this.majorName = majorName;
  }
  public int getGrade() {
    return grade;
  }
  public void setGrade(int grade) {
    this.grade = grade;
  }
  public char getCredit() {
    return credit;
  }
  public void setCredit(char credit) {
    this.credit = credit;
  }


}
```

- 쇼핑몰 주문 객체 만들기

  1) 주문 번호

  2) 구매자 아이디

  3) 판매자 아이디

  4) 상품 코드

  5) 주문 일자

```
package com.java.ch2;

import java.util.Date;

public class Order {
  //1. 주문번호
  private int orderId;
  //2.구매자 아이디
  private String buyerId;
  //3.판매자 아이디
  private String sellerId;
  //4.상품코드
  private int productId;
  //5.주문일자
```

```
  private Date orderDate;
  public int getOrderId() {
    return orderId;
  }
  public void setOrderId(int orderId) {
    this.orderId = orderId;
  }
  public String getBuyerId() {
    return buyerId;
  }
  public void setBuyerId(String buyerId) {
    this.buyerId = buyerId;
  }
  public String getSellerId() {
    return sellerId;
  }
  public void setSellerId(String sellerId) {
    this.sellerId = sellerId;
  }
  public int getProductId() {
    return productId;
  }
  public void setProductId(int productId) {
    this.productId = productId;
  }
  public Date getOrderDate() {
    return orderDate;
  }
  public void setOrderDate(Date orderDate) {
    this.orderDate = orderDate;
  }


}
```

- 회원 관리 시스템

  1) 사용자 아이디

  2) 사용자 비밀번호

  3) 사용자 주소

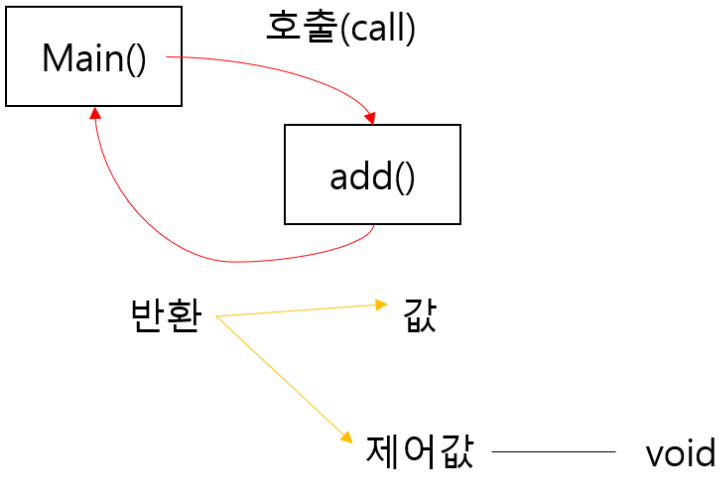
  4) 사용자 핸드폰 번호

```
package com.java.ch2;

public class UserInfo {
  //회원관리

  //1.사용자 아이디
```

```
  private String userId;
  //2.사용자 비밀번호
  private String userPassword;
  //3.사용자 주소
  private String address;
  //4.사용자 핸드폰 번호
  private long phoneNumber;
  public String getUserId() {
    return userId;
  }
  public void setUserId(String userId) {
    this.userId = userId;
  }
  public String getUserPassword() {
    return userPassword;
  }
  public void setUserPassword(String userPassword) {
    this.userPassword = userPassword;
  }
  public String getAddress() {
    return address;
  }
  public void setAddress(String address) {
    this.address = address;
  }
  public long getPhoneNumber() {
    return phoneNumber;
  }
  public void setPhoneNumber(long phoneNumber) {
    this.phoneNumber = phoneNumber;
  }

}
```

함수와 메서드

1. 메서드 : 함수의 일종
   - **클래스 내부에 있는 함수**
   - "멤버 함수(member function)"
   - 메서드 명은 **그 객체를 사용하는 객체(협력관계에 의해서)에 맞게** 짓는 것이 좋음[예: getStudentName]
2. 함수 : **"서브루틴(subroutine)"**
   - 함수가 호출된 후, **수행이 완료되면, 제어가 반환됨** 💓(이때 값이 반환되거나, 제어값만 반환됨)



- **함수를 한 번 구현해두면, 여러 곳에서 활용할 수** 있기 때문에 유연하고 유용 ❤️
- 공통된 기능을 한 군데에서 구현해둔 경우, 내용 수정이 필요한 경우 수정이 용이 ❤️

🌟 함수 구성하기!🌟

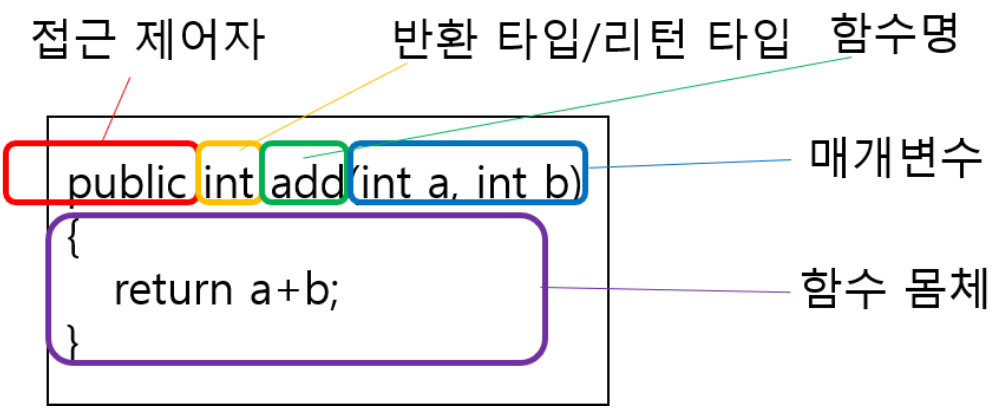
a. 접근제어자

b. 반환타입(수행만 하는 경우 void!)

c. 함수명

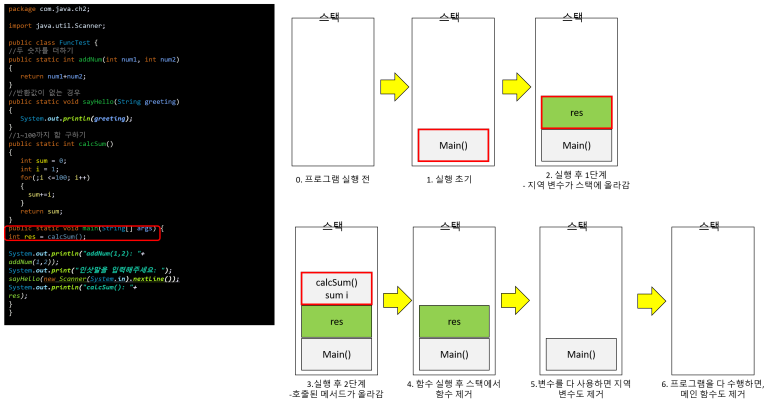
d. 매개변수(없다면, 비워두어도 됨!) - 데이터형과 함께 매개변수 명을 적어주어야 함

e. 함수 몸체



🌟 함수 호출과 스택 메모리 🌟

- 스택 : LIFO(Last In First Out)
- 나중에 힙 구조와 비교!



메서드 만들기

- static 키워드로 별도의 객체 생성 없이 접근하는 메서드를 연습해보기
- F2를 누르면 refactor- rename이 가능함(패키지, 클래스명 등에 대해서)

```
package com.java.ch2;

import java.util.Scanner;

public class FuncTest {
  //두 숫자를 더하기
  public static int addNum(int num1, int num2)
  {
    return num1+num2;
  }
  //반환값이 없는 경우
  public static void sayHello(String greeting)
  {
    System.out.println(greeting);
  }
  //1~100까지 합 구하기
  public static int calcSum()
  {
    int sum = 0;
    int i = 1;
    for(;i <=100; i++)
    {
      sum+=i;
    }
    return sum;
  }
  public static void main(String[] args) {
    System.out.println("addNum(1,2): "+
          addNum(1,2));
```

```
    System.out.print("인삿말을 입력해주세요: ");
    sayHello(new Scanner(System.in).nextLine());
    System.out.println("calcSum(): "+
          calcSum());
  }
}
```