객체지향프로그램밍이란

클래스를 설계해서
객체로 코딩한다

프로그래밍 기법

절차지향 프로그래밍
객체지향 프로그래밍

간단한 예제  를 통해
프로그램의 기법을 알아 봅시다.

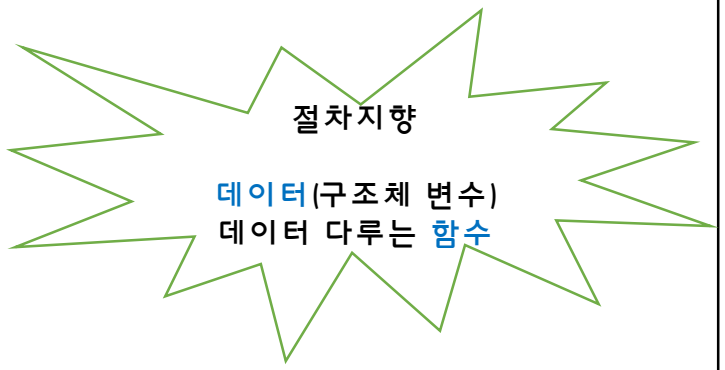
절차지향

```
main() {
//변수선언

학번
이름
주소

//값저장
학번= "S100"
이름= "홍길동 "
주소= "서울시 노원구"

//출력
System.out.println(학번);
System.outprintln(이름);
System.out.println(주소);
}
```

구조적
(함수기반)

```
main(){
        학번
        이름
        주소
        입력하기(학번, 이름, 주소)
        출력하기(학번, 이름, 주소)
    }

입력하기( 학번, 이름, 주소){
  학번= "S100" ;
  이름= "홍길동" ;
  주소= "서울시 노원구 ";
}

 출력하기( 학번, 이름, 주소){
   System.out.println(학번);
   출력(이름);
   출력(주소);
}
```

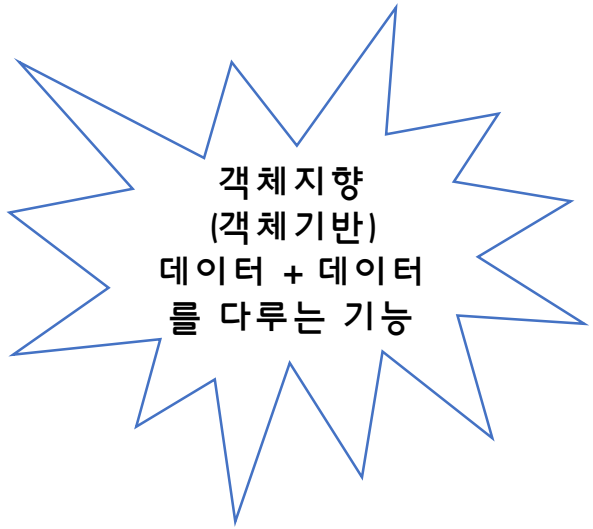
```
struct  Student{
    학번
    이름
    주소
}


 입력하기 ( Student  obj){
  obj.학번 =   "S100"  ;
  obj.이름 =  "홍길동   ";
  obj.주소  =  "서울시 노원구   ";
}


 출력하기(Student obj){
   System.out.println(obj.학번);
   System.out.println(obj.이름);
   System.out.println(obj.주소);
}
```




```
main(){

    Student obj  ;    //구조체변수

    입력하기( obj );

    출력하기(  obj);

}
```

학생정보 출력

```
class  Student{
            학번
            이름
            주소

          입력하기(학번, 이름, 주소){
             this.학번 = 학번;
             this.이름 = 이름;
             this.주소 = 주소;
          }
           출력하기( )
         {
             System.out.println(학번);
             System.out.println(이름);
             System.out.println(주소);
         }
}
```




```
//프로그램 시작
main(){

    Student  obj = new Student();
    obj.입력하기(  "S100" ,  "홍길동" ," 서울시 노원구  ");
    obj.출력하기();
}
```

클래스 무엇?
객체가 무엇?

클래스 용도:
1. 사용자 정의 Type을 위한 class
2. 라이브러리를 작성을 위한 class
3. 프로그램 작성을 위한 class

Type

```
class  Student{
        학번
        이름
        주소

      입력하기( 학번, 이름, 주소){
        this.학번 = 학번;
        this.이름 = 이름;
        this.주소 = 주소;
      }
       출력하기()
     {
        System.out.println(학번);
        System.out.println(이름);
        System.out.println(주소);
     }
}
```

Program

```
class Program{

   public static void main( String[] args){

        Sudent s = new Student();
        s.입력하기( "12" ," 홍 ",  "서울 ");
        Calculator  cal= new Calculator();
        int result= cal.add(5,3);

   }

}
```

Library

```
class  Calculator{

   public int add( int su1, int su2){
        return su1+su2;
    }

   public int sub( int su1, int su2){
       return  su1- su2;
   }
}
```

Day

```
class Day{

    String date;
    String content;


    void input ( String date, String content){
          this.date  = date;
          this.content = content;
    }

    void print() {
       System.out.println( date);
       System.out.println( content );
     }

}
```

일정 정보를 저장할 Type 만들기

주의 사항!!

클래스는 일정정보를 담을 수 없다.
일정정보를 담기위해서는

클래스를 이용해서 변수를 만드는 과정을 거쳐야 한다

```
Day day = new Day();
day.input(  "2024-08-11"   ,"  빨래하기"  );
day.print();
```

input, print매서드는 일정정보를 다루는 매서드이다
Day 객체가 생성된 이후 (즉 day정보를 담을 수 있는)
변수가 만들어진 이후 부터 사용가능하다.!
반드시 new (메모리확보) 이후 부터 사용가능하다

DayMgt

```
class DayMgt{

        public static void main( String[] args) {

                Day[] days = new Day[10];
                int index=0;

                 loop:   while(true) {
                    System.out.println(  "1.등록, 2. 조회, 2.종료 );
                    switch( menu){
                     case 1:
                      System.out.println(  "등록"  );
                      break;
                     case 2:
                      System.out.println(  "조회"  ) ;
                      break;
                     case 3:
                       System.out.println(  "종료"  );
                       break loop;
                    }
                }
        }

}
```

DayMgt

```
class DayMgt{
        Day[ ] days= new Day[10];
        int index=0;

        public void register(){
          //일정 등록하기
        }
        public void print(){
           //일정 조회하기
        }
           //
        public void run( ){   //

                loop:   while(true) {
                      System.out.println(  "1.등록, 2. 조회, 2.종료 );
                       switch( menu){
                        case 1:
                          register(); break;
                        case 2:
                          print();break;
                        case 3:
                          break loop;
                       }
                  }
        }
       public static void main(Stting[] args){
                DayMgt mgt= new DayMgt();
                mgt.run();
       }
}
```

용어정리

자료형

변수

class

객체

new

인스턴스

생성자

자료형

클래스

변수

객체

자료형(사용자정의 자료형 이제 클래스라고 부른다)

```
int a;
```

변수(이제 객체라고 부른다)
:참조형변수이다

```
Customer s = new  Customer( "홍길동" , 25);
```

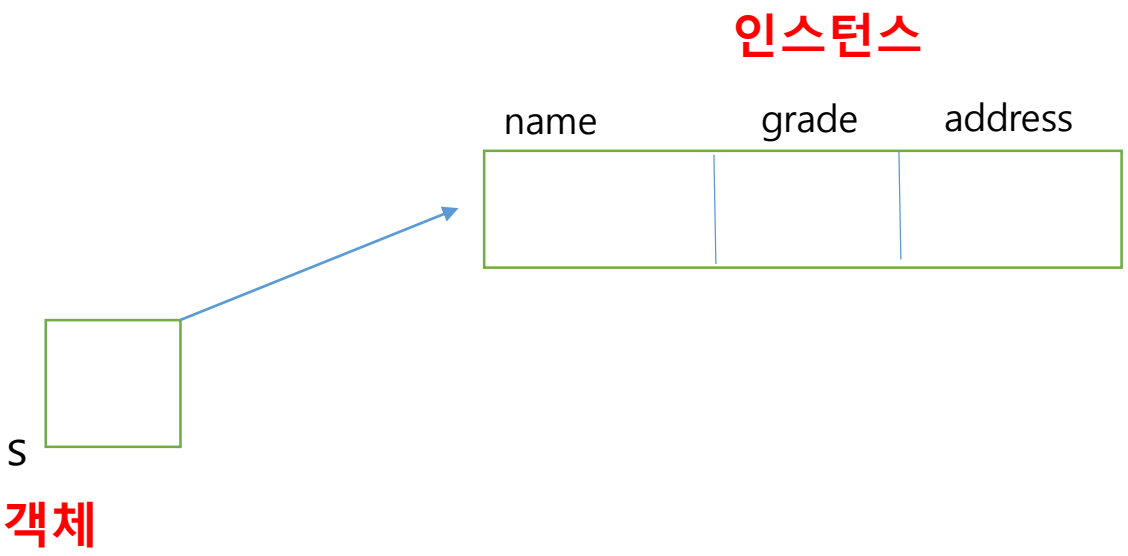
★클래스 : 사용자정의 자료형 (구조화된 데이터 변수)이다.
객체 : 객체참조형변수 이다.
인스턴스 : 객체참조변수가 참조하는 실체를 말한다.

클래스 : 객체를 만들어 내기 위한 설계도
객체: 클래스 모양 그대로 생성된 메모리

클래스 type

```
class Student{
    String name;
    int grade;
    int address;
}
```

```
Student s = new Student();
```



```
int a;
a= 78;

int a= 78;
```

```
Customer c = new Customer( "홍길동" ,25);
```

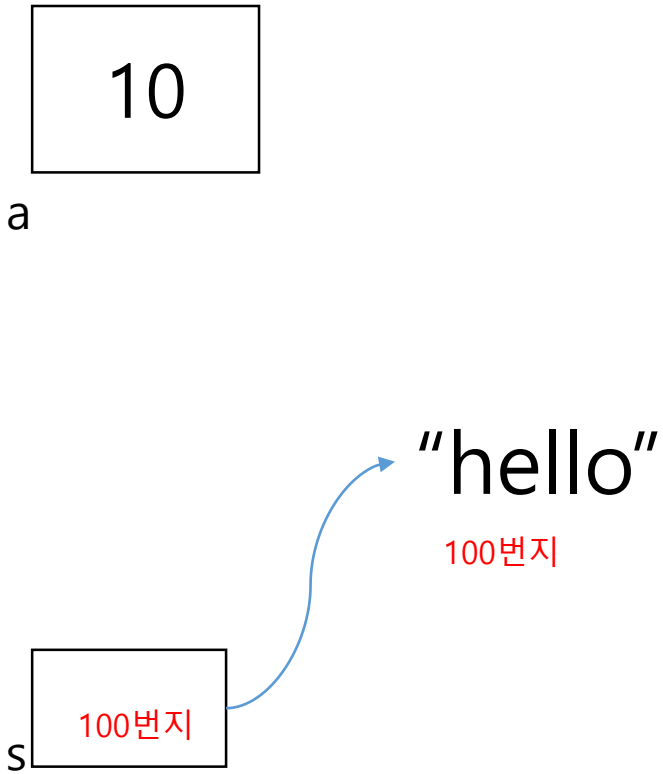
변수 선언과 초기화
선언과 초기화가 구분되었다.

객체 생성과 초기화
초기화는 반드시 객체 생성 시에만 가능하다

String 맛보기

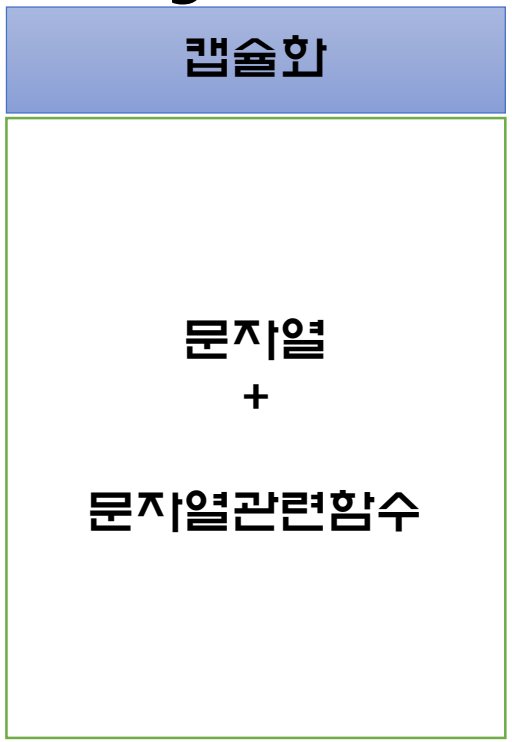
```
int a=10;
```

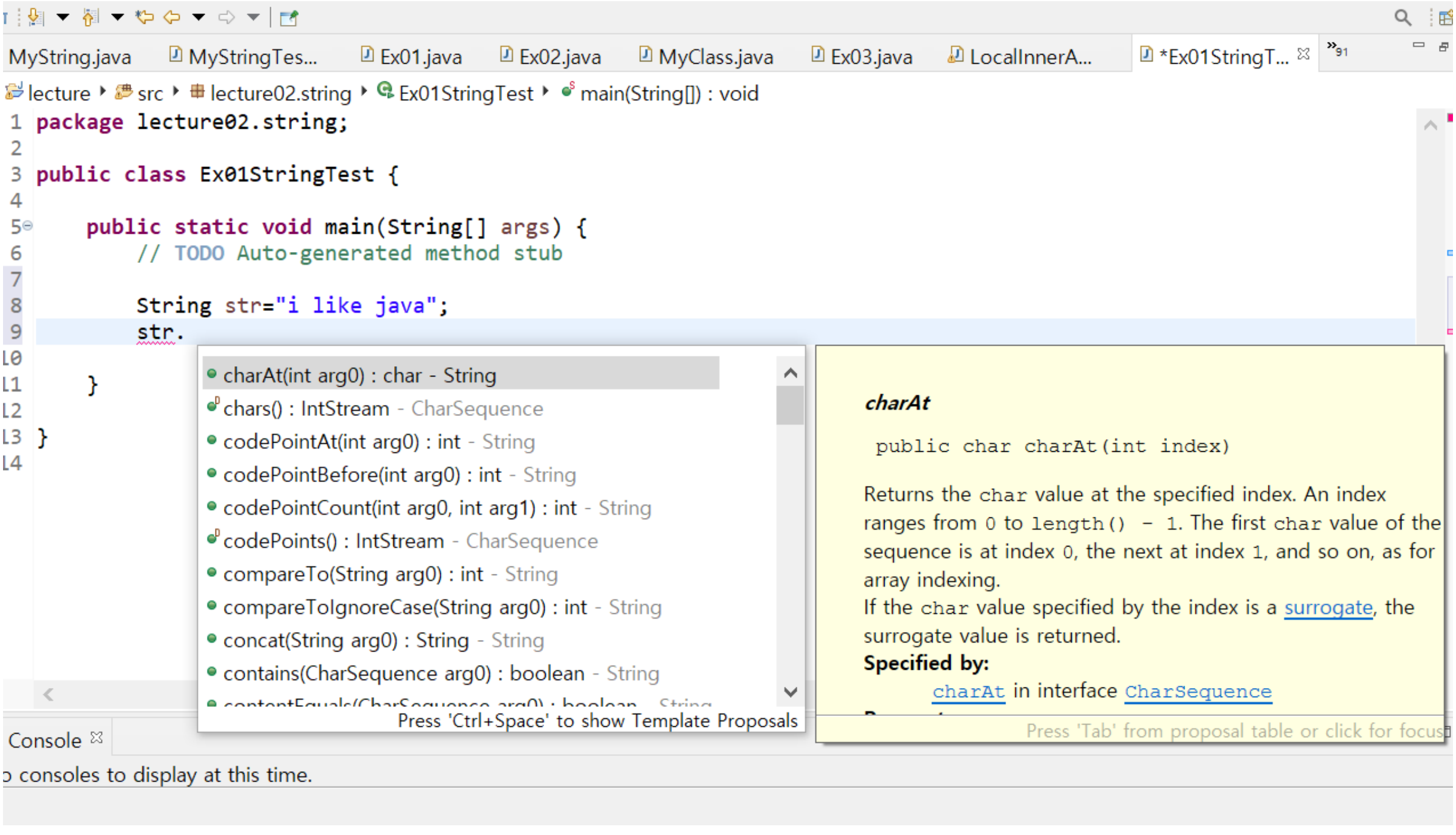
```
String s =new String(  "hello"  );
String s2=  "hello"  ;
```



관련있는 데이터와 함수를 묶어서 하나의 단위로 제공한다.

String




MyString.java
MyStringTes...
Ex01.java
Ex02.java
MyClass.java
Ex03.java
LocalInnerA...
*Ex01StringT...
lecture ▸ src ▸ lecture02.string ▸ Ex01StringTest ▸ main(String[]) : void
1 package lecture02.string;
2
3 public class Ex01StringTest {
4
5 public static void main(String[] args) {
6 // TODO Auto-generated method stub
7
8 String str="i like java";
9 str.
10
11 }
12
13 }
14
charAt(int arg0) : char - String
chars() : IntStream - CharSequence
codePointAt(int arg0) : int - String
codePointBefore(int arg0) : int - String
codePointCount(int arg0, int arg1) : int - String
codePoints() : IntStream - CharSequence
compareTo(String arg0) : int - String
compareToIgnoreCase(String arg0) : int - String
concat(String arg0) : String - String
contains(CharSequence arg0) : boolean - String
Press 'Ctrl+Space' to show Template Proposals
charAt
public char charAt(int index)
Returns the char value at the specified index. An index ranges from 0 to length() - 1. The first char value of the sequence is at index 0, the next at index 1, and so on, as for array indexing.
If the char value specified by the index is a surrogate, the surrogate value is returned.
Specified by:
charAt in interface CharSequence
Press 'Tab' from proposal table or click for focus
Console
o consoles to display at this time.

```
public class Ex01StringTest {

public static void main(String[] args) {


        String str="java";
        System.out.println(str.charAt(0));
        System.out.println(str.compareTo("java"));
        System.out.println(str.equals("java"));

        }

}
```

메서드실행결과
결과값 예상하기

객체 멤버 접근

. 연산자

클래스형으로 생성된 객체의
멤버를 접근할때 사용

```
public class Circle {

	private int radius;
	private String name;

	public Circle()
	{

	}

	public  Circle( int  m_radius  , String m_name)
	{
		radius= m_radius;
		name = m_name;

	}
	public double getArea(    )
	{
		return 3.14*radius*radius;
	}
|
```

```
        public static void main(String[] args)
        {

                Circle pizza = new Circle(10,"자바피자" );
                Circle pizza1 = new Circle(20,"자바피자중" );
                Circle pizza2 = new Circle(30,"자바피자대" );

                System.out.println(    pizza.getArea()   );
                System.out.println(    pizza1.getArea()   );
                System.out.println(    pizza2.getArea()   );

        }
}
```

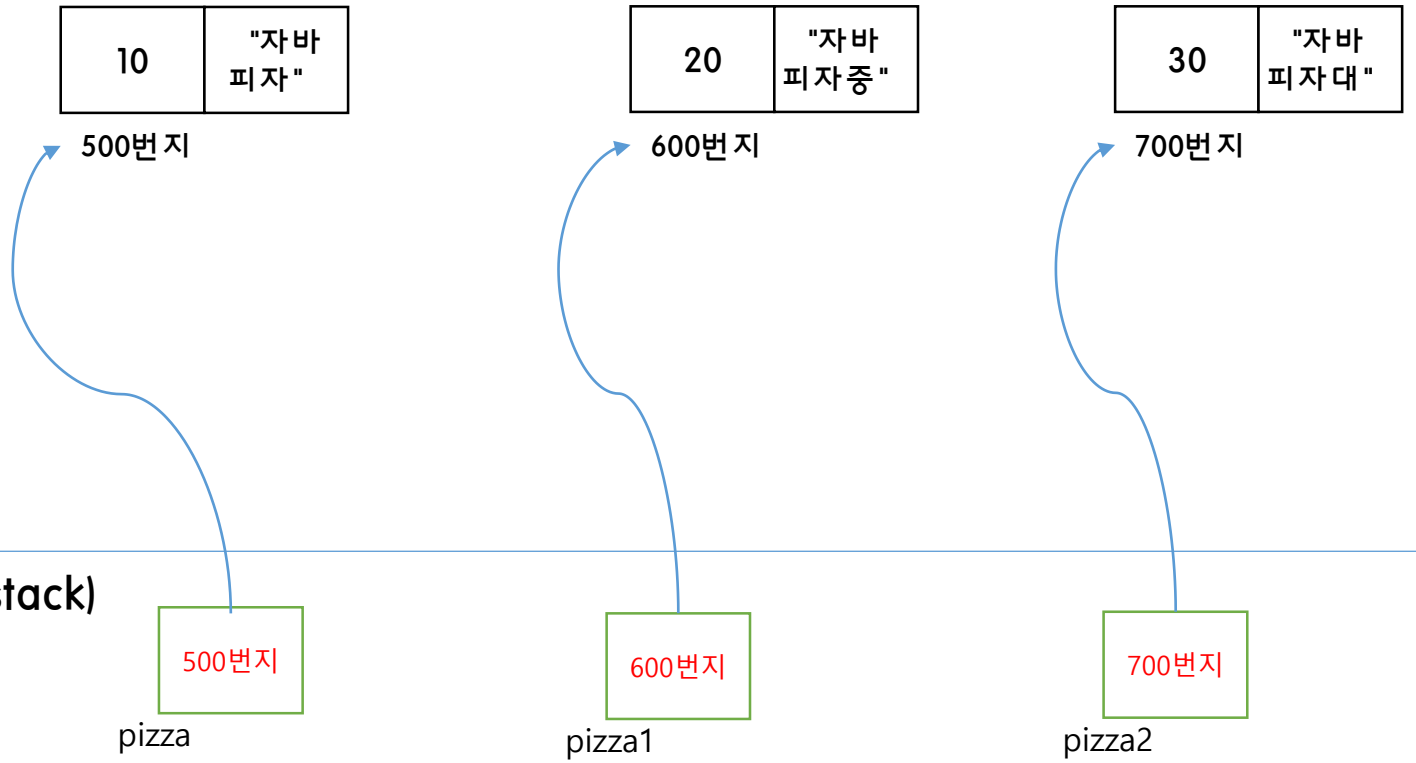
코드영역
데이타영역
(static)

```
double getArea(  ★  ){
    return 3.14*radius*radius
}
```

100번지

heap

```
Circle pizza = new Circle(10,"자바피자" );
Circle pizza1 = new Circle(20,"자바피자중" );
Circle pizza2 = new Circle(30,"자바피자대" );

System.out.println(   pizza.getArea(   )  );
System.out.println(   pizza1.getArea(   )  );
System.out.println(   pizza2.getArea(   )  );
```



this

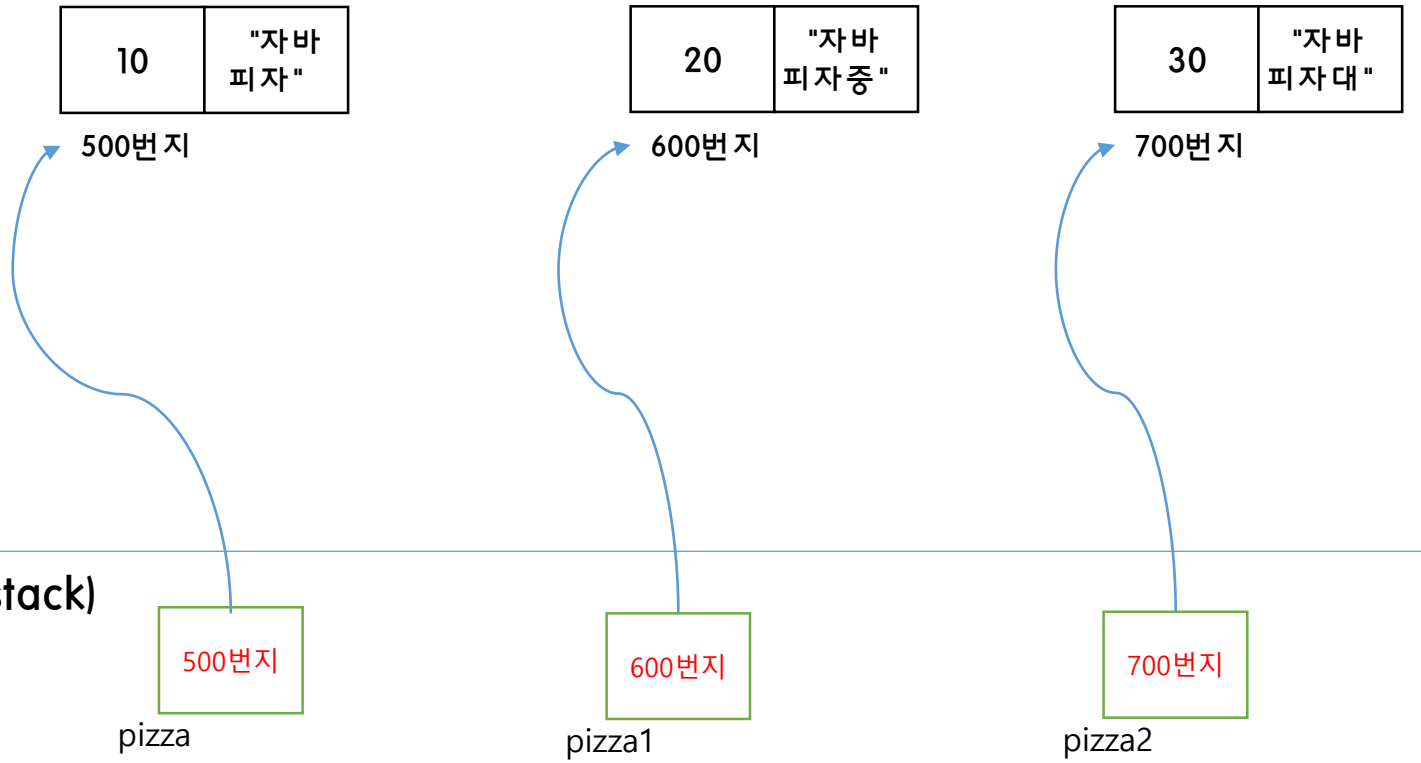
코드영역
데이타영역
(static)

```
double getArea(    Circle this       ){
    return 3.14*radius*radius
}
```

100번지

heap

```
Circle pizza = new Circle(10,"자바피자" );
Circle pizza1 = new Circle(20,"자바피자중" );
Circle pizza2 = new Circle(30,"자바피자대" );

System.out.println(    pizza.getArea(    pizza    )    );
System.out.println(    pizza1.getArea(    pizza1    )    );
System.out.println(    pizza2.getArea(    pizza2    )    );
```



캡슐화 도구

클래스 문법 제공 (데이터 + 매서드), this 제공됨
접근제어자 문법
생성자 문법

캡슐화로 얻는것

1. 구조화된 데이터의 종속적인 함수들의 변경 작업이 편리해짐
   :
   구조화된 데이터 와 그 데이터를 사용하는 함수를 묶기 때문에
   구조화된 데이터의 변경에 대한 유지 보수가 쉽다.
   (에러의 범위가 클래스 내에 집중)

2. 데이터보호 (정보은닉)
   (외부에서 의도, 실수등 데이터의 변경을 요청하는 작업이 완전히 배제됩니다.)

1. 접근제어자

접근제어자로 캡슐화 구현
정보은닉

private default public protected

2. 생성자(constructor)

목적: 멤버변수의 초기값 설정

규칙

1. 생성자의 이름은 클래스이름과 동일하다.(리턴형을 쓰지 않는다. 주의 할 것)
2. 생성자를 만들지 않으면 기본생성자를 하나 만들어 준다.

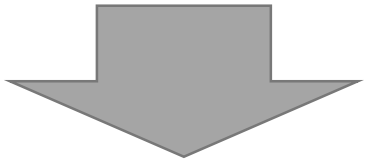
3. 생성자는 여러 개 작성할 수 있다.(생성자 오버로딩)

4. 생성자는 new를 통해 객체 생성시 한 번 만 호출된다.

5. 생성자에는 리턴타입을 지정할 수 없다.

6. 생성자를 하나라도 만들게 되면 디폴트(기본)생성자가 제공되지 않는다
   =>별도의 기본 생성자를 만드는 것이 좋다.

생성자로 값을 초기화 하는 의미 생각해 보자



데이타보호, 캡슐화(정보은닉)

객체가 생성될 때 단 한번 값 설정

객체의 값이 실수에 의해 변경되지 않도록 하겠다

```
class Score{
    private int kor;
    private int eng;
    private double avg;

    public Score() {  }
    public Score(int p_kor, int p_eng)
    {
          kor=p_kor;
          eng = p_eng;
    }
     public void input (int p_kor, int p_eng )
    {
          kor=p_kor;
          eng = p_eng;
    }
    public double getAvg( ) {
          int total  = kor+ eng;
          avg= total/2.0;
          return avg;
     }
}
```

생성자를 두는 이유?

```
class  Program{
       public static void mian(String[] args)
       {
         Score s = new Score(10,10);
         Score s1 = new Score();
         s1.input(10,10);

       }


}
```

생성자 오버로딩 (overloading)

매개변수가 다른 같은 이름의 생성자 작성하는 것

매서드 오버로딩 (overloading)

매개변수가 다른 같은 이름의 매서드 작성하는 것

객체지향의 캡슐화

데이터 변수는 감추겠다.
필요한 서비스(기능을)를 제공하겠다

변수는 데이터가 보호 되나요?

```
int su=10;
```

```
su=5;
```

객체는 데이터가 보호 되나요?

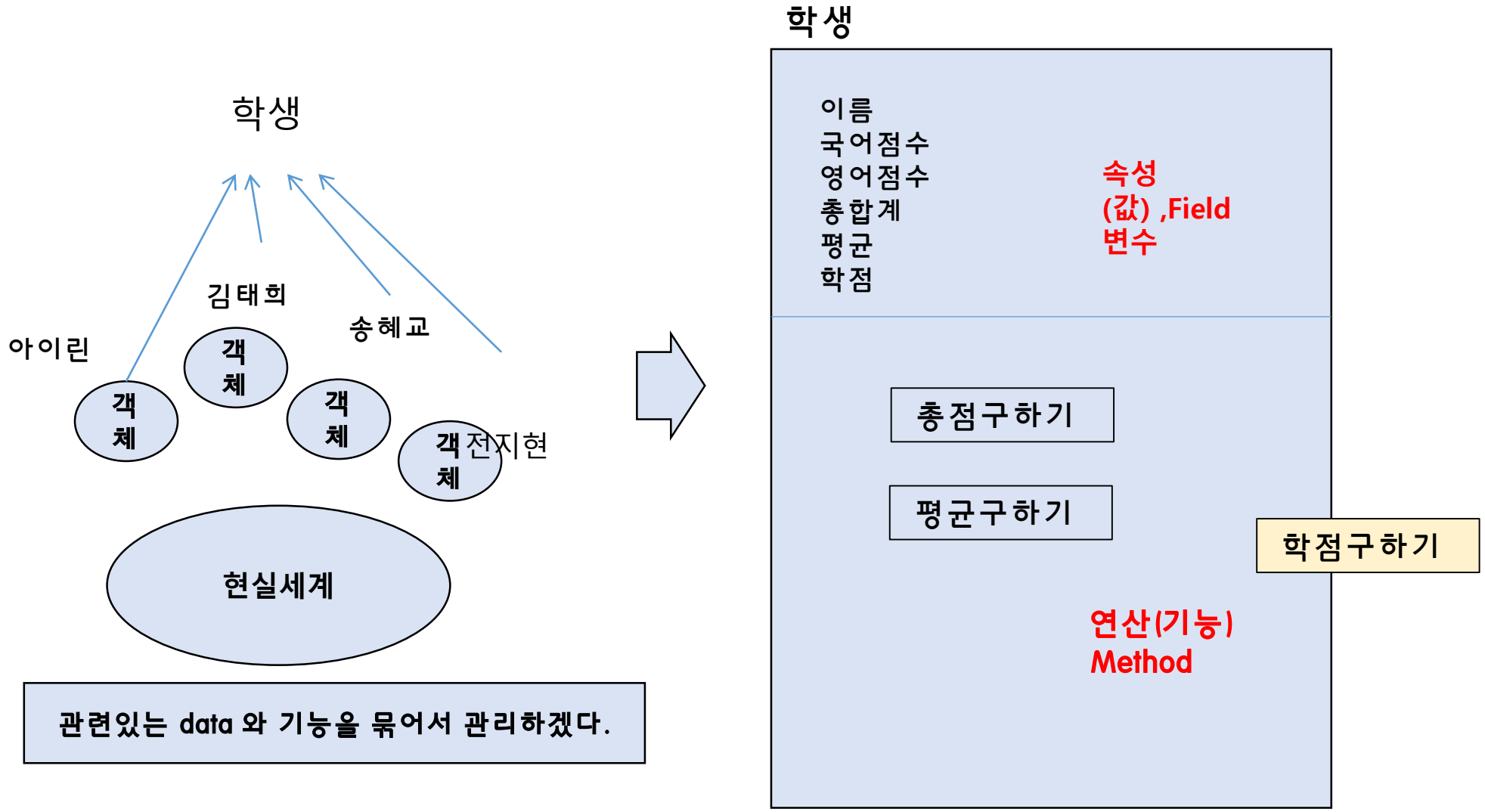
```
class A{
  private int su;

  public A(int p_su){
        su=p_a;

  public double squre(){
     return su*su ;
  }
}
```

```
A a = new A(5);
a.su = 10 ;
a.squre();
```

생성자가 한 번만 호출되는이유?

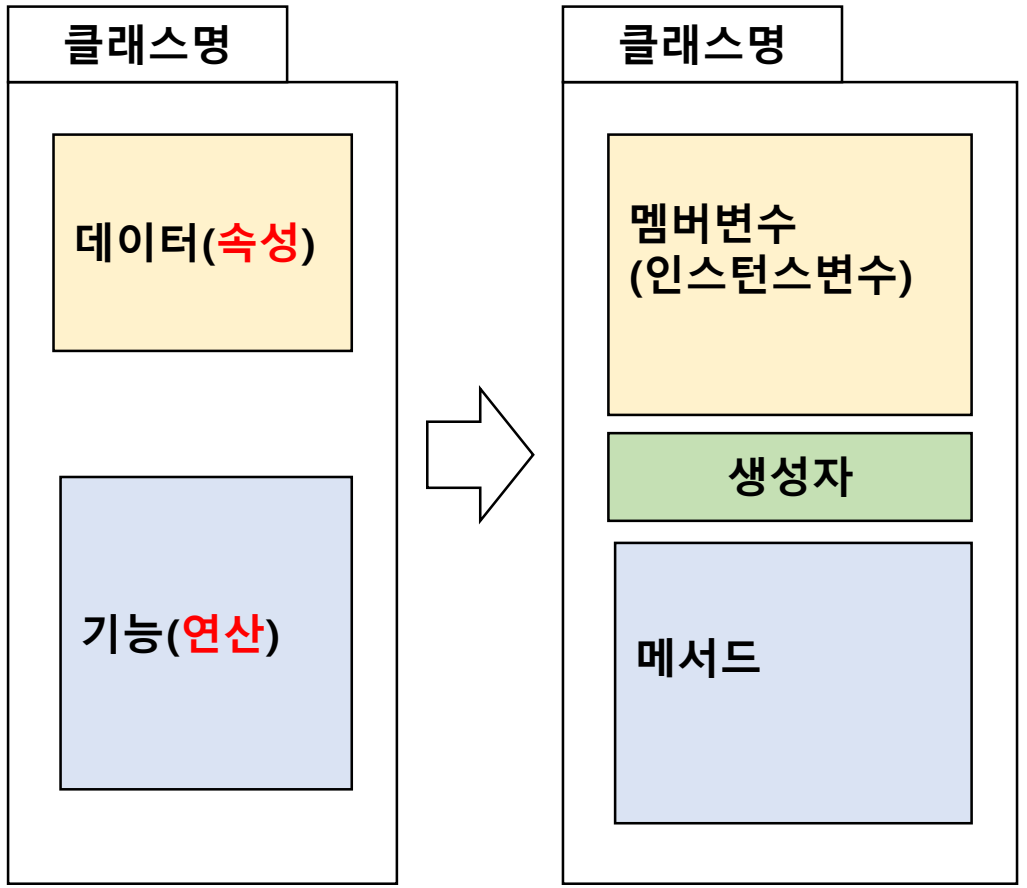
알겠나요?

성적처리를 위한 객체 모델링



<실세계에 존재하거나 생각할 수 있는 것을 **객체라고 한다.**>

독립적으로 존재하는 유,무형의 실체이다


클래스명
데이터(속성)
기능(연산)
클래스명
멤버변수
(인스턴스변수)
생성자
메서드

객체의 내부(데이터+기능)을 감춰진 채로 외부(공개된) 단순 inferface을 통해 객체를 이용할 수 있도록 한 것(캡슐화)

=>
접근지정자, 생성자를 이용
캡슐화 구현