객체지향프로그램밍이란

클래스를 이용해서
객체로 코딩한다

프로그래밍 기법

절차지향 프로그래밍
구조적 프로그래밍
객체지향 프로그래밍

간단한 예제  를 통해
프로그램의 기법을 알아 봅시다.

학생정보출력
//변수선언

학번
이름
주소

//값저장
학번="S100"
이름="홍길동"
주소="서울시 노원구"

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

학생정보출력

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);
}

```
class 학생{
    학번
    이름
    주소
}


 입력하기 ( 학생   학생1){
  학생1.학번= "S100";
  학생1.이름="홍길동";
  학생1.주소 ="서울시 노원구";
}


 출력하기( 학생 학생1){
   System.out.println(학생1.학번);
   System.out.println(학생1.이름);
   System.out.println(학생1.주소);
}
```

```
//프로그램시작 , 약식표현임
main(){

    학생 학생1= new 학생();

    입력하기( 학생1);
    출력하기( 학생1);

}
```

학생정보 출력

```
class 학생{
        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(){

    학생  학생1 = new 학생();
    학생1.입력하기( "S100", "홍길동","서울시 노원구");
    학생1.출력하기();
}
```

클래스 무엇?

객체가 무엇?

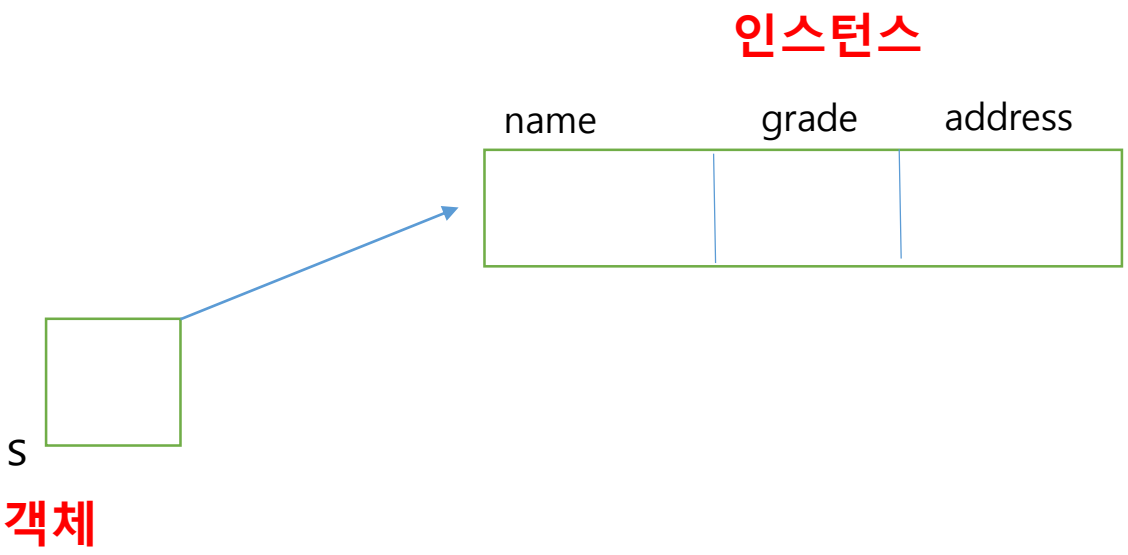
★클래스 : 사용자정의 자료형 (구조화된 데이터 저장)이다.
객체 : 객체참조형변수 이다.
인스턴스 : 객체참조변수가 참조하는 실체를 말한다.

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

클래스

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

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



용어정리

자료형

변수

class

객체

new

인스턴스

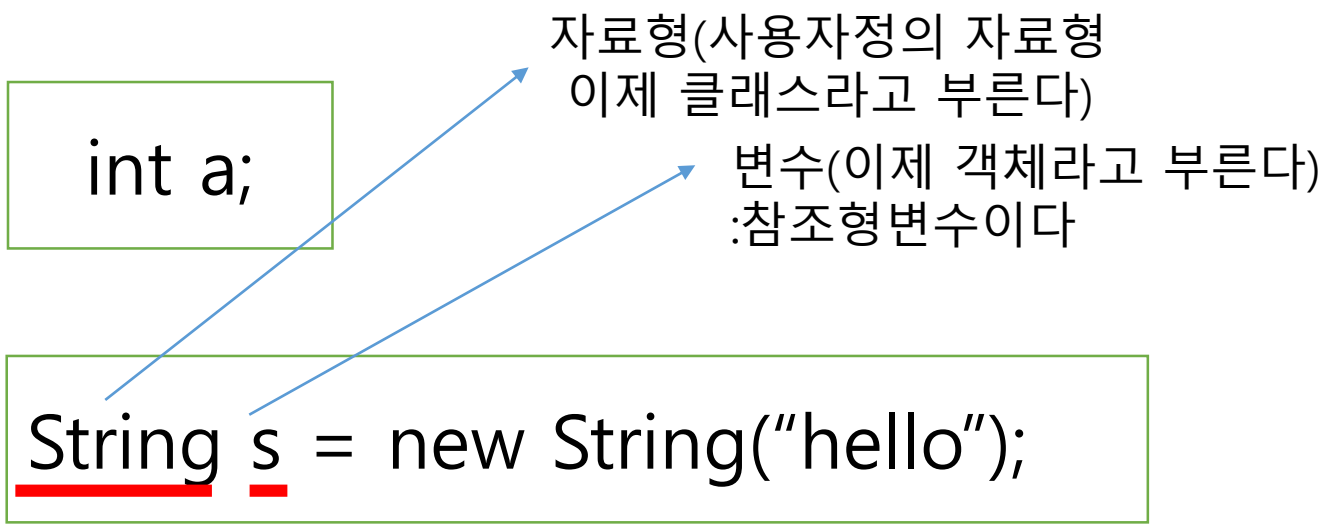
가끔 혼용해서 쓰기도 합니다.

자료형

클래스

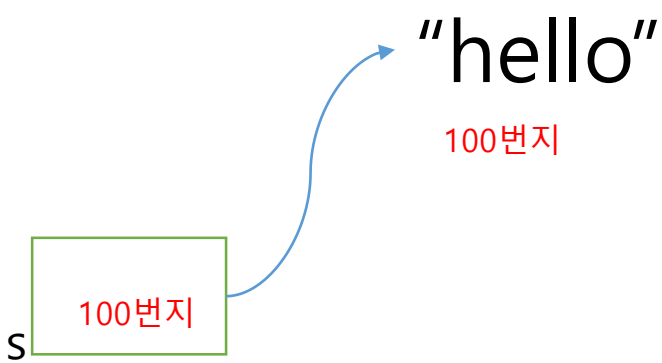
변수

객체



```
int a=10;
```


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



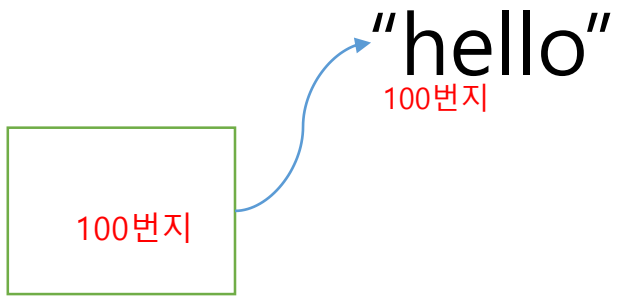
int a;
int a= 78;

변수 선언과 초기화

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

객체 생성과 초기화

a



s

변수가 메모리를 쓰는 상태를 메모리에 태어났다라는 의미를
부여함. 이제 변수(객체가) 메모리에 생성 되었다 라고 부른다.

```
int a;
a= 78;
```

```
int a= 78;
```

변수 선언과 초기화
선언과 초기화가 구분되었다.

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

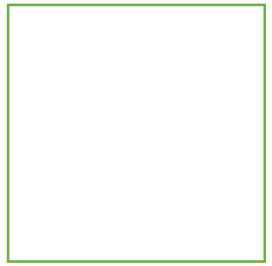
객체 생성과 초기화
초기화는 반드시 객체 생성 시에만 가능하다

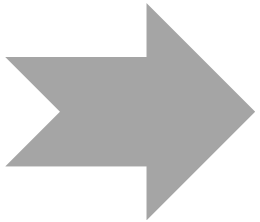
자료형은 메모리에 변수라는
이름을 갖고 태어난다.

생성된다.

메모리라는 세계

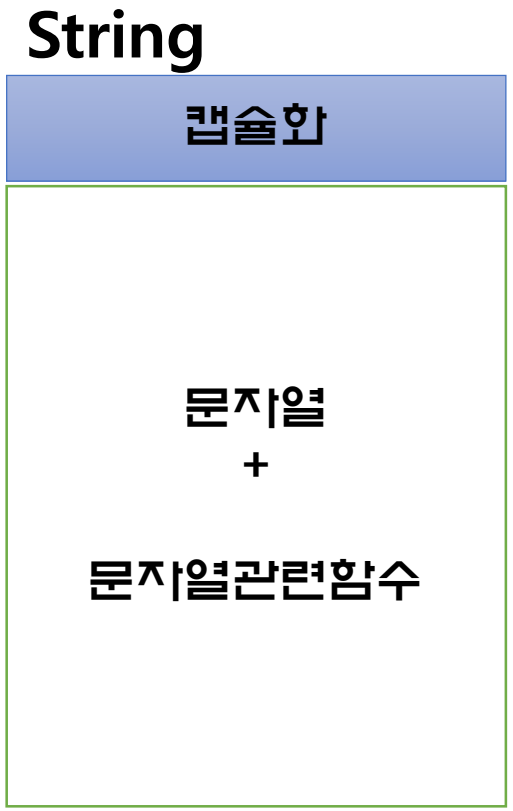
객체가 태어난다

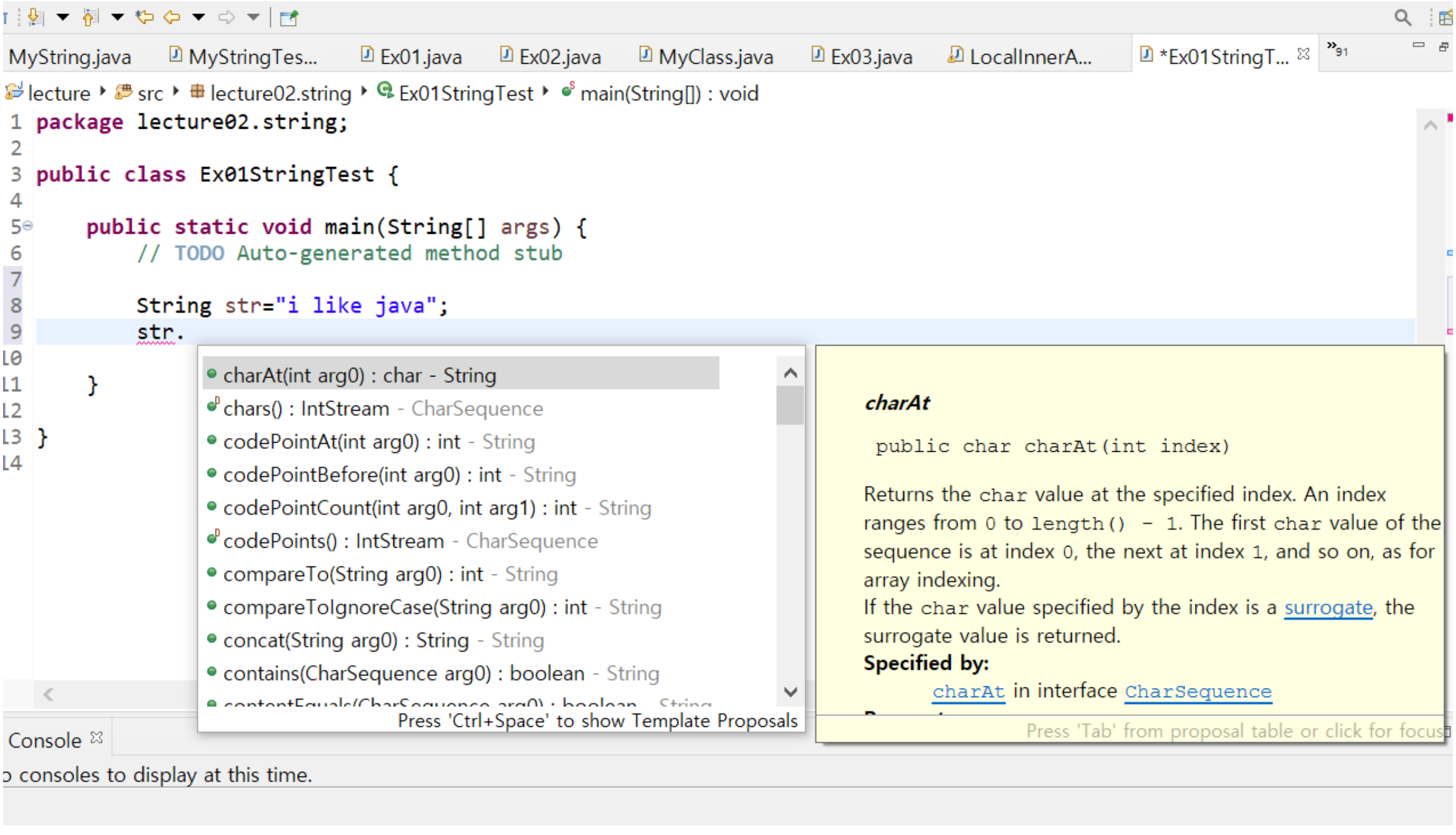




생성되었다

관련있는 데이터와 함수를 묶어서 하나의자료형으로 제공한다.




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

```
class Rectangle{

    int width;
    int height;
    public int getArea()
    {
        return width * height;
     }
}
```

```
public class RectApp{

     public static void main(String[] args)
     {
            Rectangle r = new Rectangle();
            r.width=5;
            r.height=3;
            System.out.println(   r.getArea());
    }

}
```

```
public class Rectangle{

    int width;
    int height;
    public Rectangle(int w, int h)
    {
            width= w;
            height= h;
    }
    public int getArea()
    {
        return width * height;
     }
}
```

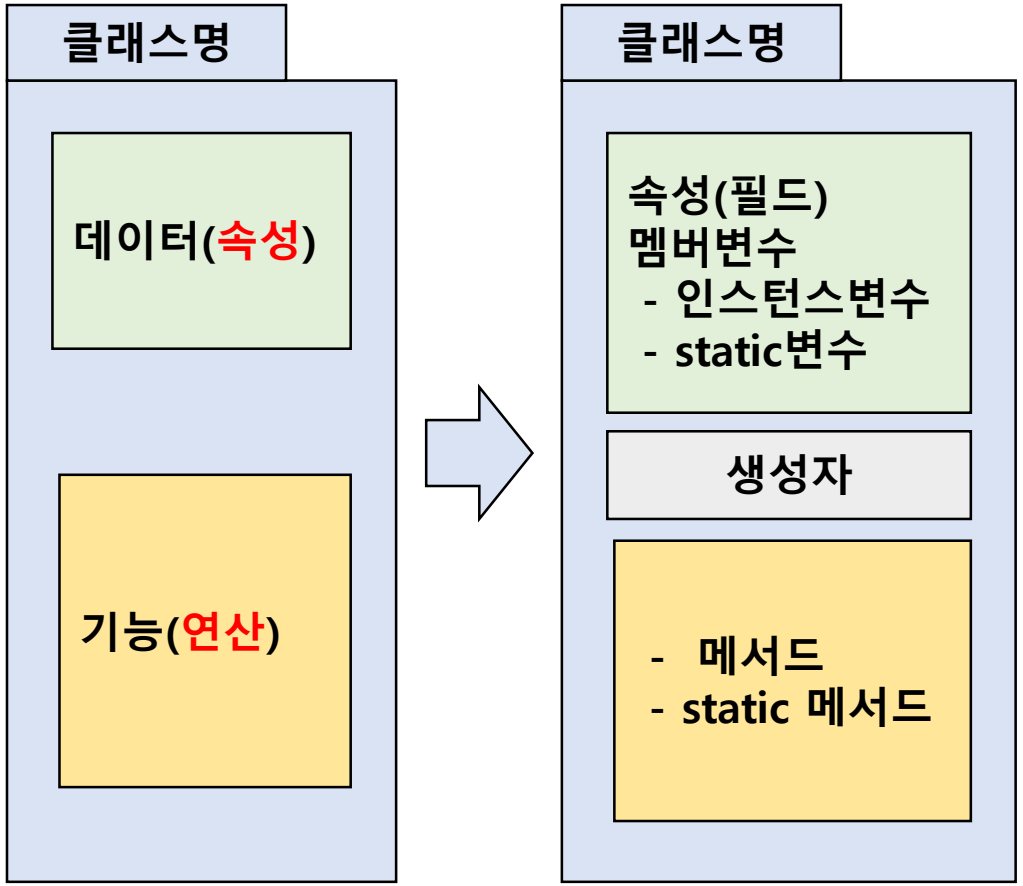
```
public class RectApp{

     public static void main(String[ ] args)
     {
            Rectangle r = new Rectangle(5,3);
            System.out.println(   r.getArea() );
    }
}
```

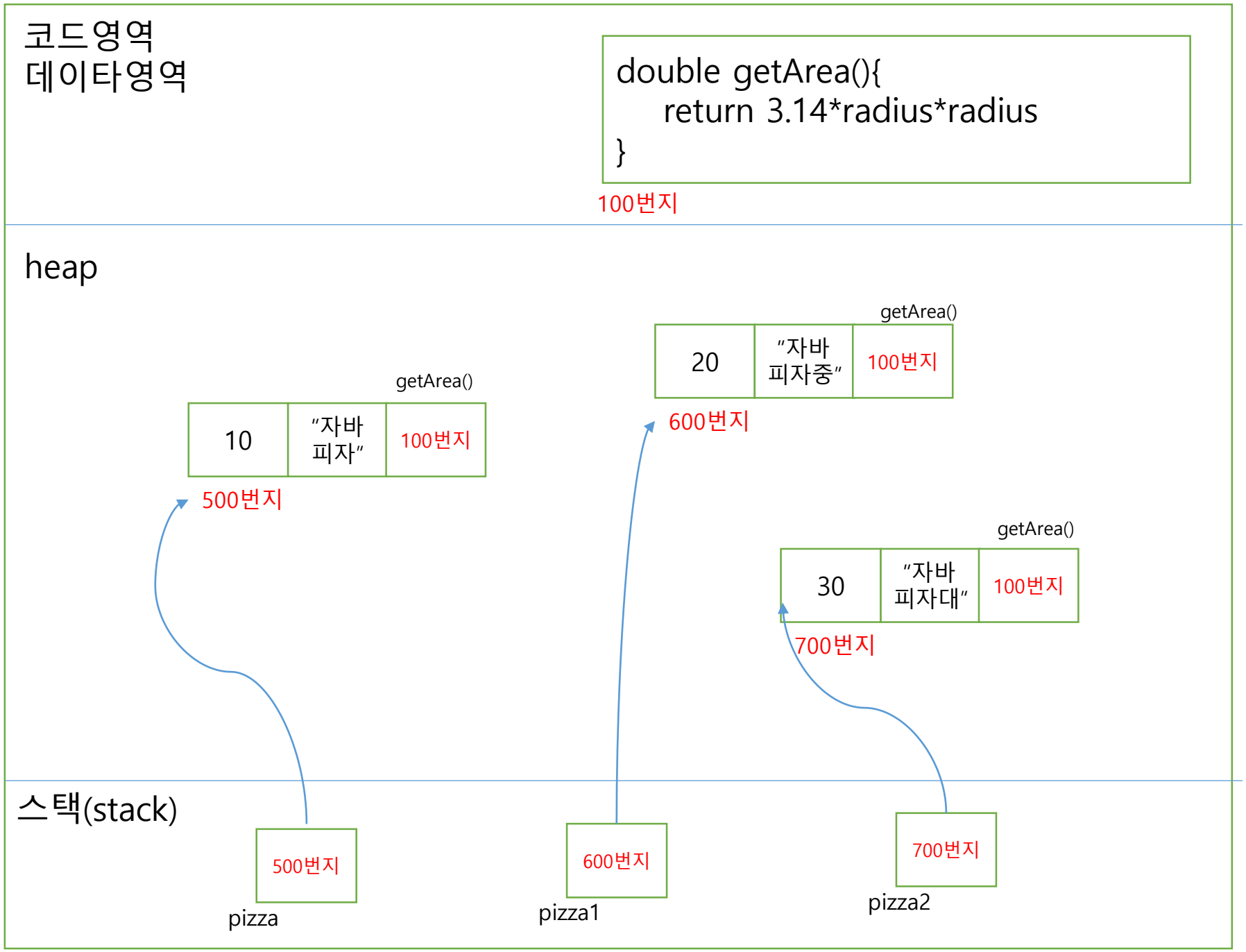
객체의 멤버를 사용할때
.(점)을 사용한다.

```
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()   );


}
```

```
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()   );

        }
}
```


클래스명
데이터(속성)
기능(연산)
클래스명
속성(필드)
멤버변수
- 인스턴스변수
- static변수
생성자
- 메서드
- static 메서드


코드영역
데이타영역
double getArea(){
return 3.14*radius*radius
}
100번지
heap
getArea()
10
"자바 피자"
100번지
500번지
getArea()
20
"자바 피자중"
100번지
600번지
getArea()
30
"자바 피자대"
100번지
700번지
스택(stack)
500번지
pizza
600번지
pizza1
700번지
pizza2

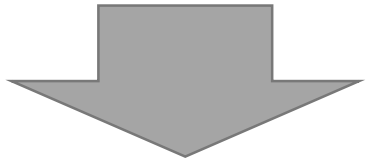
생성자에 대해

기본생성자 : 생성자를 만들지 않으면 하나 만들어 준다.
하는 일은 없음

외부값으로 클래스내부의 변수값을 초기화 원한다면 외부값을 받는
생성자를 준비해야 한다.
다양한 형태의 생성자를 준비할 수 있다.

주의사항: 생성자를 하나라도 만들면 기본생성자는 제공되지 않는다.
기본생성자는 작성하는 것이 좋다.

생성자로 값을 초기화 하는 의미 생각해 보자



데이타보호, 캡슐화(정보은닉)

객체가 생성될 때 단 한번 값 설정

객체의 속성값을 변화하지 못하게 하자

생성자

목적: 멤버변수의 초기값 설정

1. 생성자의 이름은 클래스이름과 동일하다.(리턴형을 쓰지 않는다. 주의 할 것)
2. 생성자를 만들지 않으면 기본생성자를 하나 만들어 준다.
3. 반드시 public으로 작성한다. (주의)
4. 생성자는 여러 개 작성할 수 있다.(오버로딩)
5. 생성자는 new를 통해 객체 생성시 한 번 만 호출된다.
6. 생성자에는 리턴타입을 지정할 수 없다.
7. 생성자를 하나라도 만들게 되면 디폴트(기본)생성자가 제공되지 않는다 =>기본 생성자를 만드는 것이 좋다.

생성자를 두는 이유?

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

    public Score() {  }
    public Score(int p_kor, int p_eng)
    {
        kor=p_kor;
        eng = p_eng;
    }
     public void init(int p_kor, int p_eng )
    {
        kor=p_kor;
        eng = p_eng;
    }
```

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

}


}
```

생성자예제

```
public class Construct {

public static void main(String[] args) {
        Circle c  = new Circle();
        Circle c1 = new Circle(10, "자바피자");
    }
}
```

접근제어자로 캡슐화 구현

정보은닉

캡슐화로 얻는것

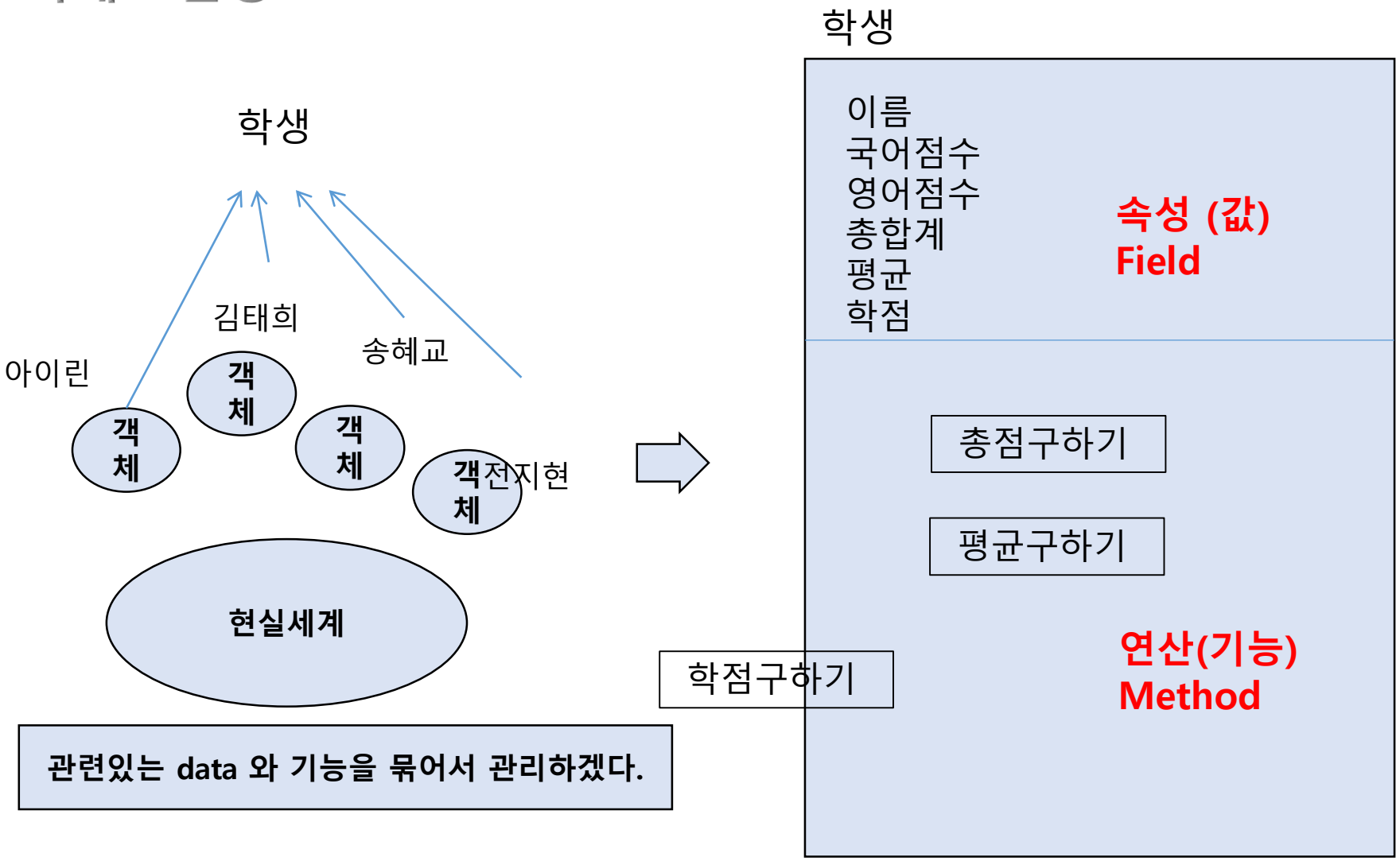
1. 구조화된 데이터의 종속적인 함수들의 변경 작업이 편리해짐
   :
   구조화된 데이터 와 그 데이터를 사용하는 함수를 묶기 때문에
   구조화된 데이터의 변경에 대한 유지 보수가 쉽다.
   (에러의 범위가 클래스 내에 집중)

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

객체지향의 캡슐화

데이터 변수는 감추겠다.
필요한 서비스를 요청하라

객체모델링



<실세계에 존재하거나 생각할 수 있는 것을 **객체라고 한다.>**
독립적으로 존재하는 유,무형의 실체이다

```
int su=10;
```

```
su=5;
```

변수는 데이터가 보호 되나요?

```
class A{
  private int su;

  public A(int p_su){
        su=p_a;

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

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

생성자가 한 번만 호출되는이유?

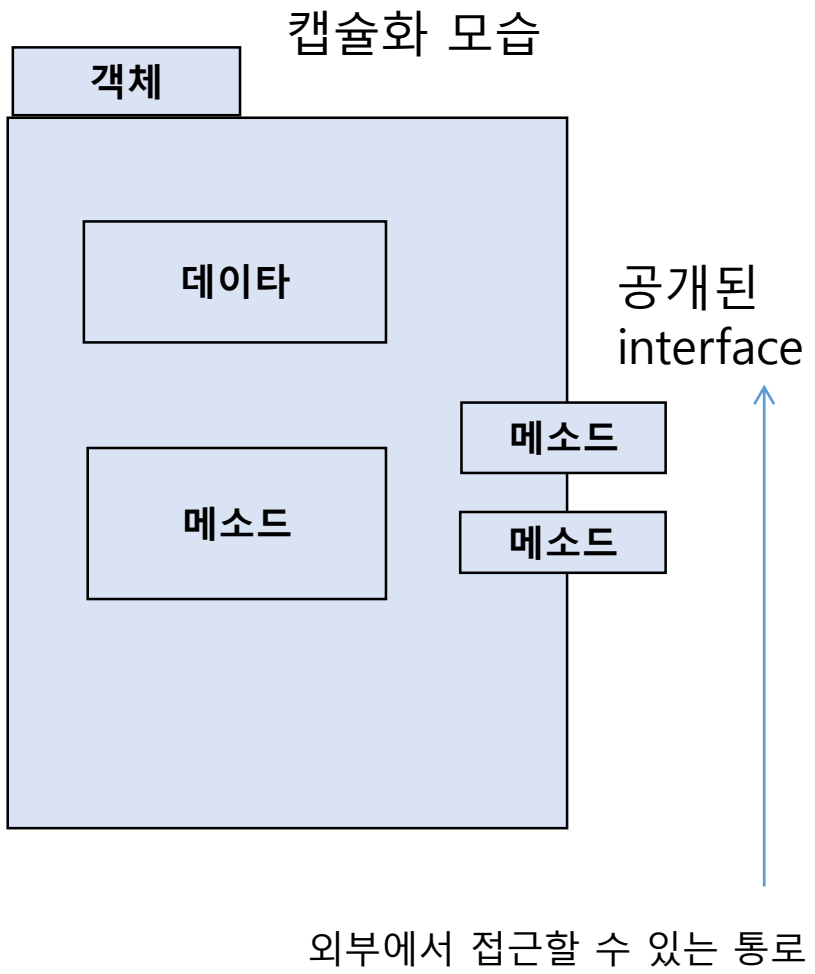
알겠나요?

객체가 생성될때 초기값으로
설정된 것을
함부로 바꿀 수 없게 하기
위함입니다

데이터의 보호

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

=>접근지정자를 통해서
캡슐화 구현



클래스를 만들 때는 set과 get을 지양하고
필요한 서비스를 요청하라

데이터를 요구하지 말고 원하는 것을 말하라는 뜻

this

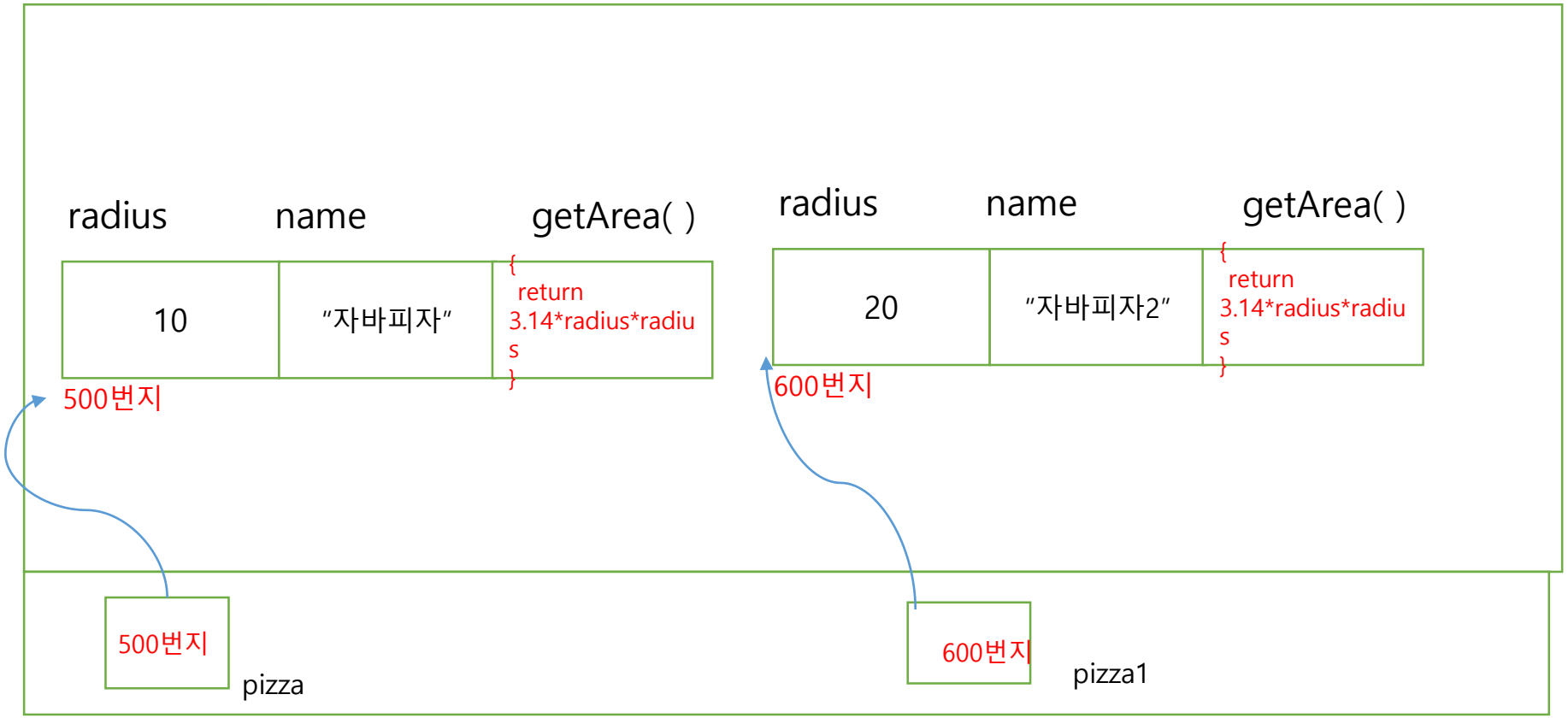
this 래퍼런스

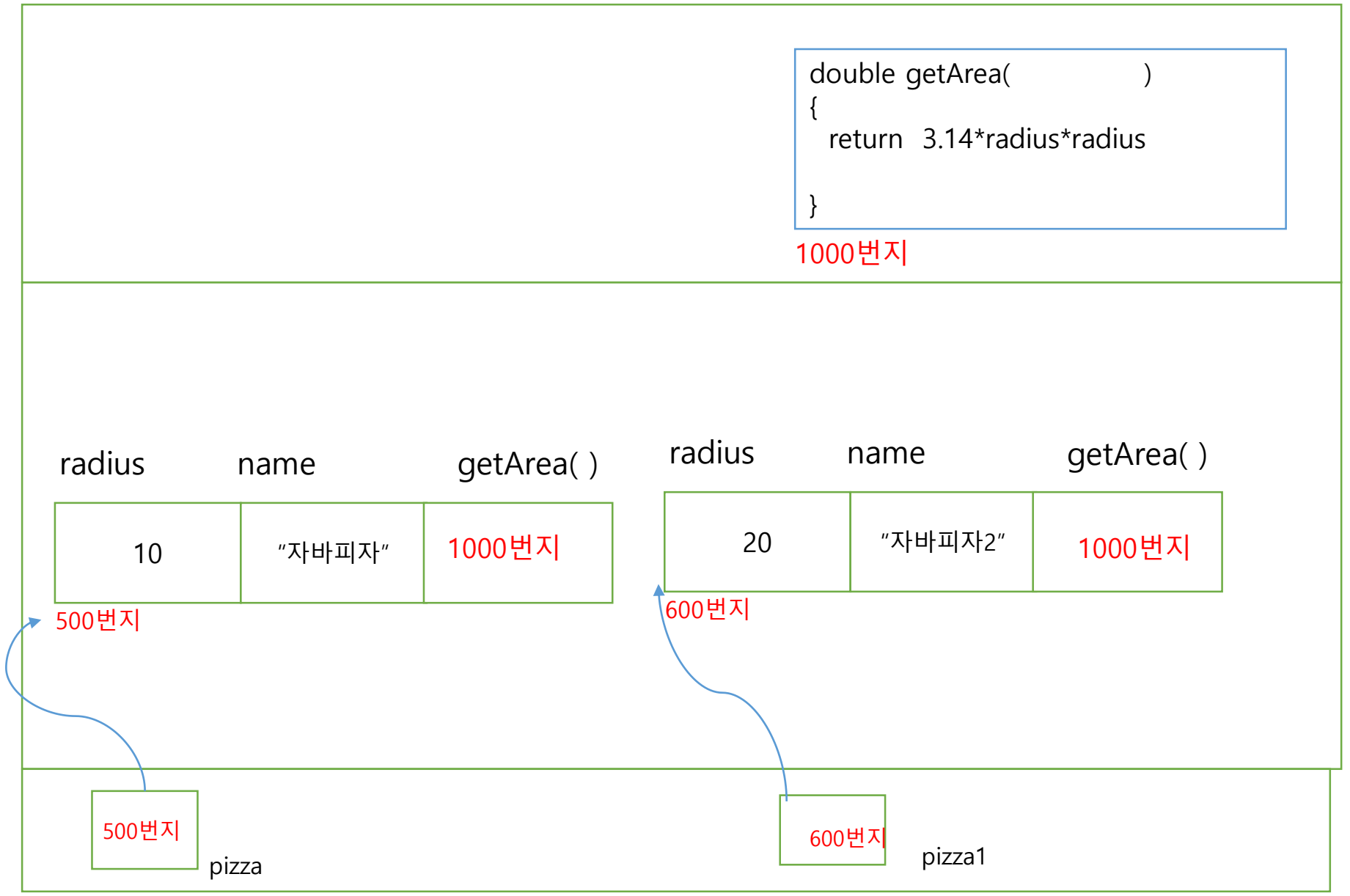
this 는 참조형변수 입니다

자신을 가르키는 래퍼런스 변수이다

```
Circle pizza = new Circle( 10, "자바피자");
Circle pizza1 = new Circle( 20, "자바피자2");
```

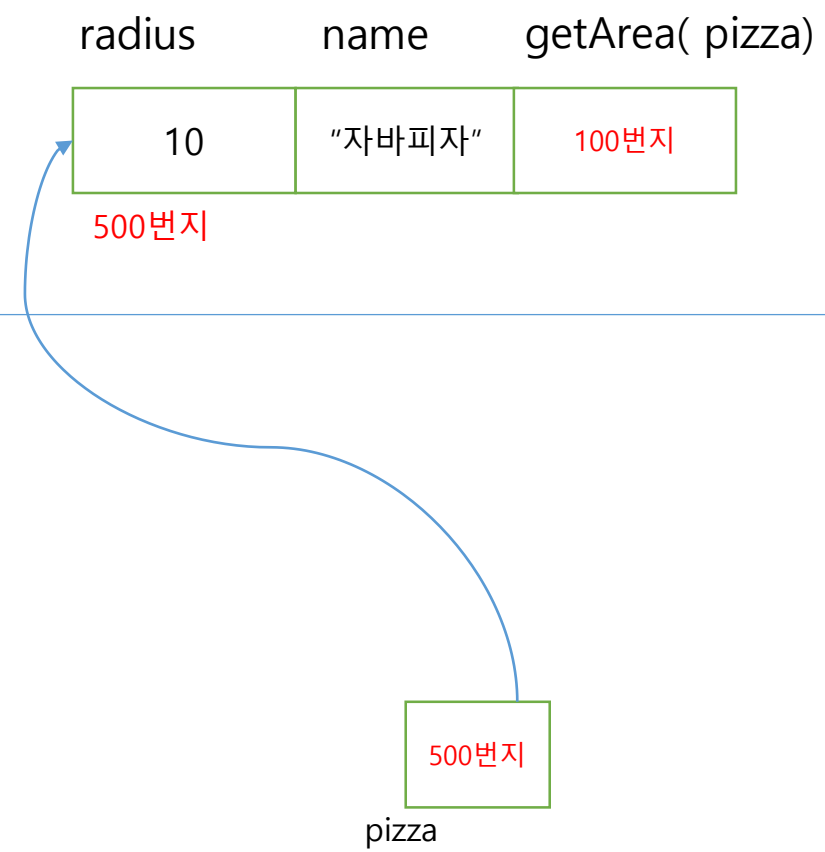
메모리에 할당된 모습




double getArea( )
{
return 3.14*radius*radius
}
1000번지
radius
name
getArea( )
10
"자바피자"
1000번지
500번지
radius
name
getArea( )
20
"자바피자2"
1000번지
600번지
500번지
pizza
600번지
pizza1

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

100번지




```
Circle pizza = new Circle(10,"자바피자");
pizza.getArea(pizza );
```

this 참조형변수

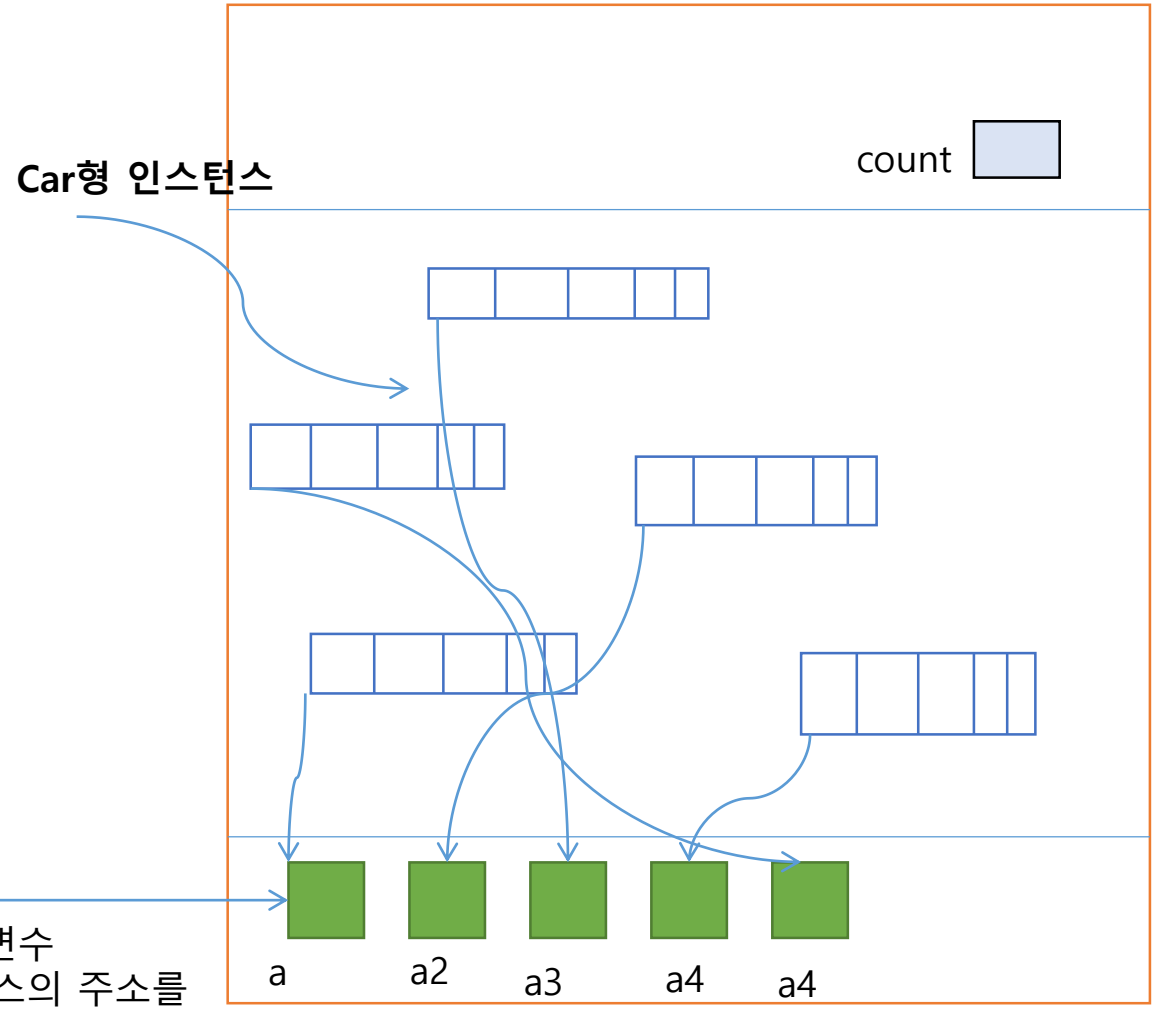
왜 this로 지었을까요 ?

Static 멤버변수

Static 멤버

Static 멤버메서드

```
Car a = new Car();
Car a1 = new Car();
Car a2 = new Car();
Car a3 = new Car();
Car a4 = new Car();
Car a5 = new Car();
Car a6 = new Car();
```



참조형변수
:인스턴스의 주소를 담고 있음
(리모콘 변수)

static

프로그램실행시
main 보다 먼저 실행됨

클래스 멤버
인스턴스 멤버

```
public static void main(String[] args)
{
    이제 왜 main메서드가 static인지 알겠나요?
객체가 생성되기 전에 main은 호출되어야 합니다.
        그래서 main은 static인 거죠
}
```

static은 객체 생성과 상관없이 사용할 수 있다.
Static 메서드에서 객체의 일반변수 , 즉 static이 아닌 변수를 접근 할 수 있다.
너무 당연하다.

우리가 클래스의 멤버메서드를 사용하려면 객체 생성
new 연산자를 통해 객체생성을 한 후 메서드를 호출할 수 있습니다.
그러나 static이 붙은 메서드는 클래스이름으로 바로 메서드를 호출할 수 있습니다.
어차피 메모리에 올라와 져 있으니까요
Math.random() 기억나시나요?
그렇다면 random()은 static으로 만들어져 있는 메서드라는 걸 알 수 있습니다.

static

클래스 내에서 멤버변수와 관련한 작업을 하지 않는 멤버함수를 만들때
사용한다. (예전의 함수의 의미가 강함)

어떤 경우에 static 키워드가 붙는가?

클래스의 멤버변수와 관련된 작업은 메서드(인스턴스 메서드라고 부른다.)
클래스의 멤버변수와 관련이 없는 작업은 static 메서드로 만든다.
(예전의 함수라고 생각하면 됨)

static 변수를 두는 이유?
클래스에서 공통으로 사용하는 변수로 사용할 목적

static메서드
 주로 일반멤버변수를 사용하지 않는다면 static 고려
 static 멤버 변수를 사용하는 메서드를 만들 때

java.lang.Math

```
public class Math{
        public static int abs(int a){}
        public static int max(int a, int b){}
        public static double random(){}

}
```

static 메서드

```
Class MayMath{
        public static int abs(int a) {return a>0?a:-a); }
        public static int max(int a, int b) { return (a>b)?a:b;}
        public static int min(int a, int b)  { return (a>b)?b:a; }
}


public class Ex01{
    public static void main( String[] args)
   {
        System.out.println(MayMath.abs(-5) );
        System.out.println(MayMath.max(10,8) );
        System.out.println(MayMath.min(-3, -8) );

    }
}
```

```
public class Hero {
                                        //필드 (멤버변수)
    String name;
    int hp;
    static int count;
                                        //생성자
    public Hero(String a,  int b)
    {
      count++;
      name=a;
      hp=b;
    }
                                        //매서드
    public void punch()
    {
      System.out.println(name + "펀치  =>" + hp);
    }
    public static void main(String[] args) {
      //실습 :여러 히어로객체를 만들고 객체의 수를 카운트 하자
         Hero h1= new Hero("아서스",200);
         Hero h2= new Hero("레오닉",180);
         Hero h3= new Hero("제이나",170);
     //생성된 객체의 수
          System.out.println(Hero.count);
     }
}
```

```
public class MyCalc
{
 //필드 (멤버변수)
  int M_a;
  int M_b;

 //생성자
 public MyMath(int a, int b)
 {
  M_a = a;
  M_b = b;

 }
 //인스턴스 메서드
  public int  add()
  {
    return M_a + M_b;
  }
  //클래스 메서드
  static int add(int a, int b)
  {
   return a+b;
  }
}
```

static 메소드 : 클래스 메소드

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

    MyMath m = new MyCalc(5,3);
    System.out.println( m.add());

    System.out.println(MyCalc.add(5, 11));
 }

}
```

Random

java.util
Random
double nextDouble()

java.lang
Math
static double random()

```
import java.util.Random;
public class RandomTest {

public static void main(String[] args) {

    Random r = new Random();   //객체생성 후 사용
    for(int i=0; i<=20 ; i++)
    {
       System.out.println(r.nextDouble());
     }

    System.out.println("------------------------------");
    for(int i=0; i<=20 ; i++)
    {
       System.out.println(Math.random());
    }
   }
}
```