

5장부터 자바의 객체 지향 개념에 대해 학습한다. 자바는 객체 지향 언어이므로 자바에서 제공 하는 모든 기능은 5~7장에서 나오는 객체 지향 개념으로 만들어져 제공된다. 그러므로 자바 프로그래밍을 잘하려면 객체 지향 개념에 대해 이해해야 한다.

자바는 앞의 3, 4장에서 배운 프로그래밍(절차 지향 프로그래밍)을 잘해야만 지금부터 배우는 객체 지향 개념을 쉽게 이해할 수 있다. 그리고 객체 지향 개념을 잘 이해해야만 자바에서 제공하는 여러 가지 기능(API)을 사용하여 프로그래밍을 할 수 있다.

클래스를 처음 접하는 사람들은 이 객체 지향 개념이 일종의 장벽으로 느껴질 수 있다. 하지만 학습을 하다 보면 객체 지향 개념을 이용하여 프로그래밍을 하는 것이 훨씬 쉽고 효율적이라는 것을 알게 될 것이다. 따라서 인내심을 가지고 꾸준히 학습하기 바란다.

- 1.절차 지향 프로그래밍 및 객체 지향 프로그래밍의 정의와 차이점
- 2.자바에서의 객체 지향 프로그래밍 과정
- 3.생성자(Constructor)
- 4.오버로딩(overloading) 생성자 및 메서드

- 5.패키지(package)
- 6.import
- 7.기본형 변수와 참조형 변수
- 8.렌터카 예약 시스템을 클래스로 구현하기
- 9.클래스 실습

- 절차 지향 프로그래밍의 정의

프로그래밍을 정해진 순서, 즉 절차에 따라 프로그래밍하는 방식

- 절차 지향 프로그래밍의 특징

- 일을 처리하는 순서와 과정이 중요하다.
- 순서, 과정이 달라지면 새로운 작업 모델이 필요하다.(bottom-up방식)
- 컴퓨터의 작업 방식을 프로그래밍에 적용한 것이다.
- 재사용성이 매우 낮다.



- 객체 지향 프로그래밍의 정의

'절차(순서) 중심'이 아닌 '**객체 중심**'으로 프로그래밍하는 방식

- 객체 지향 프로그래밍의 특징

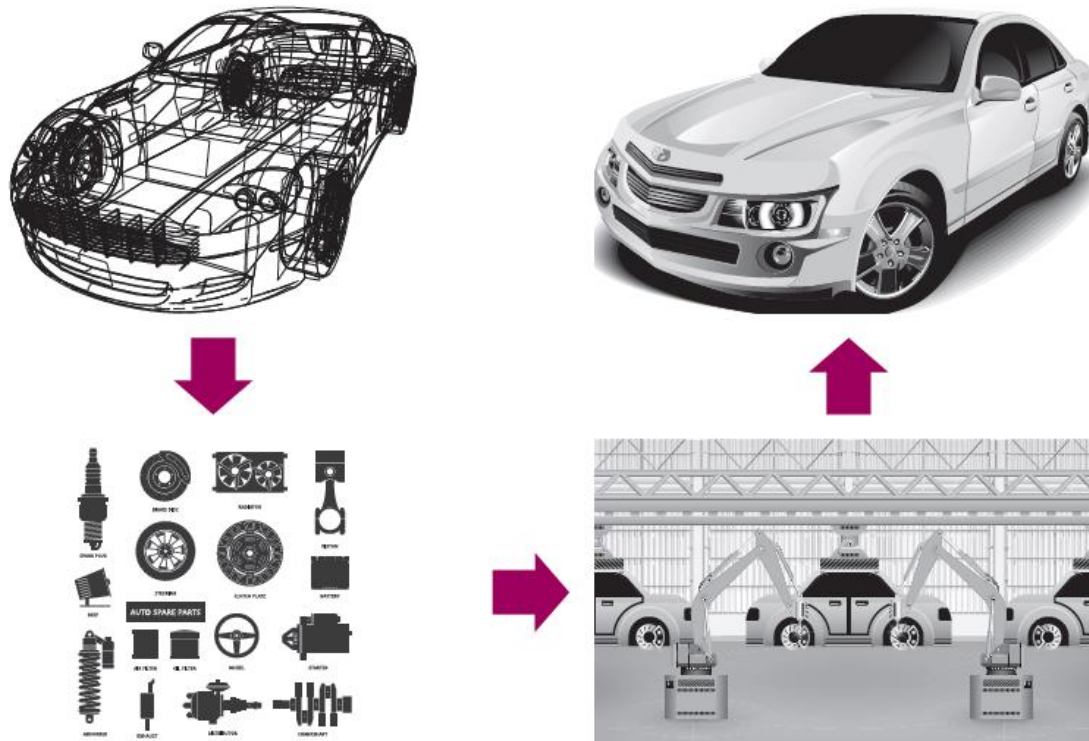
- 사람이 행동하고 생각하는 방식으로 프로그래밍을 한다.
- **순서, 과정이 중요하지 않다.**
- **이해하기 쉽고, 모듈화(조립화)하기가 쉽다.**
- **재사용성이 높고, 유지보수가 용이하다.**
- **설계 중심 언어(top-down 방식)이다.**



- 객체 지향 프로그래밍 방법

객체 지향 프로그래밍 과정은 현실에서 실제 물건(제품)을 만드는 과정과 유사하다.

- 자동차 생산 과정



■ 객체 지향 프로그래밍 과정

- 현실 세계에서 **사람(주체)**이 **바라본 관점**으로 프로그래밍하는 것
- 용어 설명
 - 주체(subject) : 일반 사용자, 사람
 - 객체(object) : 주체가 바라본 대상(사람, 사물, 개념 등)

■ 렌트카 회사를 통한 객체 지향 프로그래밍 과정

-렌트카 회사의 일반적인 업무

렌터카 회사에서는 일단 렌터카를 회사 주차장에 주차해 놓고 **직원이** 관리한다. 그리고 렌터카를 사용하려고 하는 **고객**이 회사에 방문하여 원하는 차를 직원에게 요구하면 직원은 그 **렌터카**로 고객을 안내하여 함께 살펴본 후에, 고객이 마음에 들어 하면 예약을 하고 예약한 날에 해당 렌터카를 대여해준다. 그러면 고객은 그 렌터카를 예약된 기간 동안 사용한 후에 반납 예정일에 맞춰 렌터카 회사에 반납한다.

- 객체 지향 프로그래밍으로 렌트카 예약 프로그램 개발 과정
 - 일반인(subject)의 렌트카 회사 업무 분석



일반인(subject)

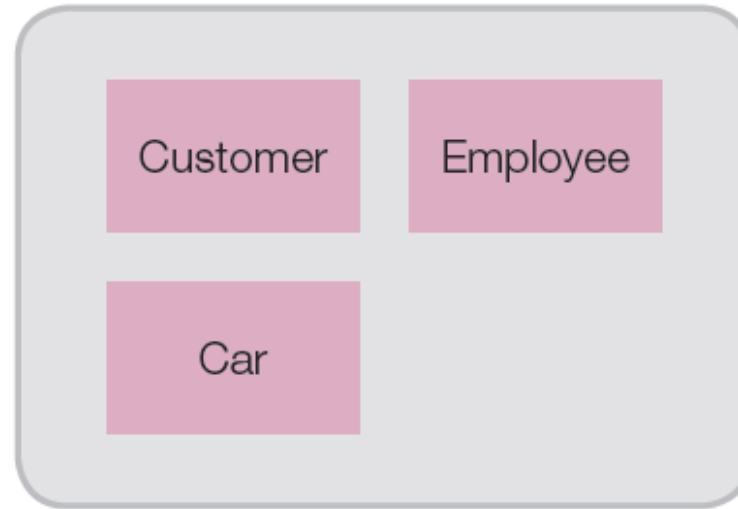


렌트카 회사에 관계되는 여러가지 객체

- 객체 지향 프로그래밍으로 렌트카 예약 프로그램 개발 과정
 - 렌터카 회사 업무를 분석한 후 객체 지향 프로그래밍 과정



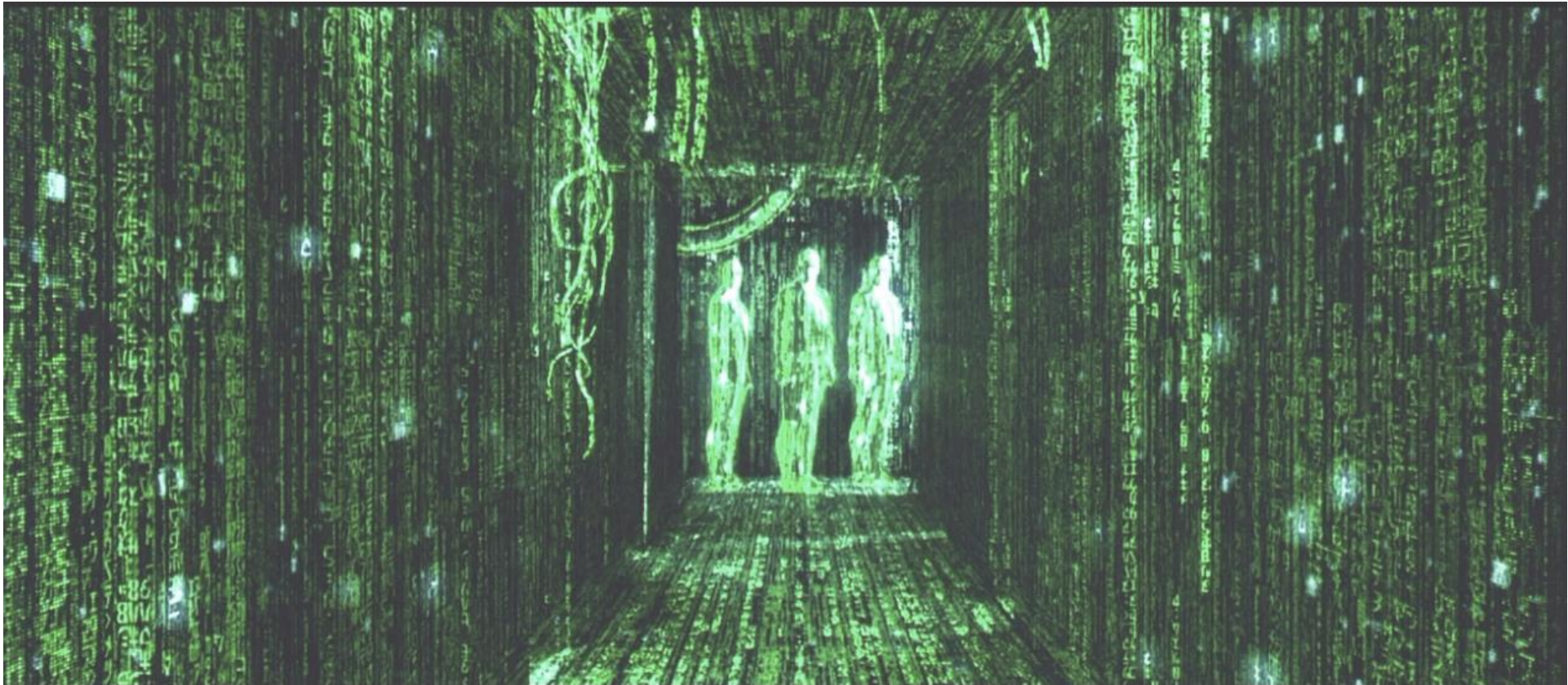
개발자



자바 클래스

- 객체 지향 프로그래밍으로 만들어진 프로그램

객체 지향 프로그램은 **현실의 객체를 클래스로 코드화 시켜서 컴퓨터 시스템에서 구현한 가상현실**이다.



- 자바 객체 지향 프로그래밍 과정

아이디어 도입

----- 구현될 프로그램에 대한 요구를 나타낸다.

아이디어 분석

----- 주체의 관점에서 아이디어를 분석한다.

객체 추출

----- 주체의 관점에서 객체를 추출한 후 속성과 동작을 나열한다.

객체 모델링

----- 개발자의 관점에서 객체의 속성과 동작을 정제한다.

클래스 변환

----- 객체를 클래스로 변환한다.

인스턴스 생성 및 사용

----- 메모리에 인스턴스로 생성한 후 사용한다.

1.객체 추출

-객체의 속성과 동작(기능)의 정의

- 속성

- 객체가 가지고 있는 특징이나 상태

- 동작

- 객체가 수행하는 기능, 업무 및 행위

-렌트카 회사에 관련된 주요 객체



직원



고객



자동차

1.객체 추출

모든 객체는 속성과 동작을 가진다.

■ 객체별 속성과 동작(기능)

직원



속성	이름,나이,직급..
동작	빌려준다,예약을 확인한다,차량 관리를 한다,...

자동차



속성	차명,색상,배기량,제조사,바퀴수,...
동작	달린다,멈춘다,주차한다,...

고객



속성	이름,주소,나이,전화번호,...
동작	예약한다,사용한다,지불한다,

2.객체 모델링

•프로그래밍 시 실제로 필요한 속성과 동작을 추출하는 과정

직원



속성	이름, 나이 , 직급..
동작	빌려준다, 예약을 확인한다, 차량 관리를 한다, ...

자동차



속성	차명, 색상, 배기량, 제조사, 바퀴수 , ...
동작	달린다, 멈춘다, 주차한다, ...

고객



속성	이름, 주소, 나이 , 전화번호, 아이디, 비밀번호
동작	예약한다, 사용한다, 지불한다,

3.클래스 변환

▪ 클래스의 정의

- 객체 모델링을 통해 추출된 객체를 자바 클래스로 변환한 것

객체이름
속성1 속성2 속성3
기능1 기능2

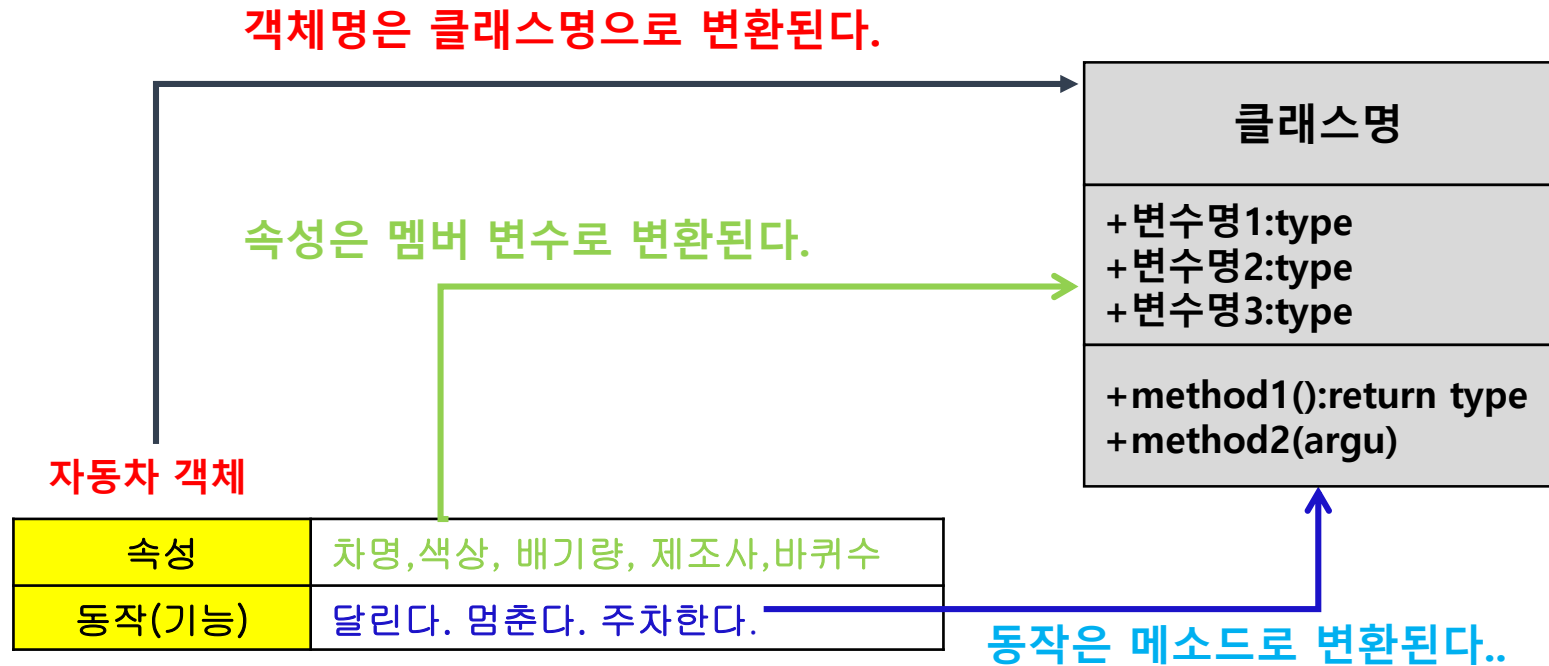


클래스명
+ 변수명1:type + 변수명2:type + 변수명3:type
+method1():return type +method2(argu)

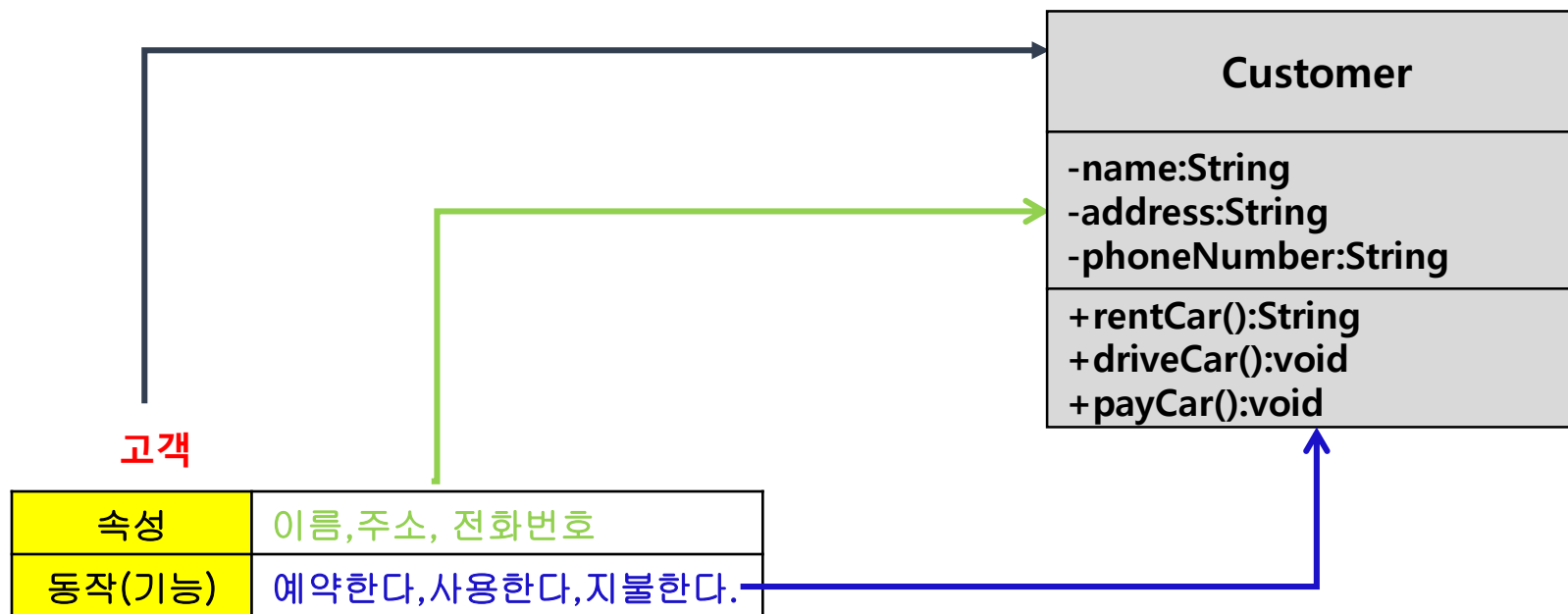
UML 접근 지정자 기호

+:public
#:protected
- :private

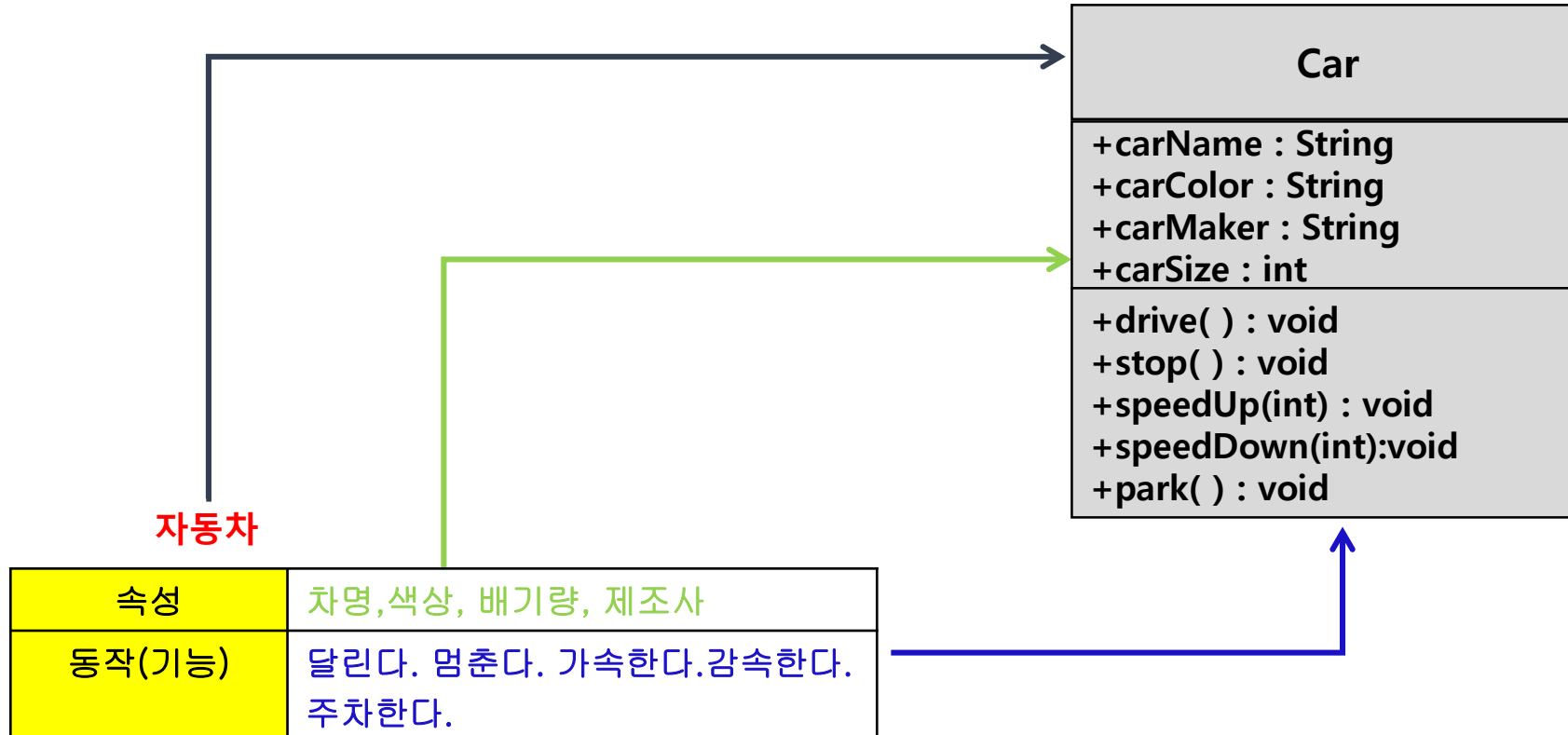
3.클래스 변환 과정



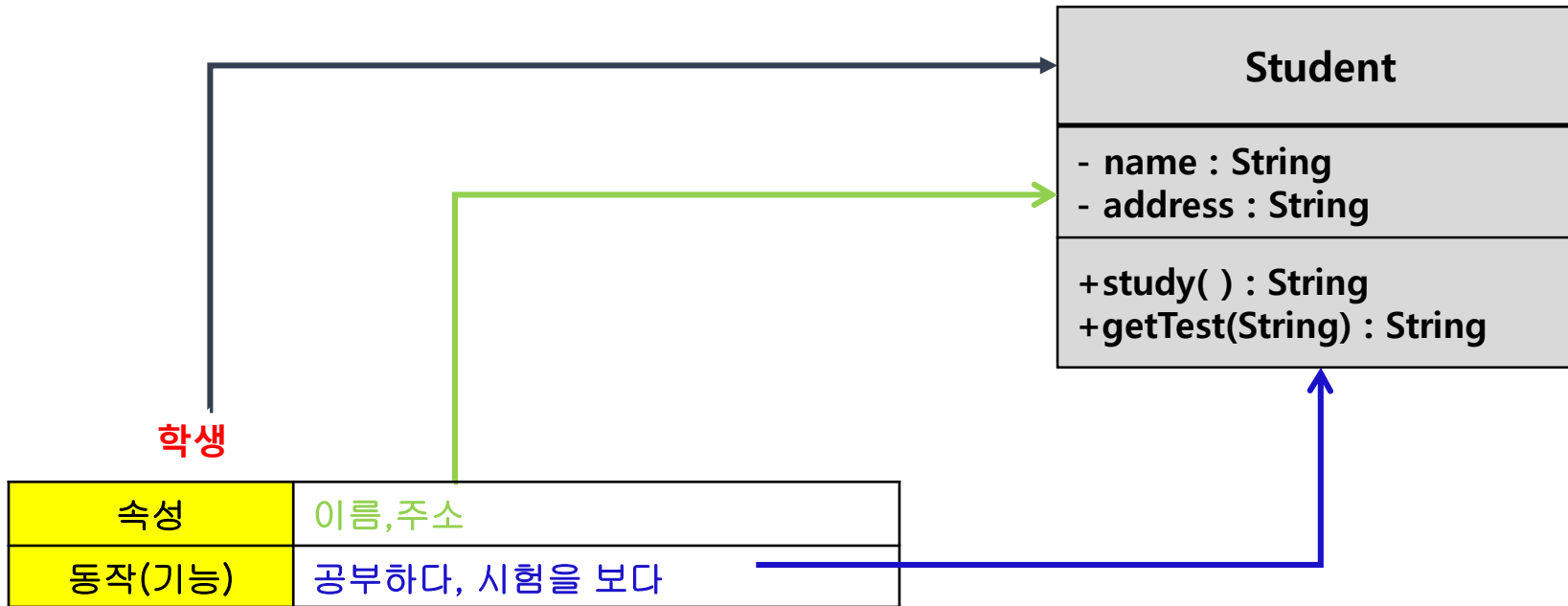
- 고객 객체 클래스 변환 과정



- 자동차 객체 클래스 변환 과정



- 학생 객체 클래스 변환 과정



4.클래스 구현

- 자바 클래스 형식

```
[지정자] class 클래스명{  
    멤버 변수(인스턴스 변수)  
    멤버 메서드  
    생성자  
}
```

- 지정자 형식

- 접근 지정자 : **public**, <default>, **protected**, **private**
- 일반 지정자 : **static**, **abstract**, **final** 등
- 일반적으로 접근 지정자와 일반 지정자를 조합하여 사용한다.

1.멤버변수(인스턴스 변수)

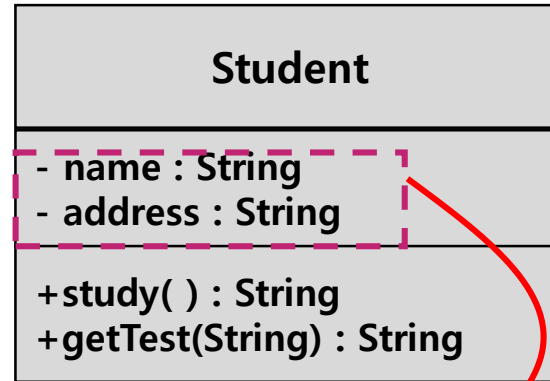
- 정의

- 객체 모델링에서 추출한 속성을 클래스에서 멤버 변수로 표현한다.
- 멤버 필드, 인스턴스 변수, 속성(attribute)라고도 한다.

- 형식

[접근 지정자] [일반 지정자] 데이터 형 변수명;

- 멤버변수 변환 예



```
1 public class Student {
2     | private String name; |
3     | private String address; |
4
5     public void study( ){
6         System.out.println("공부를 합니다.");
7     }
8     public String getTest(String score){
9         return score;
10    }
11 }
```

A red arrow points from the dashed pink box in the UML diagram to the corresponding code lines in the Java code block.

1.멤버 메서드

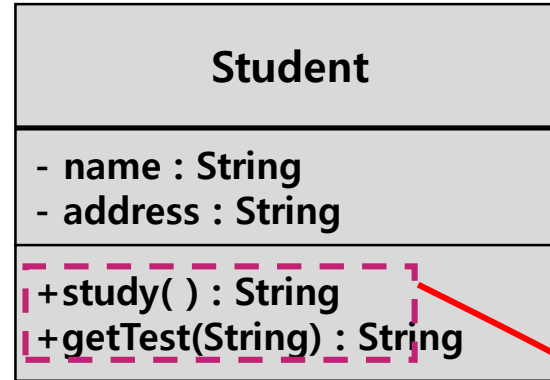
- 정의

•객체 모델링에서 추출한 동작을 클래스에서 멤버 메서드로 구현

- 형식

```
[접근 지정자] [일반 지정자] [리턴 타입] 메서드명([매개변수])  
{  
    //메서드의 기능 구현  
    ...  
}
```

- 멤버 메서드 변환 예



```
1 public class Student {  
2     private String name;  
3     private String address;  
4  
5     public String study( ){  
6         System.out.println("공부를 합니다.");  
7     }  
8  
9  
10    public String getTest(String score){  
11        return score;  
12    }  
13 }
```

■ 자바 메서드 분류

- 일반 메서드 : 클래스의 일반적인 기능을 수행한다(기능 구현).
- getter 메서드 : 멤버 변수의 값을 얻어올 때 사용
- setter 메서드 : 멤버 변수에 값을 할당할 때 사용

■ 일반 메서드

```
1 public class Student {  
2     private String name;  
3     private String address;  
4  
5     public void study( ){  
6         System.out.println("공부를 합니다.");  
7     }  
8  
9  
10    public String getTest(String score){  
11        return score;  
12    }  
13 }
```

```
1 public class Car{  
2     int velocity;  
3     int carName;  
4  
5     public void speedUp( ){  
6         velocity+=1;  
7     }  
8  
9     public void speedDown( ){  
10        velocity-=1;  
11        if(velocity < 0)  
12            velocity=0;  
13    }  
14  
15    public void stop( ){  
16        velocity=0;  
17    }  
18 }
```

getter/setter 메서드

```
1 public class Student {  
2     private String name;  
3     private String address;  
4  
5     public String getName() {  
6         return name;  
7     }  
8  
9     public String getAddress() {  
10        return address;  
11    }  
12  
13    public void setName(String _name) {  
14        name=_name;  
15    }  
16  
17    public void setAddress(String _address) {  
18        address=_address;  
19    }  
20 }
```

private이므로 외부에서 직접적으로 접근할 수 없다.

외부에서는 getter/setter 메서드가 public이므로 호출할 수 있다.

5.클래스 인스턴스(class instances)

▪ 인스턴스 정의

- 정의한 클래스를 이용하여 컴퓨터 메모리에 클래스의 속성과 기능을 가지는 객체
- '클래스 객체'라고도 한다.

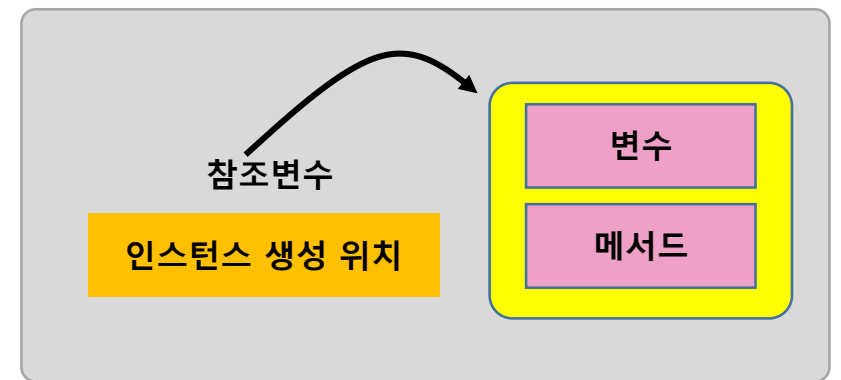
▪ 인스턴스 생성 방법

- new 키워드를 이용한다.

▪ 인스턴스 사용 방법

- 인스턴스와 같은 클래스타입 변수 선언
클래스명 변수명;
- 인스턴스 생성(객체 생성)
변수=**new** **클래스명**();

메모리



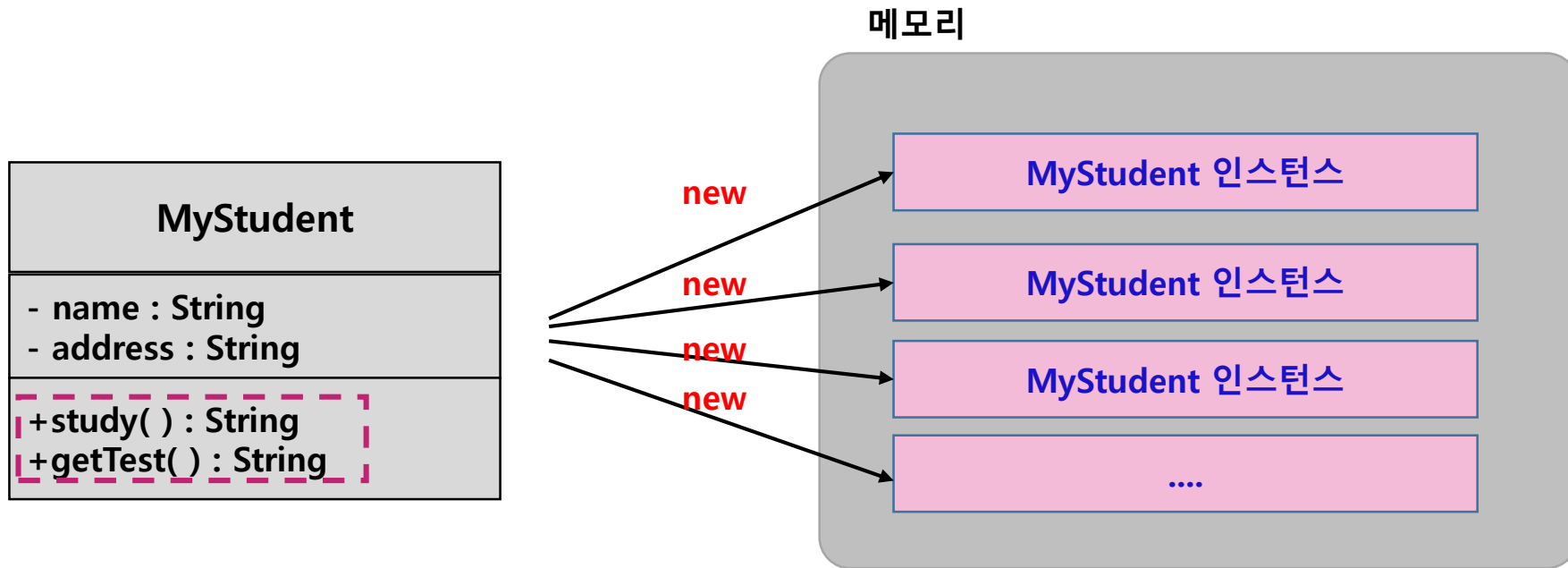
- 클래스 타입 변수(참조 변수) 사용하기

```
1  int a;  
2  a=5;  
3  a=5.6f; //오류 발생  
4  
5  Student s;  
6  s=new Student( );  
7  //Student s=new Student();  
8  s=new College( ); //오류 발생  
9  
10 College c;  
    c=new Student( ); //오류 발생
```

생성된 인스턴스는 같은 클래스 타입
참조 변수에 할당해야 한다.

클래스와 인스턴스의 관계

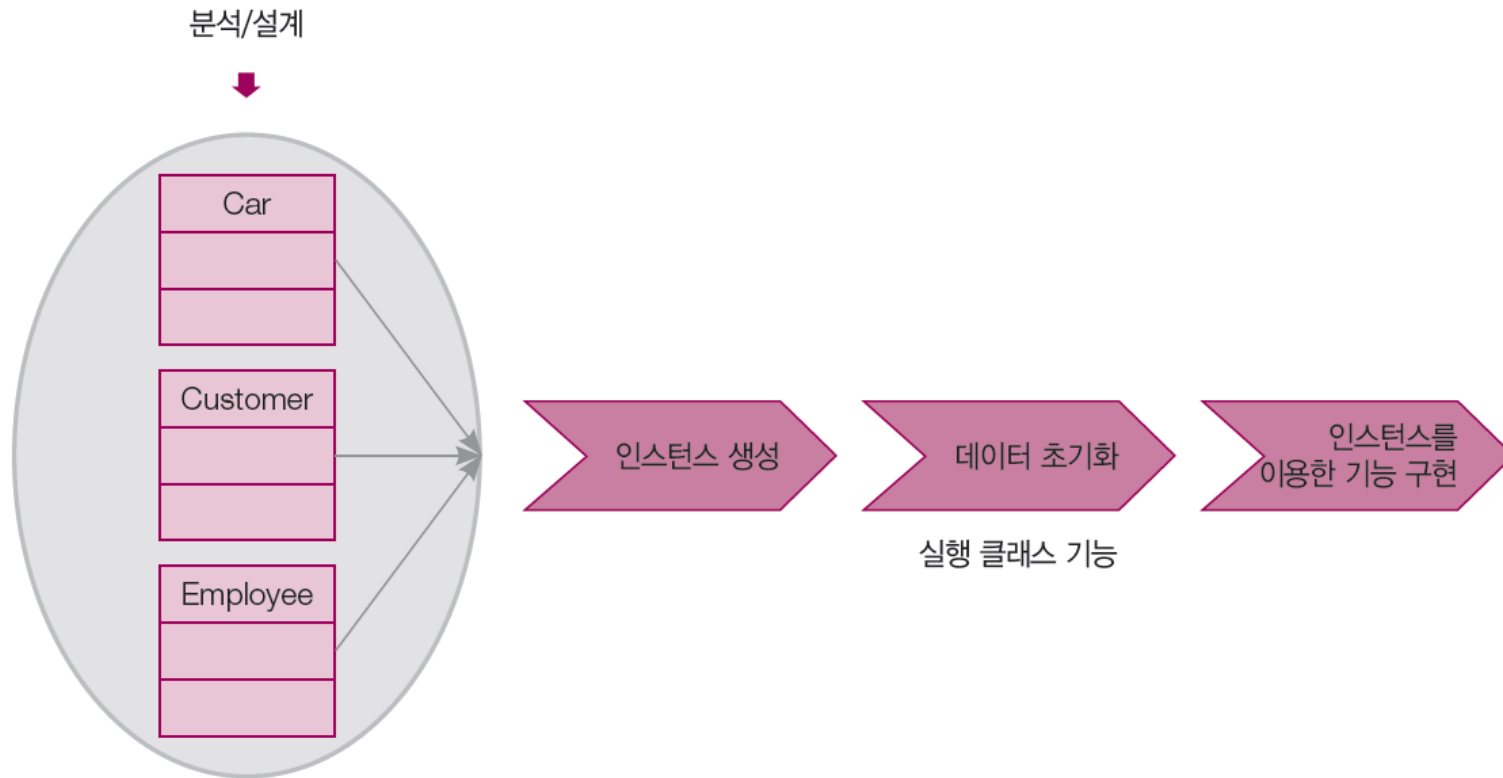
- 건물로 비유하면 클래스는 건물의 설계도이고, 인스턴스는 그 설계도로 만들어진 건물이다.
- 클래스는 한개만 존재하고 인스턴스는 언제든지 메모리에 만들어서 사용할 수 있다.



■ 실행 클래스

분석,설계한 후 구현된 클래스를 사용해서 최종적인 기능을 구현하다.

-부품을 조립하는 컨베이어 벨트와 같은 기능을 한다.



■ 실행 클래스 사용 예제

```

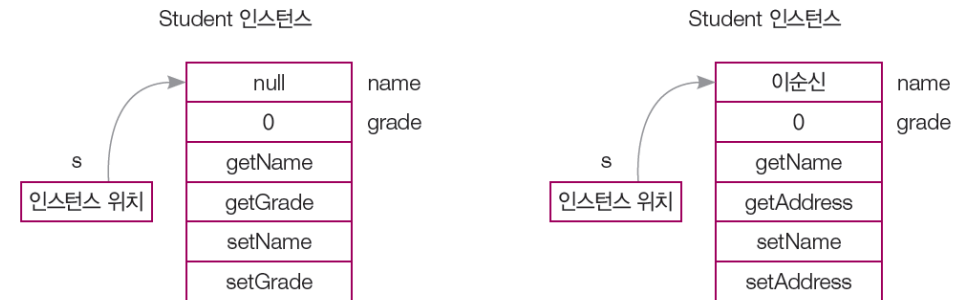
1 public class Student{
2     private String name;
3     private int grade;
4
5     public String getName( ){
6         return name;
7     }
8
9     public int getGrade( ){
10        return grade;
11    }
12
13    public void setName(String _name){
14        name=_name;
15    }
16
17    public void setGrade(int _grade){
18        grade=_grade;
19    }
20 }

```

```

1 public class StudentTest {
2     public static void main(String[] args){
3         Student s= new Student( ); //메모리에 인스턴스를 생성한다.
4
5         s.setName("이순신");
6         String name=s.getName( );
7         System.out.println("학생 이름은 "+name);
8     }
9 }

```



메모리에 생성된 인스턴스 상태

```

Console
<terminated> StudentTest (1)
학생 이름은 이순신

```

■ 실습 예제2

- 배열의 요소의 총합과 평균을 구하는 기능과 배열의 요소의 값을 오름차순으로 출력해주는 기능을 하는 클래스를 구현하라(ArrayUtil.java).
- 위 클래스의 인스턴스를 생성한 후 그 기능을 이용하는 실행 클래스를 구현하라(ArrayTest.java).

```
1 public class ArrayUtil {
2     //배열 요소값들의 총합과 평균을 구하는 메서드
3     public void calcArray(int[] num){
4         int total=0; //총합을 저장하는 변수
5         float average=0f; //평균을 저장하는 변수
6
7         for(int i=0; i<num.length;i++){
8             total+=num[i];
9         }
10
11         average=(float)total/num.length;
12         System.out.println("배열의 총합:"+total+",배열의 평균:"+average);
13     }
```

```
//배열요소값을 오름차순으로 정렬하는 메서드
14 public void sortArray(int[] num){
15     int temp=0;
16
17     for(int i=0; i<num.length;i++){
18         for(int j=i+1; j<num.length;j++){
19             if(num[i]>num[j]){
20                 temp=num[i];
21                 num[i]=num[j];
22                 num[j]=temp;
23             }
24         }
25     } //end for
26
27     for(int i=0; i<num.length;i++){
28         System.out.print(num[i]+"Wt");
29     }
30 }
31 }
```

```

1 public class ArrayTest {
2     public static void main(String[] args) {
3         //지역변수 선언
4         int[] score1={23,45,67,87,88,99,13,77,65};
5         int[] score2={56,78,98,88,77,55,43,21,99,67};
6
7         //인스턴스 생성
8         ArrayUtil u=new ArrayUtil();
9
10        //메서드 호출하기
11        u.calcArray(score1);
12        u.sortArray(score1);
13
14        System.out.println();
15        u.calcArray(score2);
16        u.sortArray(score2);
17    }
18 }

```

Problems @ Javadoc Declaration Console History Debug

<terminated> ArrayTest [Java Application] C:\Program Files\Java\jdk1.8.0_51\bin\javaw.exe (2015. 8. 31. 오전 11:19:25)

배열의 총합 : 564, 배열의 평균 : 62.666668

13	23	45	65	67	77	87	88	99
----	----	----	----	----	----	----	----	----

배열의 총합 : 682, 배열의 평균 : 68.2

21	43	55	56	67	77	78	88	98	99
----	----	----	----	----	----	----	----	----	----

- 실습 예제3

토지세는 3년 보유한 토지에 대해서는 매년 5%씩, 최대 50%까지 감면될 수 있다. 즉, 토지 소유 기간이 3년 미만인 토지는 100%, 소유 기간이 12년 이상인 토지는 50% 감면된다.

A : 표에 의해 계산된 면적세액

N : 토지 보유 기간(년)

지방 교육세 : 연세액(A)의 30%


```
1 public class LandTaxUtil {
2     public float calcLandTax(int type,int size,int year){
3         float total_tax=0f;
4         float tax=0f;
5         float area_tax=0f; //면적당토지세액
6         int tax_rate=0;
7         int n=0;
8         if(type==1){ //건물이 없는 토지인 경우
9             System.out.println("토지 종류 : 건물이 없는 토지");
10            if(size<=1800)
11                tax_rate=1800;
12            else if(size <=2500)
13                tax_rate=2000;
14            else
15                tax_rate=3000;
16            area_tax=tax_rate*size;
17            System.out.println("토지면적 세액 : "+area_tax);
18            if(year<3)
19                n=0;
20            else if(year <13)
21                n=year-2;
22            else
23                n=10;
24            tax=area_tax-area_tax*0.05f*n;
25        }else {
```

```
26        System.out.println("토지 종류 : 건물이 있는 토지 ");
27        if(size<=1800)
28            tax_rate=3600;
29        else if(size <=2500)
30            tax_rate=4000;
31        else
32            tax_rate=6000;
33        area_tax=tax_rate*size;
34        System.out.println("토지면적 세액은 "+area_tax);
35        if(year <3)
36            n=0;
37        else if(year <13)
38            n=year-2;
39        else
40            n=10;
41        tax=area_tax-area_tax*0.05f*n;
42    }
43    total_tax=tax+0.3f*area_tax; //최종 토지 보유세에 지방 교육세를
44                                포함한다.
45    return total_tax;
46    }
47 }
```

```
1 public class LandTest {
2     public static void main(String[] args) {
3         float my_total_tax=0f;
4         int my_land_type=1;
5         int my_land_size=25000;
6         int my_land_own_year=5;
7         float your_total_tax=0f;
8         int your_land_type=2;
9         int your_land_size=15000;
10        int your_land_own_year=3;
11
12        LandTaxUtil util=new LandTaxUtil( );
13        my_total_tax=util.calcLandTax(my_land_type, my_land_size, my_land_own_year);
14        System.out.println("내 토지 보유세액은 "+my_total_tax+"입니다.");
15        System.out.printf("내 토지 보유액은>>%5.1f원입니다.\n",my_total_tax);
16
17        your_total_tax=util.calcLandTax(your_land_type, your_land_size,your_land_own_year);
18        System.out.println("당신의 토지 보유세액은 "+your_total_tax+"입니다.");
19        System.out.printf("당신의 토지 보유세액은>>%5.1f원입니다.\n",your_total_tax);
20    }
```

Console

<terminated> LandTest [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (2015. 3. 13.)

토지 종류 :건물이 없는 토지

토지면적 세액은 7.5E7

내 토지 보유세액은 8.625E7입니다.

내 토지 보유세액은 86250000.0원입니다.

토지 종류 : 건물이 있는 토지

토지면적 세액은 9.0E7

당신의 토지 보유세액은 1.125E8입니다.

내 토지 보유세액은 112500000.0원입니다.

■ 생성자 정의

- 클래스가 객체 생성될 때 맨 처음 호출되는 것
- 반드시 클래스명과 동일해야 한다.

■ 생성자 기능

- 클래스 인스턴스를 메모리에 생성한다.
- 주로 인스턴스 변수를 초기화한다.

■ 생성자 형식

```
[접근 지정자] 클래스명([매개변수]){  
    //인스턴스 변수 초기화  
    //명령문  
    ....  
}
```

■ 생성자 특징

- 리턴 타입이 없다.
- 생성자를 구현해주지 않으면 컴파일러가 컴파일 시 default 생성자를 추가한다.
- 생성자를 명시적으로 구현하면 default 생성자는 추가되지 않는다.
- default 생성자 형식

public 클래스명() { }

```
1 public class Student{
2     private String name;
3     private int grade;
4     public String getName( ){
5         return name;
6     }
7     public int getGrade( ){
8         return grade;
9     }
10
11     public void setName(String _name){
12         name=_name;
13     }
14     public void setGrade(int _grade){
15         grade=_grade;
16     }
17 }
```

```
1 public class StudentTest {
2     public static void main(String[] args){
3         Student s= new Student( );
4         s.setName("이준진");
5         String name=s.getName( );
6         System.out.println(" 학생 이름은 " +name);
7     }
8 }
9
```

컴파일 시 자동으로 추가된다.

public Student(){ }

■ 사용자 정의 생성자 사용 예제

```

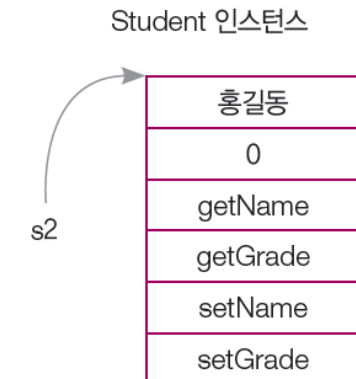
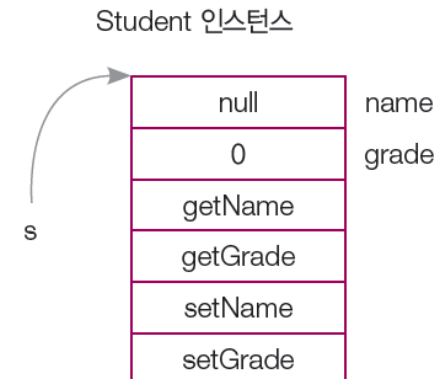
1  public class Student{
2      private String name;
3      private int grade;
4
5      public Student(String _name){
6          name=_name;
7      }
8      public Student( ){
9      }
10
11     public String getName( ){
12         return name;
13     }
14     public int getGrade( ){
15         return grade;
16     }
17     public void setName(String _name){
18         name=_name;
19     }
20     public void setGrade(int _grade){
21         grade=_grade;
22     }
23 }

```

```

1  public class StudentTest {
2      public static void main(String[ ] args){
3          Student s= new Student( );
4          Student s2= new Student("홍길동");
5          //s.setName("이순신");
6          String name=s.getName( );
7          int grade=s.getGrade( );
8          System.out.println("첫 번째 학생의 이름은 " + name+", 학년은
9                                  "+grade);
10
11         name=s2.getName( );
12         grade=s2.getAge( );
13         System.out.println("두 번째 학생의 이름은 "+name+", 학년은
14                                 "+grade);
15     }
16 }

```



▪ 매개변수가 두개인 생성자 사용 예제

```

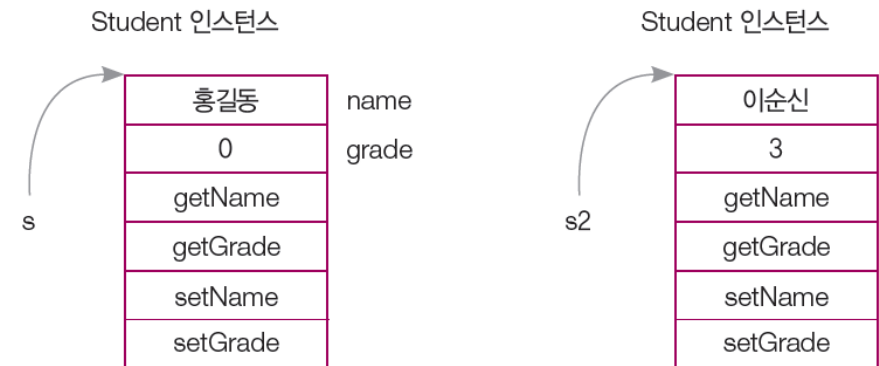
1 public class Student{
2     private String name;
3     private int grade;
4
5     public Student(String _name,int _grade){
6         name=_name;
7         grade=_grade;
8     }
9     public Student(String _name){
10        name=_name;
11    }
12    public Student( ){
13    }
14
15    public String getName( ){
16        return name;
17    }
18    public int getGrade( ){
19        return grade;
20    }
21    public void setName(String _name){
22        name=_name;
23    }
24    public void setGrade(int _grade){
25        grade=_grade;
26    }
27 }

```

```

1 public class StudentTest2 {
2     public static void main(String[ ] args){
3         Student s= new Student("홍길동");
4         Student s2= new Student("이순신",3);
5         String name=s.getName( );
6         int grade=s.getGrade( );
7         System.out.println("첫번째 학생의 이름은 " + name+" , 학년은 "+grade);
8
9
10        name=s2.getName( );
11        grade=s2.getGrade( );
12        System.out.println("두번째 학생의 이름은 "+name+" , 학년은 "+grade);
13
14    }
15 }

```



■ 생성자에 다른 명령문 추가한 경우

```

1 public class Student{
2     private String name;
3     private int grade;
4
5     public Student(String _name,int _grade){
6         System.out.println("인자가 2개인 생성자 호출");
7         name=_name;
8         grade=_grade;
9     }
10    public Student(String _name){
11        System.out.println("인자가 1개인 생성자 호출");
12        name=_name;
13    }
14    public Student( ){
15        System.out.println("디폴트 생성자 호출");
16    }
17
18    public String getName( ){
19        return name;
20    }
21    ....
22
23

```

```

1 public class StudentTest2 {
2     public static void main(String[ ] args){
3         Student s2= new Student("이순신",3);
4         Student s= new Student("홍길동");
5
6         String name=s.getName( );
7         int grade=s.getGrade( );
8         System.out.println("첫번째 학생의 이름은 " + name+" , 학년은
9                                 "+grade);
10
11        name=s2.getName( );
12        grade=s2.getGrade( );
13        System.out.println("두번째 학생의 이름은 "+name+" , 학년은
14                                "+grade);
15    }
16 }

```

Problems @ Javadoc Declaration Console

<terminated> StudentTest2 [Java Application] C:\Program Files\Java\

인자가 2개인 생성자 호출
 인자가 1개인 생성자 호출
 첫번째 학생의 이름은 홍길동 ,학년은 0
 두번째 학생의 이름은 이순신 ,학년은 3

- 'address' 속성을 추가한 경우

```
1  public class Student{
2      private String name;
3      private int grade;
4      private String address;
5
6      public Student(String _name,int _grade){
7          System.out.println("인자가 2개인 생성자 호출");
8          name=_name;
9          grade=_grade;
10     }
11     public Student(String _name){
12         System.out.println("인자가 1개인 생성자 호출");
13         name=_name;
14     }
15     public Student( ){
16         System.out.println("디폴트 생성자 호출");
17     }
18
19     public String getName( ){
20         return name;
21     }
22     ....
23
24 }
```


■ 생성자 사용 시 주의할 점

```

1 public class Student{
2     private String name;
3     private int grade;
4
5     public Student(String _name,int _grade){
6         System.out.println("인자가 2개인 생성자 호출");
7         name=_name;
8         grade=_grade;
9     }
10    public Student(String _name){
11        System.out.println("인자가 1개인 생성자 호출");
12        name=_name;
13    }
14
15    /*
16    public Student( ){
17        System.out.println("디폴트 생성자 호출");
18    }
19    */
20
21    public String getName( ){
22        return name;
23    }
24    ....

```

```

1 public class StudentTest{
2     public static void main(String[] args){
3         Student s1=new Student();
4         Student s2=new Student("이순신",23);
5
6         String name=s1.getName( );
7         int age=s1.getAge( );
8         System.out.println("첫 번째 학생의 이름은 " + name+",
9                             나이는 "+age);
10
11        name=s2.getName( );
12        age=s2.getAge( );
13        System.out.println("두 번째 학생의 이름은 "+name+",
14                            나이는 "+age);
15    }
16 }

```

사용자가 직접 생성자를 정의한
경우디폴트 생성자는
자동으로 추가되지 않는다.

■ 오버로딩 생성자의 정의와 규칙

• 정의

- 클래스에 동일한 이름의 생성자가 여러 개 존재하는 것
- 클래스이름을 재사용한다.**

• 규칙

- 반드시 생성자 매개변수 형식이 달라야 한다(**순서, 타입, 개수**)

```
1 public class Student{
2     private String name;
3     private int grade;
4
5     public Student(String _name,int _grade){
6         System.out.println("인자가 2개인 생성자 호출");
7         name=_name;
8         grade=_grade;
9     }
10
11    public Student(String _name){
12        System.out.println("인자가 1개인 생성자 호출");
13        name=_name;
14    }
15
16    public Student(){
17        System.out.println("디폴트 생성자 호출");
18    }
19
20    public String getName(){
21        return name;
22    }
23
24    .....
```

■ 오버로딩 메서드의 정의와 규칙

- 정의
 - 같은 클래스에서 동일한 이름의 메서드가 여러 개 존재하는 것
 - 메서드명을 재사용한다.
- 규칙
 - 반드시 메서드 매개변수 형식이 달라야 한다(순서, 타입, 개수)
 - 리턴 타입은 달라도 상관없다.

```
1 public class ClassUtil {  
2     public void get( ){}  
3     public void get(int n){ }  
4     public void get(String n){ }  
5     public void get(String n, int a){ }  
6     //public int get( ) { }  
7 }  
8
```

■ 오버로딩 메서드 사용 예

-오버로딩 적용 전 메서드

```
1 public class AddUtil1 {  
2     public int add2(int a,int b){  
3         int x=a+b;  
4         return x;  
5     }  
6  
7     public int add3(int a,int b,int c){  
8         int x=a+b+c;  
9         return x;  
10    }  
11  
12    public int add4(int a,int b,int c,int d){  
13        int x=a+b+c+d;;  
14        return x;  
15    }  
16 }
```

-오버로딩 적용 후 메서드


```
1 public class AddUtil2 {  
2     public int add(int a,int b){  
3         int x=a+b;  
4         return x;  
5     }  
6  
7     public int add(int a,int b,int c){  
8         int x=a+b+c;  
9         return x;  
10    }  
11  
12    public int add(int a,int b,int c,int d){  
13        int x=a+b+c+d;;  
14        return x;  
15    }  
16 }
```

■ 오버로딩 메서드 사용 실습 예제4

1에서 임의의 자연수까지의 합과 임의의 두 수 사이의 자연수의 합을 구하는 기능을 overloading을 이용하여 메서드를 구현하라(MyUtil.java).
MyTest.java에서 두 수를 입력한 후 각각의 결과값을 출력하라.

```
1 public class MyUtil {
2     public void summarize(int num1){
3         int sum=0;
4         for(int i=0; i<=num1;i++){
5             sum+=i;
6         }
7         System.out.println("1에서 "+num1+" 사이에 있는
8                             자연수들의 합은 "+sum);
9     }
10
11 // public void summarize2(int num1,int num2){
12     public void summarize(int num1,int num2){
13         int sum=0;
14         for(int i=num1; i<=num2;i++){
15             sum+=i;
16         }
17         System.out.println(num1+"과 "+num2+" 사이의 자
18                             연수들의 합은 "+sum);
19     }
20 }
```

```
1 public class MyTest {
2     public static void main(String[ ] args){
3         int num1=10;
4         int num2=100;
5
6         MyUtil my=new MyUtil( );
7         my.summarize(num1);
8         my.summarize(num1,num2);
9     }
10 }
```

Console 
<terminated> MyTest (1) [Java Application] C:\Program Files\Wj
1과10사이에 있는 자연수들의 합은 55
10과100사이의 자연수들의 합은 5005

- 오버로딩 메서드 사용 실습 예제5

다음 두 배열을 차례로 전달받아 배열의 요소를 오름차순으로 출력하는 메서드와 임의의 두 배열을 전달받아 두 배열의 요소를 비교한 후 오름차순으로 출력하는 메서드를 overloading 을 이용(메서드명 : sortArray)하여 클래스를 구현하라(ArrayUtil.java).

```
int[ ] num1={1,3,4,56,78,91,23};  
int[ ] num2={21,5,65,57,79,17,33};
```

```
1 public class ArrayUtil {  
2     //배열을 오름차순으로 정렬하는 메서드  
3     public void sortArray(int[] arr1){  
4         ...  
5     }  
6  
7     //두 배열 전체의 값을 비교한후 오름차순으로 정렬하는 메서드  
8     public void sortArray(int[] arr1,int[] arr2){  
9         ...  
10    }  
11 }  
12
```

```
1 public class MyTest {  
2     public static void main(String[ ] args){  
3         int[ ] num1={1,3,4,56,78,91,23};  
4         int[ ] num2={21,5,65,57,79,17,33};  
5  
6         ArrayUtil a=new ArraytUtil( );  
7         a.sortArray(num1);  
8         a.sortArray(num1,num2);  
9  
10    }  
}
```

■ 패키지 정의와 특징

- 정의
 - 같은 기능을 하는 클래스들을 모아놓은 그룹
- 특징
 - 같은 기능을 하는 클래스들을 편리하게 관리하기 위해 사용한다.
 - 윈도우의 폴더와 비슷하다.
 - 자바에서 제공하는 API는 모두 package로 제공된다.

■ 패키지 형식

- : package 패키지명;
- : package 패키지명1.패키지명2;
- 규칙
 - 반드시 클래스 첫 라인에서 사용한다.
 - 패키지명은 소문자로 작성한다.
 - 자바 파일 최상단에 한 번만 선언한다.

■ 자바 패키지 명명법

• 역도메인 명명법

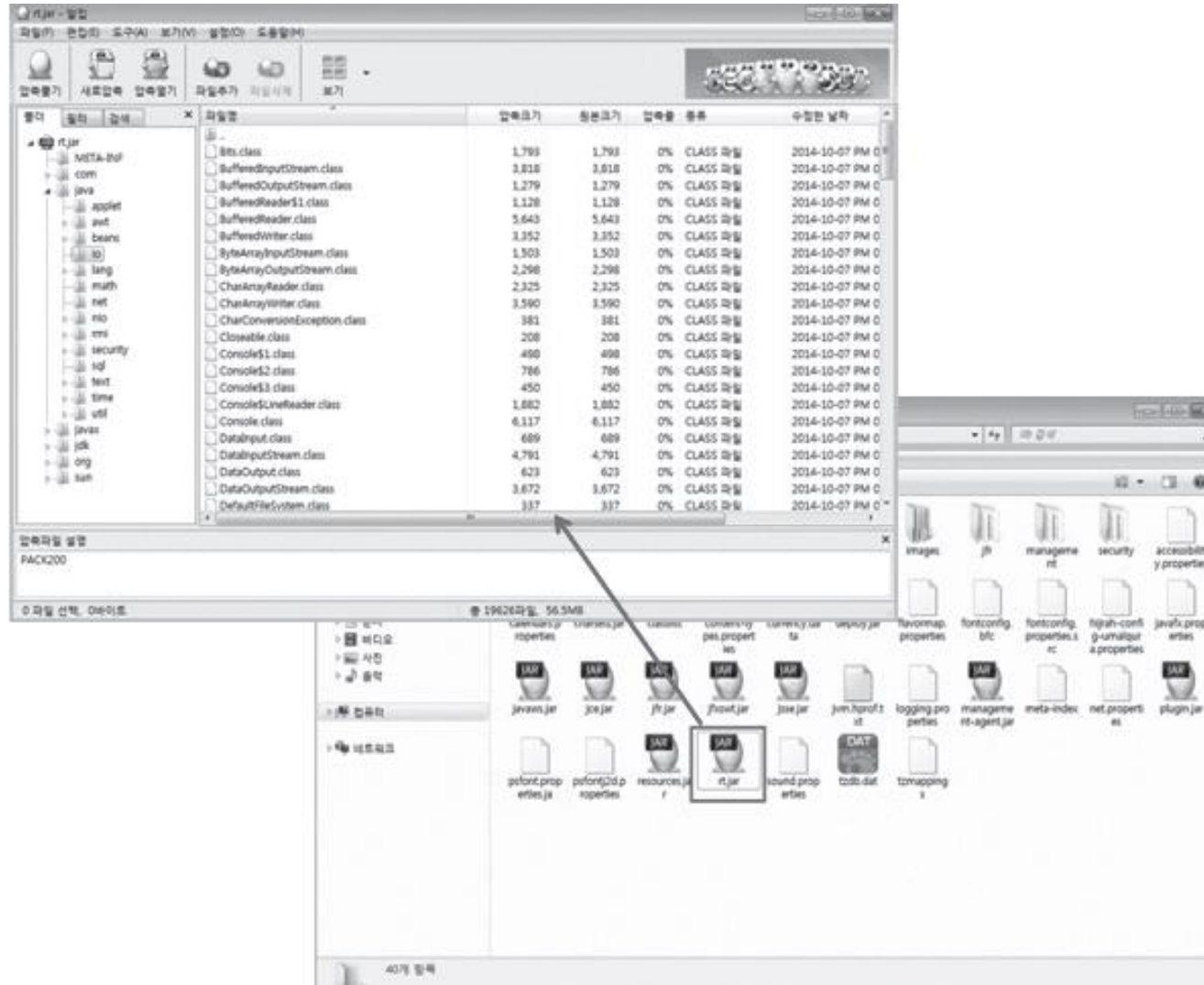
-자기가 가지고 있는 도메인을 역순으로 배열한다(jweb.com → com.jweb)

• 패키지 명명법의 예

-패키지 뒤에 클래스명을 붙인다. 이때 주의해야 할 사항은 클래스명은 반드시 단어의 첫글자를 대문자로 써야 한다는 것이다.

예) com.jweb.prjname.member.MemWindow(패키지명.프로젝트명.기능명.클래스명)

패키지로 제공되는 자바 API



■ 임포트의 용도

-패키지가 다른 클래스에 접근할 때에 사용한다.

• 사용법

import 패키지명.클래스명;

[import 패키지명.*;]

성능에는 차이가 없으나 가독성을
위해서 소스에서 사용하고 있는
클래스를 구체적으로 나열해 주는 것이 좋다.

• 특징

-반드시 클래스보다 먼저 선언되어야 한다(패키지 다음에 선언).

-모든 자바 API를 사용할 때에는 반드시 import해야 한다.

-java.lang 패키지는 자동으로 import된다.

예)String 클래스

-import문은 여러 번 선언할 수 있다.

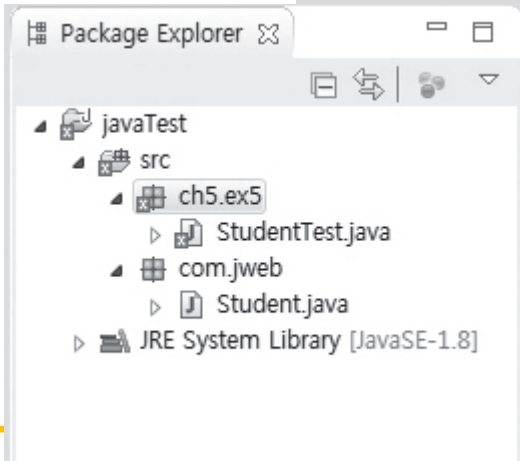
- 다른 패키지의 클래스 импорт하기

com.jweb에 있는 클래스

```
1 package com.jweb;  
2  
3 public class Student {  
4     private String name;  
5     private int grade;  
6     public Student(String _name){  
7         System.out.println("매개변수가 1개인 생성자 호출");  
8         name=_name;  
9     }  
10    public Student( ){  
11        System.out.println("디폴트 생성자 호출");  
12    }  
13  
14    public String getName( ){  
15        return name;  
16    }
```

ch5.ex5 패키지에 있는 실행 클래스

```
1 package ch5.ex5;  
2  
3 import com.jweb.Student;  
4  
5 public class StudentTest {  
6     public static void main(String[ ] args){  
7         Student s= new Student( );  
8         Student s2= new Student("홍길동");  
9         String name=s.getName( );  
10        int grade=s.getGrade( );  
11        System.out.println("첫 번째 학생의 이름은 " +  
12                                name+", 학년은 "+grade);  
13  
14        name=s2.getName( );  
15        grade=s2.getGrade( );  
16        System.out.println("두 번째 학생의 이름은  
17                                "+name+", 학년은 "+grade);  
18    }  
20 }
```

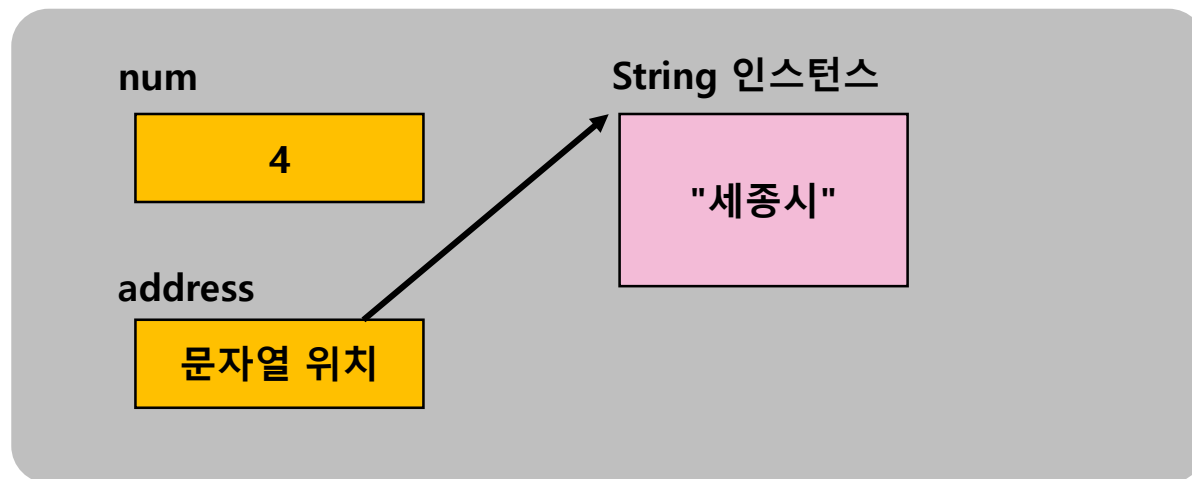


■ 각 변수의 특징

- 기본형 변수의 특징
 - 변수에 실제값이 저장된다.
- 참조형 변수의 특징
 - 메모리에 생성되어 있는 인스턴스의 위치값을 저장한다.

```
int num=4;  
String address="세종시";
```

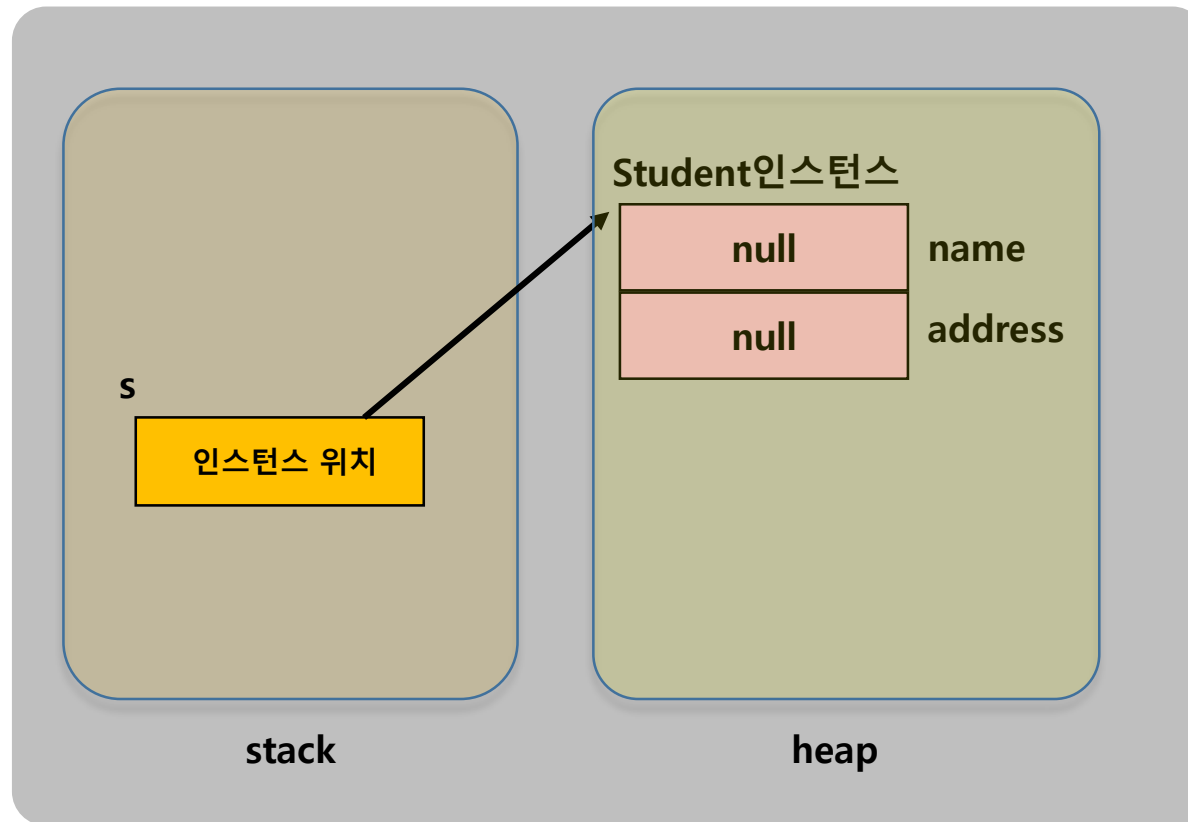
메모리



- 인스턴스 생성 후 참조 변수에 저장 과정

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

메모리



- 자바 객체 지향 프로그래밍 과정

아이디어 도입

----- 구현될 프로그램에 대한 요구를 나타낸다.

아이디어 분석

----- 주체의 관점에서 아이디어를 분석한다.

객체 추출

----- 주체의 관점에서 객체를 추출한 후 속성과 동작을 나열한다.

객체 모델링

----- 개발자의 관점에서 객체의 속성과 동작을 정제한다.

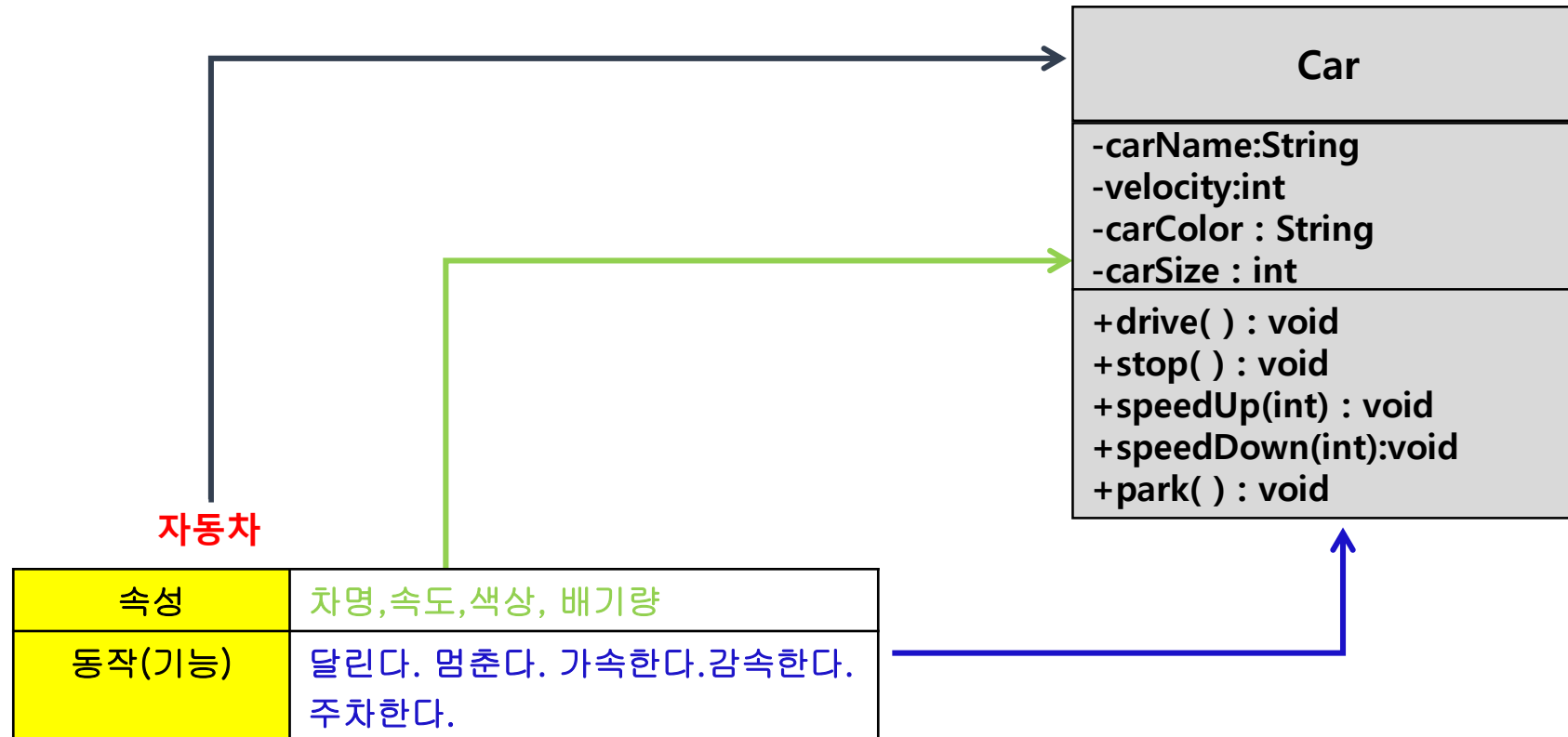
클래스 변환

----- 객체를 클래스로 변환한다.

인스턴스 생성 및 사용

----- 메모리에 인스턴스로 생성한 후 사용한다.

- 자동차 객체 클래스로 변환하기



```
1 public class Car{
2     private String carName;
3     private String carColor;
4     private int carSize ;
5     private int velocity;
6     public String getCarName( ){
7         return carName;
8     }
9     public void setCarName(String _name){
10         carName=_name;
11     }
12     public String getCarColor( ){
13     }
14     public void setCarColor(String _carColor){
15         carColor=_carColor;
16     }
17     public int getCarSize( ){
18         return carSize;
19     }
20     public void setCarSize(int _carSize){
21         carSize = _carSize;
22     }
23     public int getVelocity{
24         return velocity;
25     }
26     public void setVelocity(int _velocity){
27         velocity = _velocity;
28     }
```

```
29     public void speedUp( ){
30         velocity = velocity + 1;
31     }
32     public void speedDown( ){
33         velocity =velocity -1;
34         if(velocity < 0)
35             velocity =0;
36     }
37     public void stop( ){
38         velocity =0;
39     }
40     public void park( ){
41         System.out.println("주차합니다.");
42     }
43 }
```



```
1 public class MyCarTest{
2     public static void main(String[ ] args){
3         Car myCar;
4         myCar = new Car( );
5         myCar.setCarName("소나타");
6         myCar.setCarColor("은색");
7         myCar.setCarSize(2000);
8         myCar.setVelocity(60);
9
10        String carName=myCar.getCarName( );
11        String carColor=myCar.getCarColor( );
12        int carSize=myCar.getCarSize( );
13        int velocity=myCar.getVelocity( );
14        myCar.speedUp( );
15        myCar.speedUp( );
16        velocity=myCar.getVelocity( );
17
18        System.out.println("내 차 정보 출력 : ");
19        System.out.println("차 이름 : " +carName +
20                            ", 색상 : " +velocity +
21                            ", 배기량 : " +carSize+"cc"+
22                            ", 현재 속도 : " + velocity+" 입니다.");
23    }
24 }
```

Console

<terminated> MyCarTest [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (201

내 차 정보 출력 :

차 이름 : 소나타, 색상 : 은색, 배기량 : 2000cc, 현재 속도 : 62 입니다.

■ 생성자 구현하기

```

1  public class Car{
2      private String carName;
3      private String carColor;
4      private int carSize ;
5      private int velocity;
6
7      //생성자 정의
8      public Car(String _carName,String _carColor,int _carSize,int _velocity){
9          carName=_carName;
10         carColor=_carColor;
11         carSize=_carSize;
12         velocity=_velocity;
13     }
14
15     public String getCarName( ){
16         return carName;
17     }
18     public void setCarName(String _name){
19         carName=_name;
20     }
21     ...

```

```

29     public class YourCarTest{
30         public static void main(String[ ] args){
31             Car yourCar;
32             yourCar = new Car("그랜저","검은색",2500,60);
33             /*
34             yourCar.setCarName("그랜저");
35             yourCar.setCarColor("검은색");
36             yourCar.setCarSize(2500);
37             yourCar.setVelocity(60);
38             */
39
40             String carName=yourCar.getCarName;
41             String carColor=yourCar.getCarColor( );
42             int carSize=yourCar.getCarSize( );
43             int velocity=yourCar.getVelocity( );
44             yourCar.speedUp( );
45             yourCar.speedUp( );
46             velocity=yourCar.getVelocity( );
47             System.out.println("당신 차 정보 출력>> ");
48             System.out.println("차 이름 : " +carName +
49                 ", 색상 : " +carColor +
50                 ", 배기량 : " +carSize+"cc"+
51                 ",현재속도 : " + velocity+" 입니다.");
52         }
53     }

```

Problems @ Javadoc Declaration Console

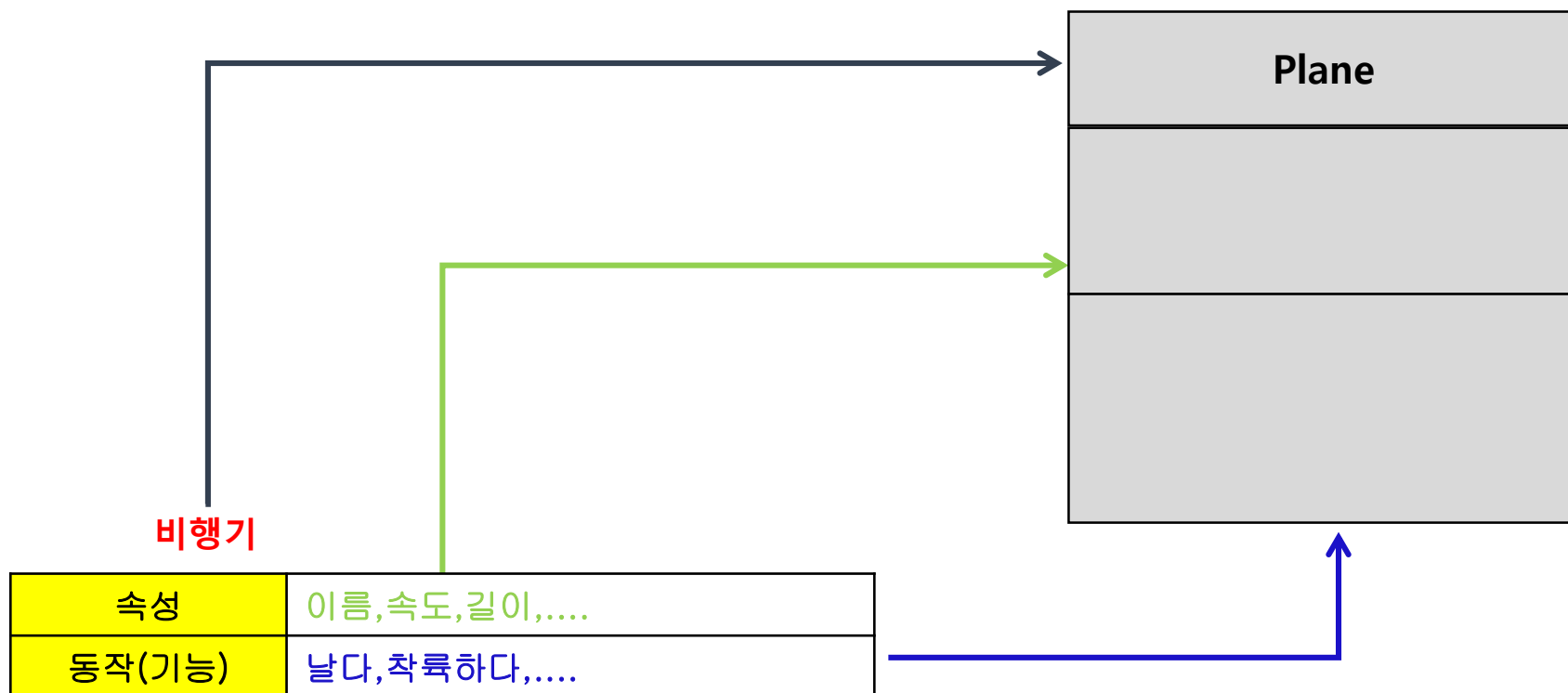
<terminated> YourCarTest [Java Application] C:\Program Files\Java\jre1.8.0_25\bin\javaw.exe (2015. 6. 1

당신 차 정보 출력>>

차 이름 : 그랜저 , 색상 : 검은색 , 배기량 : 2500cc , 현재속도 : 62 입니다 .

■ 실습예제6

1. 비행기를 클래스로 구현하라.
2. 스마트폰을 클래스로 구현하라.



■ 실습예제7

다음 글을 참고해서 우주(MyUniverse)를 클래스로 설계하라.

현실세계에서 모든 물체는 주변환경과 상호작용을 하고 있으며,외부로부터 약간의 영향이 개입되면 두 개의 파동함수는 더 이상 결맞음상태를 유지하지 못하고 결어긋남상태로 변한다.

...

...

그러나 결어긋남을 자연스럽게 확장하면 이 문제를 해결할 수 있다.

휠러의 또 다른 제자인 휴 에버렛 3세(Hugh Everett III)는 죽은 고양이와 살아 있는 고양이가 서로 다른 우주에 동시에 존재한다는 가설을 도입하여 선택과 관련된 문제를 우회적으로 해결하였다.

에버렛은 '다중우주이론'을 주제로 하여 1957년에 박사학위논문을 제출했으나, 당시만 해도 그런 황당한 이론에 관심을 갖는 사람은 거의 없었다. 그러나 세월이 지나면서 그의 이론은 점차 진가를 발휘하기 시작했고 지금은 양자역학의 역설을 해결해줄 가장 강력한 후보로 인정받고 있다.

출처:평행우주(미치오 카쿠 저, p.271)

우주(MyUniverse)

속성	...
동작(기능)	...

- 객체 지향 프로그래밍으로 렌트카 예약 프로그램 개발 과정
 - 일반인(subject)의 렌트카 회사 업무 분석



일반인(subject)

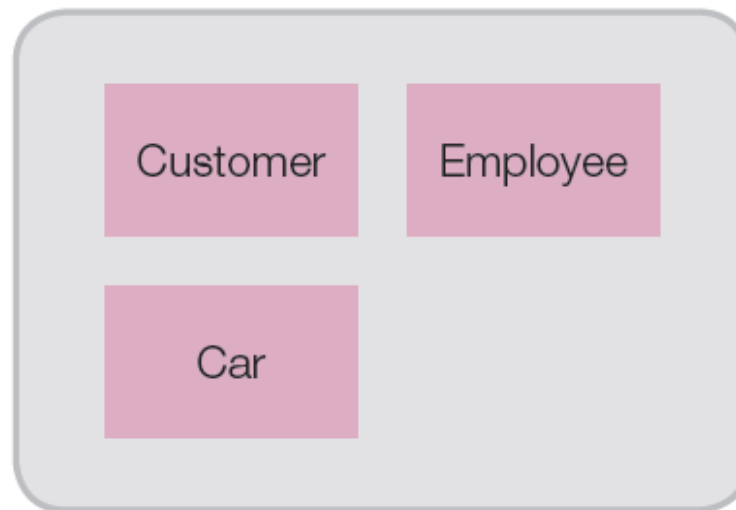


렌트카 회사에 관계되는 여러가지 객체

- 객체 지향 프로그래밍으로 렌트카 예약 프로그램 개발 과정
 - 렌터카 회사 업무를 분석한 후 객체 지향 프로그래밍 과정



개발자



자바 클래스

객체

고객

차

예약

클래스

Customer

Car

Reserve

실행 클래스

인스턴스
생성데이터
초기화기능
구현

고객(회원) 객체

속성	아이디, 비밀번호,이름,주소,전화번호
동작	회원정보를 조회한다. 회원등록한다. 회원정보를 수정한다. 회원정보를 삭제한다.

Member
id:String password:String name:String address:String phoneNum:String
+regMember() :void +viewMember():String +modMember():void +delMember():void

카(렌트카) 객체

속성	차번호,차명,색상, 배기량, 제조사
동작	차정보를 조회한다. 차정보를 등록한다. 차정보를 수정한다. 차정보를 삭제한다.

Car
carNumber: String carName: String carColor:String carSize:int carMaker:String
+checkCarInfo:String +regCarInfo:void +modCarInfo:void +delCarInfo:void

예약 객체

속성	예약차번호 예약일자 차이용시작일자 차반납예정일자
동작	차를 예약하다. 예약정보를 수정하다. 예약을 취소하다.

Reserve
resCarNumber: String resDate: String useBeginDate:String returnDate:String
+reserveCar:void +modReserveInfo:void +cancelReserveInfo:void

- 회원 클래스

```
1 package ex1.member;
2
3 public class Member {
4     String id;
5     String password;
6     String name;
7     String address;
8     String phoneNum;
9
10    //새로운 회원등록을 하는 메소드
11    public void regMember(){
12        System.out.println("회원 가입합니다.");
13    }
14
15    //기존 회원의 정보를 조회하는 메소드
16    public String viewMemeber(){
17        System.out.println("회원 정보를 조회합니다.");
18        return null;
19    }
20
21    //기존 회원의 정보를 수정하는 메소드
22    public void modMember(){
23        System.out.println("회원 정보를 수정합니다.");
24    }
25
26    //기존 회원의 정보를 삭제하는 메소드]
27    public void delMember(){
28        System.out.println("회원 정보를 삭제합니다.");
29    }
30 }
```

- 렌터카 클래스

```
1  package ex1.car;
2
3  public class Car {
4      String carNumber;
5      String carName;
6      String carColor;
7      int carSize;
8      String carMaker;
9
10     //차의 정보를 조회하는 메소드
11     public String checkCarInfo(){
12         System.out.println("렌트카 정보를 조회합니다.");
13         return null;
14     }
15
16     //새 차의 정보를 등록하는 메소드
17     public void regCarInfo(){
18         System.out.println("렌트카 정보를 등록합니다.");
19     }
20
21     //차의 정보를 수정하는 메소드
22     public void modCarInfo(){
23         System.out.println("렌트카 정보를 수정합니다.");
24     }
25
26     //차의 정보를 삭제하는 메소드
27     public void delCarInfo(){
28         System.out.println("렌트카 정보를 삭제합니다.");
29     }
30 }
```

- 예약 클래스

```
1 package ex1.reserve;
2
3 public class Reserve {
4     String resCarNumber;
5     String resDate;
6     String useBeginDate;
7     String returnDate;
8
9     //차를 예약하는 기능을 하는 메소드
10    public void reserveCar(){
11        System.out.println("차를 예약합니다.");
12    }
13
14    //차 예약정보를 수정하는 메소드
15    public void modReserveInfo(){
16        System.out.println(" 예약정보를 수정합니다..");
17    }
18
19    //차 예약 정보를 취소하는 메소드
20    public void cancelReserveInfo(){
21        System.out.println(" 예약을 취소합니다..");
22    }
23 }
24
```

- 실행 클래스

```
1 package ex1;
2
3 import ex1.member.Member;
4 import ex1.car.Car;
5 import ex1.reserve.Reserve;
6
7 public class RentTest {
8     public static void main(String[] args){
9         //회원 가입하기
10        Member member=new Member();
11        member.regMember();
12
13        //렌트카를 조회한다.
14        Car car =new Car();
15        car.checkCarInfo();
16
17        //예약을 한다.
18        Reserve reserve=new Reserve();
19        reserve.reserveCar();
20    }
21 }
```