

능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Park Ju Byeong



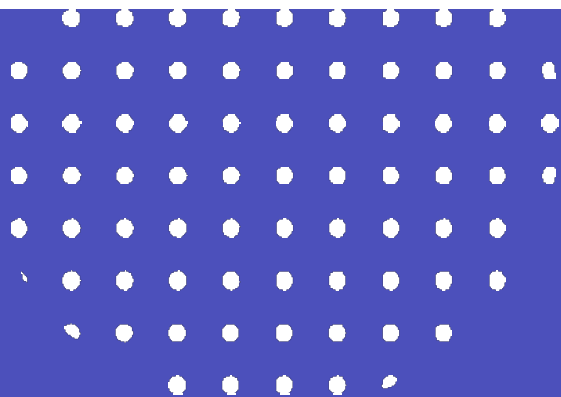
Part08 생성자

01 생성자

02 this

03 멤버변수 초기화

04 실습 문제



01

생성자



생성자

- 1.객체 생성시 **최초 1번만 실행**되는 특별한 메서드
- 2.클래스와 이름이 같아야 한다(대소문자 구분)
- 3.리턴 타입이 없어야 한다(void가 아님)
- 4.객체는 무조건 생성자를 통해서 생성된다.

```
public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;
    Student()
    {
    }
    Student(String name)
    {
        this.name = name;
    }
    Student(String name, int kor, int eng,int math)
    {
        this.name =name;
        this.kor =kor;
        this.eng = eng;
        this.math=math;
    }
}
```

생성자의 사용방법

```
Student student1 = new Student();  
student3.name = "박주병";  
student3.kor = 30;  
student3.eng = 50;  
student3.math = 20;
```



```
Student student1 = new Student();  
Student student2 = new Student("홍길동");  
Student student3 = new Student("박주병", 30, 50, 20);
```

코드가 간결해진다.

지금까지 클래스를 만들면서 생성자를 만든적이 없는데?

기본 생성자

1. 생성자가 하나도 없다면 자동으로 만들어준다.
2. 명시적으로 생성자가 한 개라도 선언이 되었다면 만들어지지 않는다.

```
public class Student
{
    String name;
    int kor;
    int eng;
    int math;

    Student()
    {
    }

    Student(String name1)
    {
        name = name1;
    }

    Student(String name1, int kor1, int eng1, int math1)
    {
        name = name1;
        kor = kor1;
        eng = eng1;
        math = math1;
    }
}
```

```
Student student1 = new Student();
```

```

public class Student
{
    String name;
    int kor;
    int eng;
    int math;

    Student(String name1)
    {
        name = name1;
    }

    Student(String name1, int kor1, int eng1, int math1)
    {
        name = name1;
        kor = kor1;
        eng = eng1;
        math = math1;
    }
}

```

명시적으로 생성자를 만들었기
때문에 기본생성자는 자동으로
만들어지지 않는다.

```

Student student1 = new Student();
Student student2 = new Student("홍길동");
Student student3 = new Student("박주병", 30, 50, 20);

```

기본 생성자가 없어 ERROR!

Park Ju Byeong

생성자 오버로딩

```
public class Student
{
    String name;
    int kor;
    int eng;
    int math;

    Student(String name1)
    {
        name = name1;
    }

    Student(String name1, int kor1, int eng1, int math1)
    {
        name = name1;
        kor = kor1;
        eng = eng1;
        math = math1;
    }
}
```

→ 생성자는 무조건 클래스 이름
이기때문에 2개이상이라면
무조건 오버로딩에 해당된다.

생성자를 오버로딩 했을때
문제점이 무엇인가?

```

class Student
{
    String name;
    int age;

    Student(String name1)
    {
        if(name1.length() < 10)
            name = name1;
        else
            System.out.println("10글자 이상은 안됩니다.");
    }

    Student(String name1, int age1)
    {
        if(name1.length() < 10)
            name = name1;
        else
            System.out.println("10글자 이상은 안됩니다.");

        age = age1;
    }
}

```

같은 필터 기능을 중복으로 써야 한다.

다른 생성자를 재활용 할순 없을까?

생성자에서 다른 생성자 호출

```
class Student
{
    String name;
    int age;

    Student(String name1)
    {
        if(name1.length() < 10)
            name = name1;
        else
            System.out.println("10글자 이상은 안됩니다.");
    }

    Student(String name1, int age1)
    {
        Student(name1);
        age = age1;
    }
}
```

매개변수1개 생성자를 재활용하면 필터 기능을 다시 안만들어도 된다.

근데 왜 안될까...?

클래스 내부에서 생성자를 호출하려면 에러가 발생한다.

생성자에서 다른 생성자 호출

```
class Student
{
    String name;
    int age;

    Student(String name1)
    {
        if(name1.length() < 10)
            name = name1;
        else
            System.out.println("10글자 이상은 안됩니다.");
    }

    Student(String name1, int age1)
    {
        this(name1);

        age = age1;
    }
}
```

1. 메서드 이름을 this로 해야 한다.
2. 생성자의 가장 첫줄에 작성해야 한다.

Student(name1)이 아닌
this로 호출을 해야 한다.

```
Student()  
{  
    System.out.println("생성자 호출");  
    this("홍길동");  
}
```

가장 첫줄에 작성해야 한다.

왜 첫줄에 작성해야 할까?

```
Student()  
{  
    name = "김길동";  
    this("홍길동");  
}
```

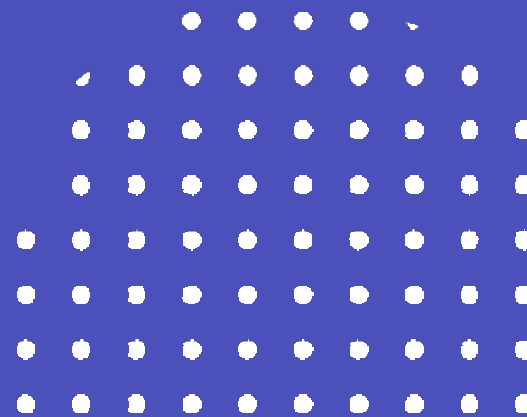
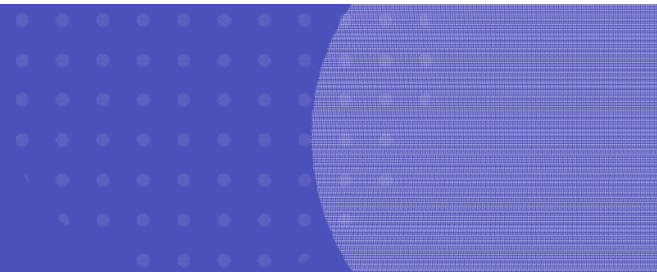
→ 의미가 없는 코드가 된다.

```
Student()  
{  
    System.out.println("생성자 호출");  
    this("홍길동");  
}
```

→ 기능적으로는 문제 없지만 프로그래밍 언어를 만드는 입장에서 컴파일 과정이 복잡해 진다. 따라서 아예 못쓰게 막았다.

—• 02

this



왜 생성자의 이름을 **this**로 바꿔야 할까?

```
Student()
{
    Student("박주병");
}

Student(String name1)
{
    name = name1;
}
```



이렇게 하면?

```
Student()
{
    new Student("박주병");
}

Student(String name1)
{
    name = name1;
}
```



```

public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;

    Student ()
    {
        new Student ("홍길동") ;
    }

    Student (String name)
    {
        this.name = name;
    }
}

```

기본 생성자를 통해 객체 생성하면 기본 이름을
홍길동으로 할거야!

```

public class Main {

    public static void main(String[] args) {

        Student s1 = new Student();

        System.out.println(s1.name);

    }
}

```

Problem @ Java... Decia...
 <terminated> Main (12) [Java Applicati
 null

기본 생성자 내부에서는 name을
초기화하는 생성자를 다시 호출하여
name에는 홍길동이 들어가 있겠지?

Park Ju Byeong

```

public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;

    Student ()
    {
        new Student ("홍길동") ;
    }

    Student (String name)
    {
        this.name = name ;
    }
}

```

객체의 주소를 어디에도 저장하지 않고 그냥 버렸다.

```

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student();
        System.out.println(s1.name);
    }
}

```

객체 생성

객체 생성

주소	이름
0x470b	홍길동
0x470c	0
0x470d	0
0x470e	0

주소	이름
0x000b	
0x000c	0
0x000d	0
0x000e	0

```

public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;

    Student ()
    {
        Student ("홍길동") ;
    }

    Student (String name)
    {
        this.name = name;
    }
}

```

```

public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;

    Student ()
    {
        new Student ("홍길동") ;
    }

    Student (String name)
    {
        this.name = name;
    }
}

```

하려는 의도가 전혀 다르지만
코드가 유사하다.
이름을 다르게 가는게
명확하다

this 변수

```
class Student
{
    String name;
    int age;

    Student(String name1)
    {
        if(name1.length() < 10)
            name = name1;
        else
            System.out.println("10글자 이상은 안됩니다.");
    }

    Student(String name1, int age1)
    {
        this(name1);
        age = age1;
    }
}
```

→ 생성자를 호출

```
public static void main(String[] args) {
    Student s1 = new Student();
    System.out.println(s1);
}

class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;
    static int count;

    {
        count++;
    }

    Student()
    {
        System.out.println(this);
    }

    Student(String name)
    {
        this.name = name;
    }
}
```

→ this 변수

this 변수

1. 클래스 생성시 자동으로 만들어지는 멤버변수
2. 객체 자신의 주소를 가지고 있다.

```
public class Bird {
```

```
    void fly()
```

```
    {
```

```
        System.out.println(this);
```

```
    }
```

→ 만들지 않아도 이미 존재하는
멤버변수이다.

```
    void eat()
```

```
    {
```

```
        System.out.println(this);
```

```
    }
```

```
}
```

→ 멤버변수이기에 클래스 내부
어디서든 사용가능하다.

this변수

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Bird bird1 = new Bird();  
        System.out.println("bird1 참조변수: "+bird1);  
  
        bird1.eat();  
    }  
}
```

```
public class Bird {  
  
    void fly()  
    {  
        System.out.println(this);  
    }  
  
    void eat()  
    {  
        System.out.println("this: "+this);  
    }  
}
```

Problems Javadoc Search Console X Git Staging
<terminated> Main (9) [Java Application] C:\Users\Wzest1\p2\pool\plu
bird1 참조변수: joo.강의8.Bird@6f2b958e
this: joo.강의8.Bird@6f2b958e

bird1과 this는 같은 주소를 가지고 있다.

```
public class Bird {
```

```
    void fly()
```

```
    {  
        System.out.println(this);  
    }
```

```
    void eat()
```

```
    {  
        System.out.println("this: "+this);  
    }
```

```
public class Main {
```

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

```
        Bird bird1 = new Bird();
```

```
        Bird bird2 = new Bird();
```

```
        Bird bird3 = new Bird();
```

```
        System.out.println("bird1 참조변수: "+bird1);
```

```
        System.out.println("bird2 참조변수: "+bird2);
```

```
        System.out.println("bird3 참조변수: "+bird3);
```

```
        bird1.eat();
```

```
        bird2.eat();
```

```
        bird3.eat();
```

```
    }
```

```
bird1 참조변수: joo.강의8.Bird@6f2b958e
```

```
bird2 참조변수: joo.강의8.Bird@5e91993f
```

```
bird3 참조변수: joo.강의8.Bird@1c4af82c
```

```
this: joo.강의8.Bird@6f2b958e
```

```
this: joo.강의8.Bird@5e91993f
```

```
this: joo.강의8.Bird@1c4af82c
```

this는 호출하는 객체의 주소를 가진다.

왜 필요할까?


```
public class Bird {
```

```
    int age;
```

```
    Bird()
```

```
    {
```

```
        age = 1;
```

```
    }
```

```
    Bird(int age)
```

```
    {
```

```
        age = 1;
```

```
    }
```



```
public class Bird {
```

```
    int age;
```

```
    Bird()
```

```
    {
```

```
        age = 1;
```

```
    }
```

```
    Bird(int age)
```

```
    {
```

```
        this.age = 1;
```


```
    }
```

지역변수가 우선시되어
멤버변수 age에 값을 저장할수
없다.

클래스밖에서 사용하듯이 참조변수를
이용해 접근하면 무조건 멤버변수이다.
클래스 밖에서는 지역변수를 사용할수 없다.

퀴즈

```
static void test()  
{  
    this.eng = 30;  
    this.getTotal();  
}
```



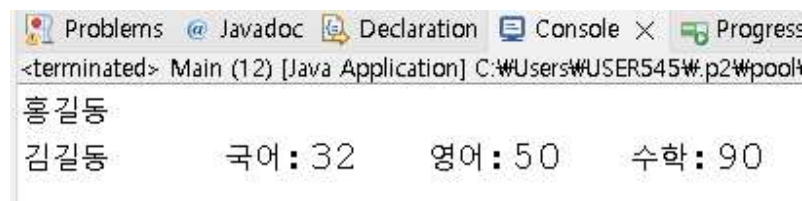
static은 객체 생성이전에도 사용할 수 있다.
따라서 객체를 가리키는 this는 사용 할 수 없다.

연습문제1

1-1 Student 클래스를 만들고 사용해보자(normal)

- 멤버변수 : String name, int kor, int math, int eng
- 생성자를 오버로딩 하여 아래의 예시처럼 3개의 생성자를 만들자.

```
public class Main {  
  
    public static void main(String[] args) {  
  
        Student student1 = new Student();  
        Student student2 = new Student("홍길동");  
        Student student3 = new Student("김길동", 32, 50, 90);  
  
        System.out.println(student2.name);  
        System.out.println(student3.name+"\t"  
            +"국어:"+student3.kor+"\t"  
            +"영어:"+student3.eng+"\t"  
            +"수학:"+student3.math+"\t"  
        );  
    }  
}
```



```
<terminated> Main (12) [Java Application] C:\Users\WUSER545\p2\pool4  
홍길동  
김길동      국어: 32      영어: 50      수학: 90
```

해설

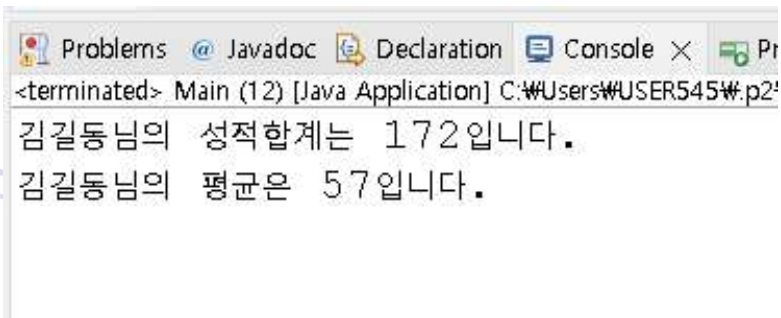
Park Ju Byeong

Park

1-2 Student클래스에 성적의 합계와 평균을 구하는 기능을 추가하자(normal)

- 멤버메서드 : getTotal() 성적의 합계를 반환한다.
getAverage() 성적의 평균을 반환한다.

```
Student student1 = new Student();  
Student student2 = new Student("홍길동");  
Student student3 = new Student("김길동", 32, 50, 90);  
  
System.out.println(student3.name+"님의 성적합계는 "  
                    + student3.getTotal()+"입니다.");  
System.out.println(student3.name+"님의 평균은 "  
                    +student3.getAverage()+"입니다.");
```



해설

Park Ju Byeong

Park Ju Byeong

1-3 Student클래스에 객체가 생성 될 때마다 객체의 카운트를 올려보자(hard)

- 멤버변수 count 추가(현재 생성된 객체의 개수를 저장한다.)

힌트

1. count 변수는 객체가 생성될때마다 1씩 증가하여 현재 객체가 몇 개 생성되었는지를 저장하고 있어야 한다.
2. static을 활용해야만 가능하다.
3. 객체가 생성될때마다 특정 작업을 하고 싶다면 생성자 내부에서 해야 한다.

```
Student student1 = new Student();  
Student student2 = new Student("홍길동");  
Student student3 = new Student("김길동", 32, 50, 90);
```

```
System.out.println("현재 생성된 학생 객체는:" + Student.count + "개 입니다.");
```



The screenshot shows an IDE window with tabs for Problems, Javadoc, Declaration, and Console. The Console tab is active, displaying the output of a Java application. The text in the console is: "현재 생성된 학생 객체는: 3개 입니다." (Currently generated student objects: 3 objects).

해설

Park
~yeong

Park ~

- void showState() 메서드를 만들어 학생 정보를 출력하자.
- Student[] 길이10의 배열을 만들어 반복문을 이용하여 객체를 생성하자
앞서 만든 생성자를 이용해 이름과 성적을 초기화 한다.(이름은 모두 다르게)
성적은 Math.random() 을 활용하여 0~100 랜덤으로 입력한다.

이름	국어	영어	수학	합계	평균
학생0	42	37	35	114	38
학생1	25	3	99	127	42
학생2	32	29	57	118	39
학생3	46	30	67	143	47
학생4	63	29	35	127	42
학생5	66	0	92	158	52
학생6	82	25	19	126	42
학생7	56	61	45	162	54
학생8	10	75	56	141	47
학생9	90	99	36	225	75

해설

Park Ju Byeong }

Park J

1-5 Student 클래스의 기능을 추가하자.(expert)

- static Student getMinAvg(Student[]) 메서드를 만들자
매개변수로 학생 리스트를 받아서 그중 평균점수가 가장 낮은
객체를 반환한다.(동점자가 있다면 먼저 찾은 객체를 반환한다)

```
final int LIST_CNT = 10;
```

```
Student[] studentList = new Student[LIST_CNT];
```

```
for(int i =0 ; i<LIST_CNT ; i++)
```

```
{
```

객체 생성 및 이름, 성적 입력

```
}
```

```
Student lastStudent = Student.getMinAvg(studentList);
```


```
System.out.println("성적이 가장 낮은 학생은?");
```

```
lastStudent.showState();
```

<terminated> Main (9) [Java Application] C:\Users\wzest1\p2\pool\plugins\org.eclipse.justj.openjdk

이름:학생0	국어:74	영어:72	수학:16	합계:162	평균:54
이름:학생1	국어:48	영어:37	수학:84	합계:169	평균:56
이름:학생2	국어:93	영어:64	수학:18	합계:175	평균:58
이름:학생3	국어:85	영어:24	수학:47	합계:156	평균:52
이름:학생4	국어:13	영어:92	수학:38	합계:143	평균:47
이름:학생5	국어:9	영어:62	수학:64	합계:135	평균:45
이름:학생6	국어:8	영어:75	수학:26	합계:109	평균:36
이름:학생7	국어:30	영어:20	수학:20	합계:70	평균:23
이름:학생8	국어:27	영어:80	수학:41	합계:148	평균:49
이름:학생9	국어:3	영어:11	수학:95	합계:109	평균:36
성적이 가장 낮은 학생은?					
이름:학생7	국어:30	영어:20	수학:20	합계:70	평균:23

해설



getMinAvg메서드는 특정 객체의 기능이 아니라 Student객체의 공통적인 기능이다. 따라서 객체 생성없이 사용가능하도록 static 메서드로 만든다.

1-6 Student 클래스의 기능을 추가하자.(expert)

- 길이10의 Student[] studentList 배열을 만든후 랜덤으로 성적을 입력해보자.
이름은 모두 다르게 설정한다.
- 길이10의 또다른 Student[] sortList 배열을 만든후 studentList의 객체들을
오름차순 대로 sortList 배열에 넣어보자
studentList(정렬전) -> sortList(정렬후)
- 정렬 방법은 선택정렬을 구현해본다.(필요한 메서드가 있다면 만들어 써보자)

힌트: 앞서 만든 getMinAvg() 를활용하여 성적이 가장 낮은 학생들을 빼와서
sortList에 넣으면 된다.

-----정렬 전-----					
이름:학생0	국어:90	영어:97	수학:74	합계:261	평균:87
이름:학생1	국어:14	영어:53	수학:71	합계:138	평균:46
이름:학생2	국어:18	영어:88	수학:43	합계:149	평균:49
이름:학생3	국어:91	영어:26	수학:18	합계:135	평균:45
이름:학생4	국어:58	영어:68	수학:87	합계:213	평균:71
이름:학생5	국어:57	영어:100	수학:96	합계:253	평균:84
이름:학생6	국어:30	영어:67	수학:82	합계:179	평균:59
이름:학생7	국어:66	영어:73	수학:71	합계:210	평균:70
이름:학생8	국어:2	영어:82	수학:94	합계:178	평균:59
이름:학생9	국어:65	영어:96	수학:80	합계:241	평균:80
-----정렬 완료-----					
이름:학생3	국어:91	영어:26	수학:18	합계:135	평균:45
이름:학생1	국어:14	영어:53	수학:71	합계:138	평균:46
이름:학생2	국어:18	영어:88	수학:43	합계:149	평균:49
이름:학생6	국어:30	영어:67	수학:82	합계:179	평균:59
이름:학생8	국어:2	영어:82	수학:94	합계:178	평균:59
이름:학생7	국어:66	영어:73	수학:71	합계:210	평균:70
이름:학생4	국어:58	영어:68	수학:87	합계:213	평균:71
이름:학생9	국어:65	영어:96	수학:80	합계:241	평균:80
이름:학생5	국어:57	영어:100	수학:96	합계:253	평균:84
이름:학생0	국어:90	영어:97	수학:74	합계:261	평균:87

studentList

객체주소	이름
0X000A	학생1
0X000B	학생2
0X000C	학생3
0X000D	학생4
0X000E	학생5

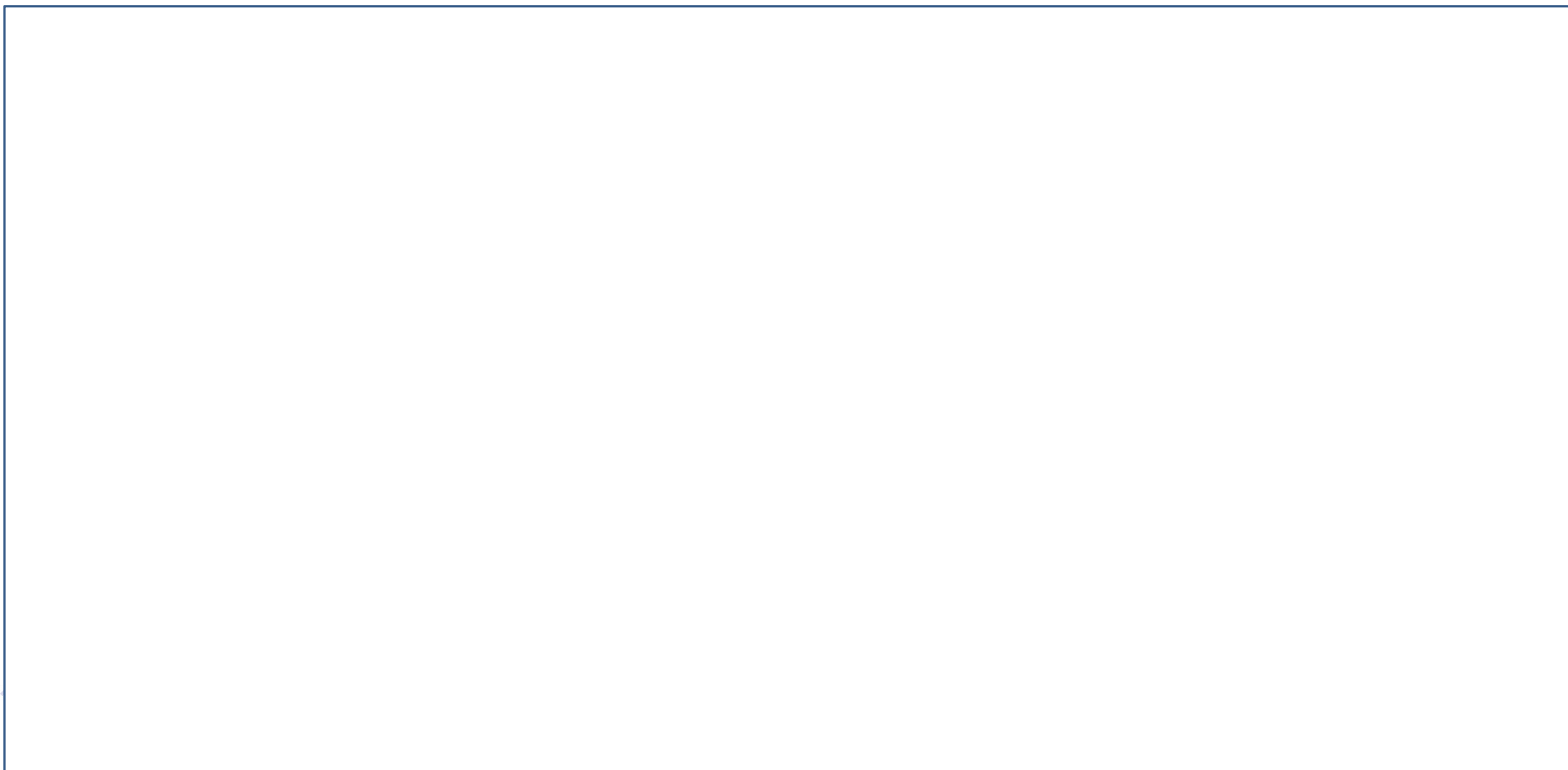
sortList

객체주소	이름
null	
null	
null	
null	
null	



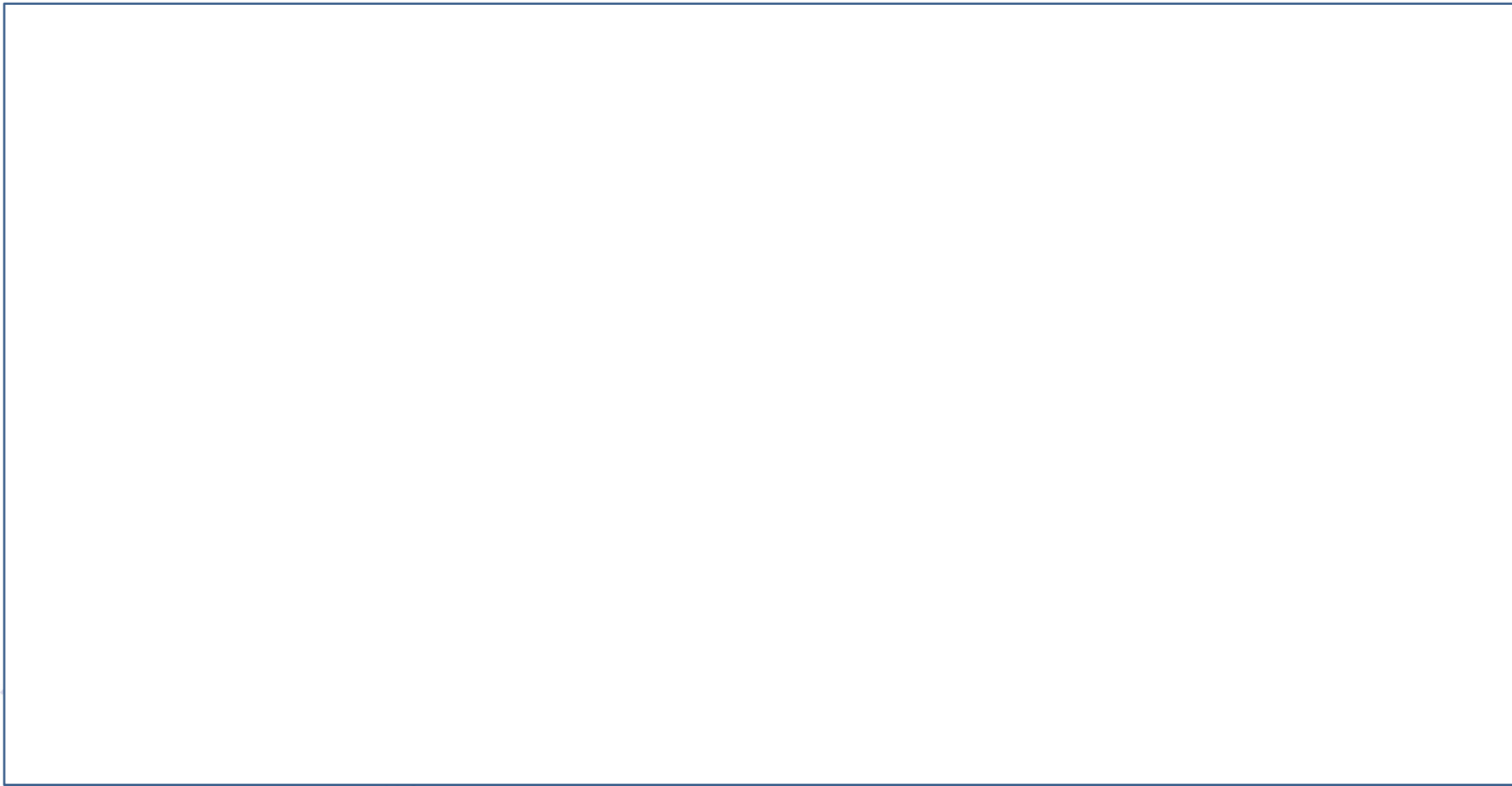
Byeong

Park Ju



Byeong

Park Ju



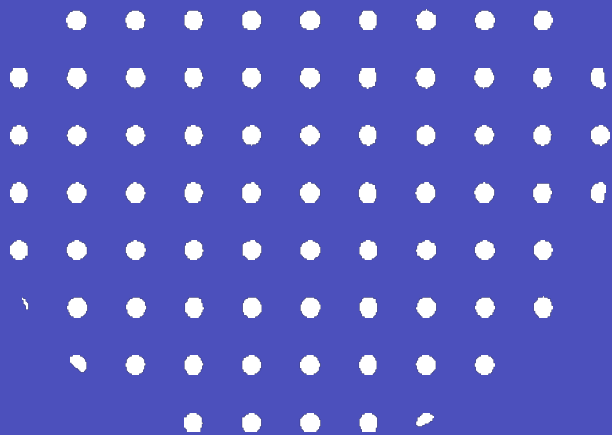
2

ong

Park

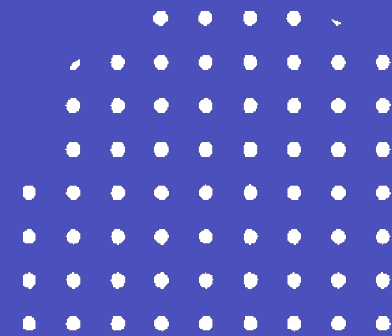
메서드는 1개의 기능만을 담당하는것이 베스트이다.

모듈화



03

멤버변수 초기화



```

class test{
    int a;
    void method()
    {
        int b;
        System.out.println(a);
        System.out.println(b);
    }
}

```

멤버변수는 기본값 초기화

지역변수는 쓰레기값

자료형	기본값
boolean	false
char	'\u0000'
참조형	null
그외	0

지역변수는 반드시 초기화를 해야 사용할수 있다.

멤버변수 초기화 방법 3가지

명시적 초기화

생성자

초기화 블록

명시적 초기화

```
2  
3 public class Student  
4 {  
5     String name = "박주병";  
6     int kor = 30;  
7     int eng = 50;  
8     int math = 20;  
9  
0  
1 Student()  
2 {  
3     System.out.println("this: " + this);  
4 }  
5
```

값을 하드코딩으로 바로 초기화 할경우 좋다.

초기화 블록

```
3 public class Student
4 {
5     String name ;
6     int kor ;
7     int eng ;
8     int math ;
9     static int number;
10
11     Student()
12     {
13         number++;
14     }
15
16     Student(String name)
17     {
18         number++;
19         this.name = name;
20     }
21
22     Student(String name, int kor, int eng, int math)
23     {
24         number++;
25         this.name = name;
26         this.kor = kor;
27         this.eng = eng;
28         this.math = math;
29     }
30 }
```



```
3 public class Student
4 {
5     String name ;
6     int kor ;
7     int eng ;
8     int math ;
9     static int number;
10
11     static { System.out.println("클래스 초기화 블록");}
12
13     {
14         System.out.println("인스턴스 초기화 블록");
15         number++;
16     }
17
18     Student()
19     {
20         System.out.println("생성자");
21     }
22
23     Student(String name)
24     {
25         this.name = name;
26     }
27
28     Student(String name, int kor, int eng, int math)
29     {
30         this.name = name;
31     }
32 }
```

클래스 초기화 블록

클래스가 메모리에 처음 로딩될때 한번만 실행
(프로그램이 실행될때!)

인스턴스 초기화 블록

인스턴스가 만들어질때 마다 생성자보다 먼저 실행

```
Student student1 = new Student();  
Student student2 = new Student();  
Student student3 = new Student();
```

```
<terminated> main [Java Applicatio  
클래스 초기화 블록  
인스턴스 초기화 블록  
생성자  
인스턴스 초기화 블록  
생성자  
인스턴스 초기화 블록  
생성자
```



```
public class Student
{
    String name ;
    int kor ;
    int eng ;
    int math ;
    static int number;

    static {
        System.out.println("클래스 초기화 블록");

        name ="김길동";
    }
}
```

→ 클래스 초기화 블록은 객체 생성보다 이전에 실행되므로 인스턴스 변수는 사용할수 없다.

초기화 블록은 왜 필요할까?

```
public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;
```

```
    static int count;
```

```
    Student()
    {
```

```
        count++;
```

→

학생 객체가 만들어 질때마다 카운트를 올려 객체가
생성된 개수를 관리한다.

```
    Student(String name)
    {
```

```
        this.name = name;
```

```
        count++;
```

```
    }
```

```
    Student(String name, int kor, int eng, int math)
    {
```

```
        this.name = name;
```

```
        this.kor = kor;
```

```
        this.eng = eng;
```

```
        this.math = math;
```

```
        count++;
```

```
    }
```

모든 생성자에 다 작성해줘야 된다.

```
public class Student
{
    public String name ;
    int kor ;
    int eng ;
    int math ;
```

```
    static int count;
```

```
    {
        count++;
    }
```

초기화블럭을 이용해 생성자에서 공통적으로
수행해야 하는 기능을 한번만 작성해주면 된다.

```
Student()
{
```

```
}
```

```
Student(String name)
{
```

```
    this.name = name;
```

```
}
```

```
Student(String name, int kor, int eng, int math)
{
```

```
    this.name = name;
```

```
    this.kor = kor;
```

```
    this.eng = eng;
```

```
    this.math = math;
```

```
}
```



THANK YOU



강사 박주병