

능동적 사고 방식의

java

강사 박주병

Park Ju Byeong

Park Ju Byeong

Part08 생성자

01 생성자

02 this

03 멤버변수 초기화

04 실습 문제

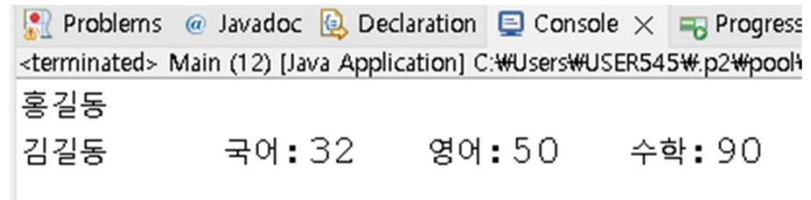
—• 02
this

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"  
        );  
    }  
}
```



Problems @ Javadoc Declaration Console × Progress
<terminated> Main (12) [Java Application] C:\Users\WUSER545W\p2\pool4
홍길동
김길동 국어: 32 영어: 50 수학: 90

해설

```
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;

    }
}
```

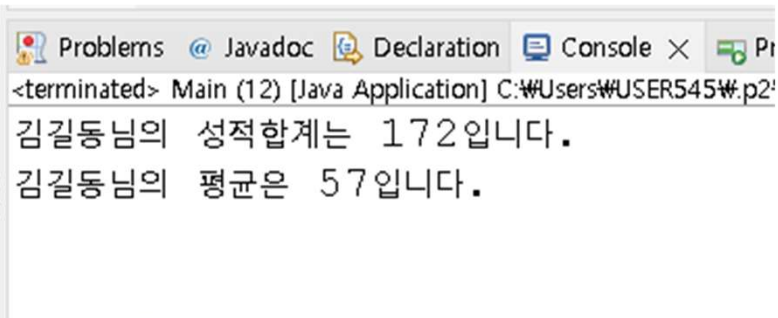
→ 생성자를 3개 제공해준다.

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

```
int getTotal()  
{  
    return kor+eng+math;  
}
```

```
int getAverage()  
{  
    return getTotal()/3;  
}
```

합계를 직접 구하는것보다 기존에 만들어 놓은 메서드를 활용하는것이 좋다.

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

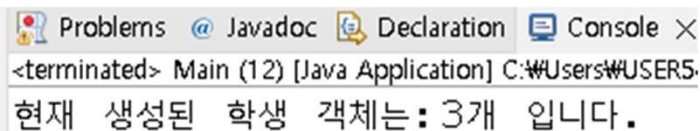
- 멤버변수 count 추가(현재 생성된 객체의 개수를 저장한다.)

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 interface 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).

```
<terminated> Main (12) [Java Application] C:\Users\USER5.  
현재 생성된 학생 객체는: 3개 입니다.
```


해설

```
Student()
```

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

```
count++;
```

```
Student(String name)
```

```
{  
    this.name = name;  
    count++;  
}
```

```
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++;  
}
```

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

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

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

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

```
count++;
```

객체 생성을 위해선 반드시 생성자가
실행되어야 한다.
만들어놓은 모든 생성자에 count를 증가
시키도록 하면
객체 생성시 반드시 카운트가 1씩 증가 한다.

1-4 Student 클래스 기능을 추가하자.(hard)

- 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.

```
public static void main(String[] args) {  
  
    final int LIST_CNT = 10;  
  
    Student[] studentList = new Student[LIST_CNT];  
  
    for(int i =0 ; i<LIST_CNT ; i++)  
    {  
  
  
  
  
  
  
  
  
  
    }  
}
```

```
<terminated> Main (9) [Java Application] C:\Users\zest1\p2\pool\plugins\org.eclipse.justj.openjdk.hot  
이름:학생0      국어:42   영어:37   수학:35   합계:114   평균:38  
이름:학생1      국어:25   영어:3    수학:99   합계:127   평균:42  
이름:학생2      국어:32   영어:29   수학:57   합계:118   평균:39  
이름:학생3      국어:46   영어:30   수학:67   합계:143   평균:47  
이름:학생4      국어:63   영어:29   수학:35   합계:127   평균:42  
이름:학생5      국어:66   영어:0    수학:92   합계:158   평균:52  
이름:학생6      국어:82   영어:25   수학:19   합계:126   평균:42  
이름:학생7      국어:56   영어:61   수학:45   합계:162   평균:54  
이름:학생8      국어:10   영어:75   수학:56   합계:141   평균:47  
이름:학생9      국어:90   영어:99   수학:36   합계:225   평균:75
```

해설

```
void showState()  
{  
    System.out.println("이름:" + name  
        + "\t국어:" + kor  
        + "\t영어:" + eng  
        + "\t수학:" + math  
        + "\t합계:" + getTotal()  
        + "\t평균:" + getAverage()  
    );  
}
```

만들어둔 메서드를 적극
활용한다.

```
final int LIST_CNT = 10;  
Student[] studentList = new Student[LIST_CNT];  
for(int i = 0 ; i < LIST_CNT ; i++)  
{  
    Student std = new Student("학생" + i  
        , (int) (Math.random() * 101)  
        , (int) (Math.random() * 101)  
        , (int) (Math.random() * 101));  
  
    std.showState();  
    studentList[i] = std;  
}
```

객체배열을 만든다.(배열을
만든것이지 Student객체를
만든게 아니다.)

요소마다 객체를 만들어
넣는다.

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.

```
static Student getMinAvg(Student[] list)
{
```

```
    int minScore = 101;
```

```
    Student target = null;
```

```
    for(int i = 0 ; i < list.length; i++)
```

```
    {
```

```
        if(minScore > list[i].getAverage())
```

```
        {
```

```
            minScore = list[i].getAverage();
```

```
            target = list[i];
```

```
        }
```

```
    }
```

```
    return target;
```

```
}
```

현재 꼴등보다 성적이 더 낮은 학생이 나오면

새로 발견한 꼴등학생의 성적을 저장한다.

꼴등 학생의 객체를 저장해둔다.

반복문이 모두 수행되면 target 변수는 성적이 가장 낮은 학생 객체를 가리키고 있다.

Park Ju Byeong

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() 를 활용하여 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	

```

static Student getMinAvg(Student[] list)
{
    Student target = null;

    for(Student student : list)
    {
        if(student == null)
            continue;

        if(target == null)
            target = student;

        if(target.getAverage() > student.getAverage())
            target = student;
    }
    return target;
}

```

```

static int getIndex(Student[] list, String name)
{
    for(int i = 0 ; i < list.length ; i++)
    {
        if(list[i] == null)
            continue;

        //이름이 같다면
        if(list[i].name.equals(name))
            return i;
    }

    //찾지 못하면 -1을 반환한다.
    return -1;
}

```



```

for(int i =0 ; i<LIST_CNT ; i++)
{
    //성적이 가장 낮은 학생을 가져온다.
    Student LastStudent = Student.getMinAvg(studentList);

    //해당 학생의 이름으로 몇번 인덱스 인지 찾는다.
    int targetIndex = Student.getIndex(studentList, LastStudent.name);

    //해당 학생의 객체주소를 새로운 배열에 넣는다.
    sortList[i] = studentList[targetIndex];

    //기존 배열에 null을 대입하여 제거한다.(새로운 배열에서 해당 객체를 가리키고 있기에 문제 없다.)
    studentList[targetIndex] = null;
}

System.out.println("-----정렬 완료-----");
for(Student std : sortList)
    std.showState();

```

getMinAvg() 에서 리턴을 Student 객체 말고
index를 반환하면 getIndex 메서드를 만들 필요 없지 않을까?

```
for(int i =0 ; i<LIST_CNT ; i++)
{
    //성적이 가장 낮은 학생을 가져온다.
    Student LastStudent = Student.getMinAvg(studentList);

    //해당 학생의 이름으로 몇번 인덱스 인지 찾는다.
    int targetIndex = Student.getIndex(studentList, LastStudent.name);

    //해당 학생의 객체주소를 새로운 배열에 넣는다.
    sortList[i] = studentList[targetIndex];

    //기존 배열에 null을 대입하여 제거한다.(새로운 배열에서 해당 객체를 가리키고 있기에 문제 없다.)
    studentList[targetIndex] = null;
}

System.out.println("-----정렬 완료-----");
for(Student std : sortList)
    std.showState();
```

바로 인덱스를 반환하도록 만들어주면
여기가 필요없어질텐데???

메서드는 1개의 기능만을 담당하는것이 베스트이다.

모듈화



THANK YOU



강사 박주병