**자료구조 실습 보고서**

[제07주] 성적처리 : 재귀(Recursion)

**2021년 04월 19일**

**201702039 오명주**

1. **프로그램 설명서**
2. **프로그램의 전체 설계 구조**

* MVC (Model – View – Controller) 구조

Model : 프로그램이 “무엇”을 할 것인지 정의. 사용자의 요청에 맞는 알고리즘을 처리하고 DB와 상호작용하여 결과물을 산출하고 Controller에게 전달.

View : 화면에 무엇인가를 “보여주기 위한” 역할. 최종 사용자에게 “무엇”을 화면으로 보여줌.

Controller : 모델이 “어떻게” 처리할 지 알려주는 역할. 사용자로부터 입력을 받고 중개인 역할. Model과 View는 서로 직접 주고받을 수 없음. Controller을 통해 이야기함.

* 리스트 성능 비교 프로그램에서의 각 클래스 별 MVC 구조 역할

**Model** :

* UnsortedArrayList : ArrayList로 구현된 정렬되지 않은 리스트
* Ban : 학생 배열을 받을 학급을 의미하는 클래스. 학생 성적 통계를 담당한다.
* Student : 학생 객체 생성할 수 있는 클래스. 객체 비교를 위한 compareTo 함수가 존재한다.
* GradeCounter : 성적에 따른 학점을 정하고 학생 수를 관리한다.
* Iterator : 반복자 인터페이스. 배열에서 반복하여 처리할 때 사용한다.

**View** :

* AppView : 프로그램의 입/출력을 담당한다.

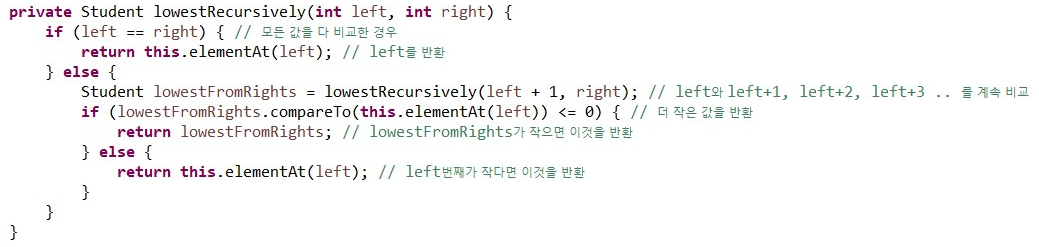
**Controller** :

* AppController : Model을 통해 생성된 결과물을 AppView를 통해 출력한다.

1. **함수 설명서**

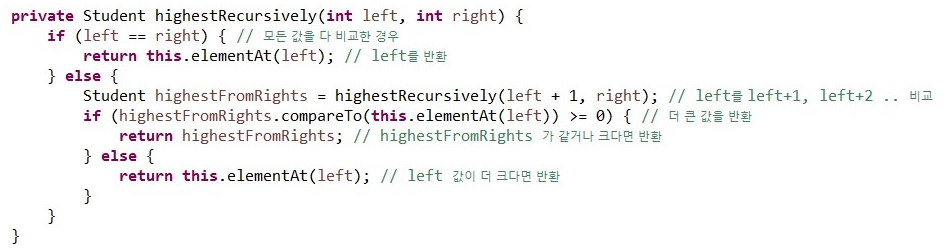
* 주요 알고리즘

1. 재귀를 이용하여 Lowest, Highest 값 반환



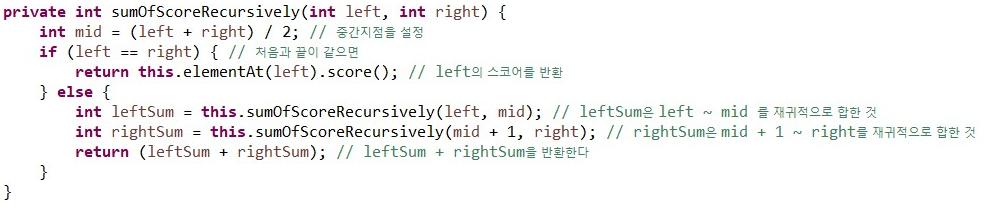
학생 성적 배열에서 가장 낮은 성적의 값을 출력할 때 사용하는 lowestRecursively 함수.

* 재귀를 이용하여 구현하였다.
* Left와 Right가 같다면 해당 element를 반환한다. (탈출 조건)
* 재귀를 통해 Right와 Right-1을 비교한 후 더 작은 값을 Right-2와 비교, … 이를 반복하여 Left 값과 마지막으로 비교하여 가장 작은 값을 반환 받게 된다.
* 결국 (Left)와 (나머지 중 가장 작은 값)을 비교한 결과 더 작은 값을 반환하는 것이다.



이는 성적 배열에서 가장 높은 성적을 구할 때도 동일하다. Right와 Right-1 인덱스의 값을 먼저 비교하여 큰 값을 Right-2와 비교, Right-3과 비교 …. 반복하여 Left 값과 비교하여 가장 큰 값을 반환한다.

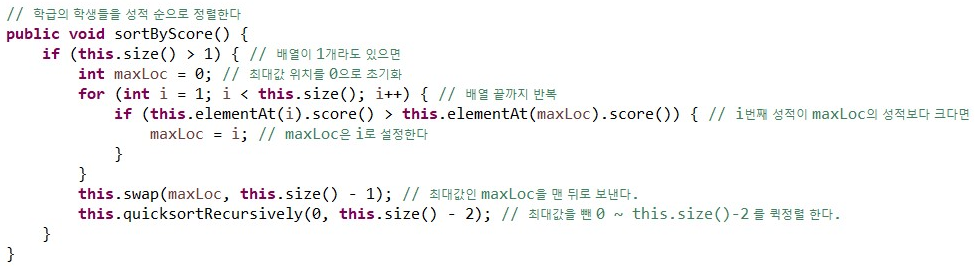
1. 재귀를 이용하여 Sum 값 반환



학생 성적 배열에서 각각의 성적에 대한 Sum 값을 구하여 반환하는 sumOfScoreRecursively 함수.

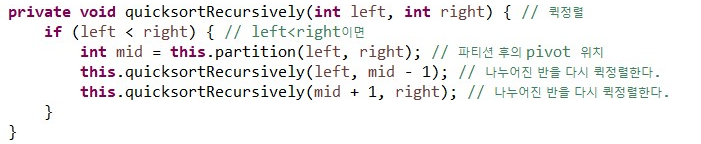
* 중간 위치인 mid를 설정한다.
* Left와 Right가 같다면 해당 score를 반환한다. (탈출 조건)
* 배열을 반으로 나누어 각각의 배열의 합을 구하는 방식이다. 재귀를 통해 배열을 쪼개어 하나의 배열이 될 때까지 반복하여 해당 값을 반환하여 Left 값과 Right 값을 더한다.
* mid를 중심으로 leftSum은 left-mid까지 값의 합을, rightSum은 (mid+1)-right까지 값의 합을 구하여 더하는 형태이다.

1. 재귀를 이용한 학생 성적 Sort

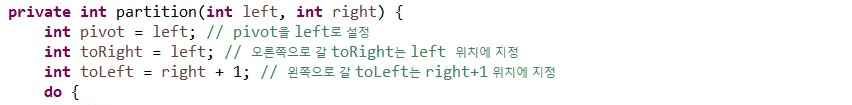


Quick Sort를 이용하여 학급의 학생들을 성적 순으로 정렬하는 sortByScore 함수.

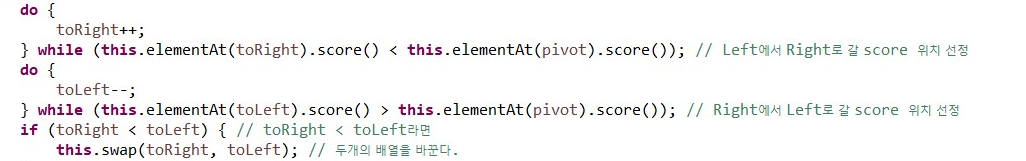
* 학생 배열에 성적이 하나라도 존재할 경우 수행한다.
* 최대값 위치를 찾아 가장 마지막으로 보낸다.
* 인덱스 0부터 size-2까지 퀵 정렬을 수행한다.



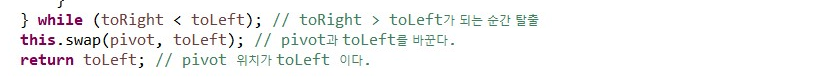
퀵 정렬은 다음과 같이 수행한다. 반을 나누어 퀵 정렬을 수행한다. Mid 변수에는 partition이 끝난 후 pivot의 위치가 반환된다.



Partition 함수를 확인하면 pivot을 기준으로 삼는데, 초기에는 left값으로 설정한다. 오른쪽으로 갈 toRigt는 Left 위치에 지정하고 왼쪽으로 갈 toLeft는 right+1 위치에 지정한다. (do-while을 통해 먼저 right--을 실행하므로 +1을 하여 위치를 설정한다)



* toRight 위치를 설정한다. pivot과 비교하여 pivot보다 작으면 반복해서 행하며 pivot보다 클 경우 반복문을 빠져나온다.
* toLeft 위치를 설정한다. pivot과 비교하여 pivot보다 크면 반복해서 행하며 pivot보다 작으면 반복문을 빠져나온다.
* swap 함수를 통해 두개의 값을 바꿔준다.



toRight 인덱스가 toLeft 인덱스보다 작아지면 (mid 부분에서 교차되면) 반복문을 빠져나온다. pivot을 중간위치로 설정해주고 pivot을 반환한다. 해당 함수는 mid를 기준으로 재귀적으로 반복되어 실행된다.

1. iterator을 이용한 학생 리스트 출력



학생 성적을 sort 하여 iterator을 이용해 순서대로 출력하는 showStudentSortedByScore 함수.

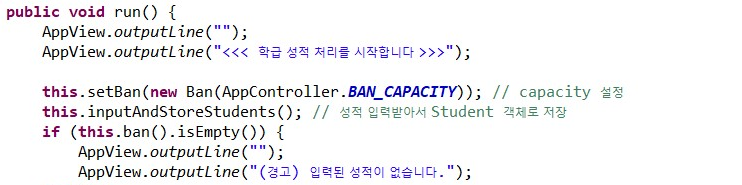
* iterator 객체를 생성하고 출력에 이용할 student 객체도 생성하여 초기화한다.
* hasNext를 가져와서 있는 경우 반복하여 student에 해당 score을 저장하여 AppView의 출력문을 통해 출력한다.

1. **종합 설명서**

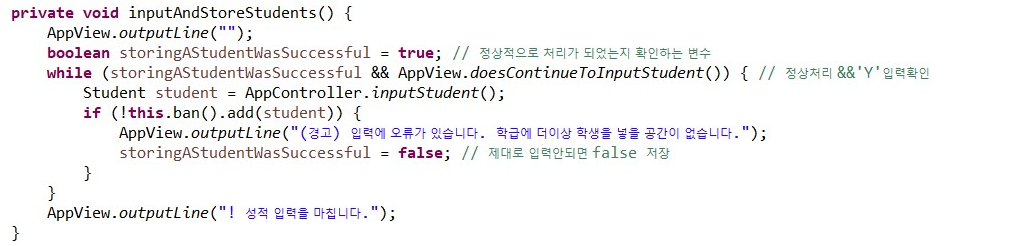
* 프로그램 실행 순서대로 설명해보자.



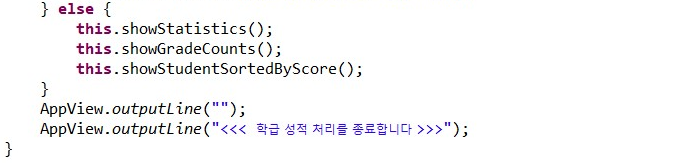
main에서 AppController 의 객체를 생성하여 run 한다. 프로그램 실행.



AppController 의 run 함수를 통해 capacity를 설정하고 inputAndStoreStudent 함수를 이용하여 사용자로부터 성적 입력 여부와, 학생 성적을 입력 받아 Student 객체에 저장한다. 입력한 성적이 없다면 없다는 경고를 출력한다.

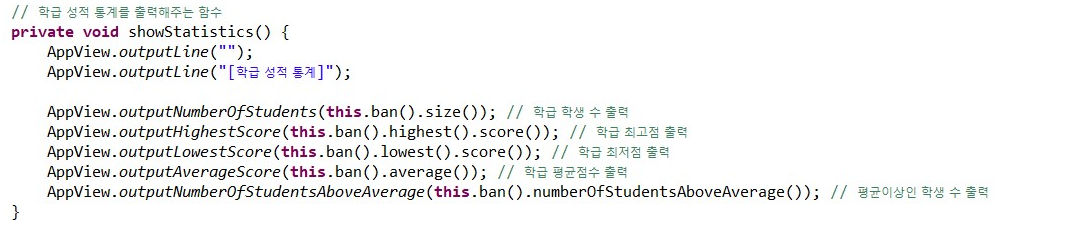


정상 처리가 되는지 여부를 storingAStudentWasSuccessful 변수에 Boolean 타입으로 저장하고 입력된 문자가 유효한지 확인한다. ‘Ban’에 저장공간이 존재하는지 확인한다. 유효한 문자이고 그 전 처리가 정상처리 되었으면 inputStudent 함수를 통해 정수를 입력 받아 Student 객체에 저장한다.



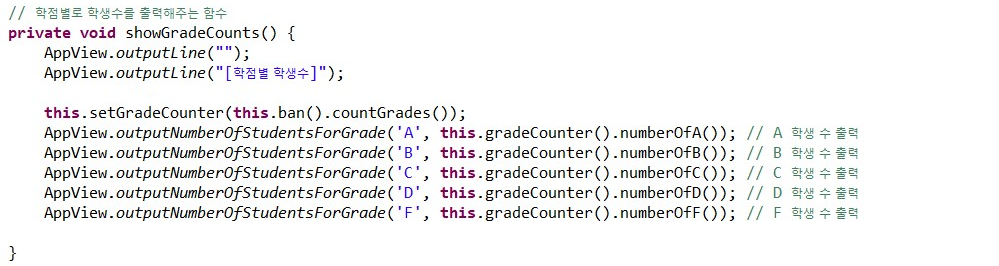
run 함수에서 프로그램 종료 시, 출력하는 함수들이다.

해당 함수들에 대한 설명은 하단에 존재.



학급 성적의 각 통계들을 출력하는 함수이다.

* size를 출력하여 총 학생 수를 출력한다.
* 재귀를 이용하여 highest, lowest 성적을 뽑아 출력한다.
* 평균 점수와 평균 이상의 학생 수를 출력한다.
* int와 double 타입을 활용한다.



학생 성적을 학점으로 변환하는 gradeCounter 함수를 이용하여 학점별로 학생 수를 출력하는 함수이다.

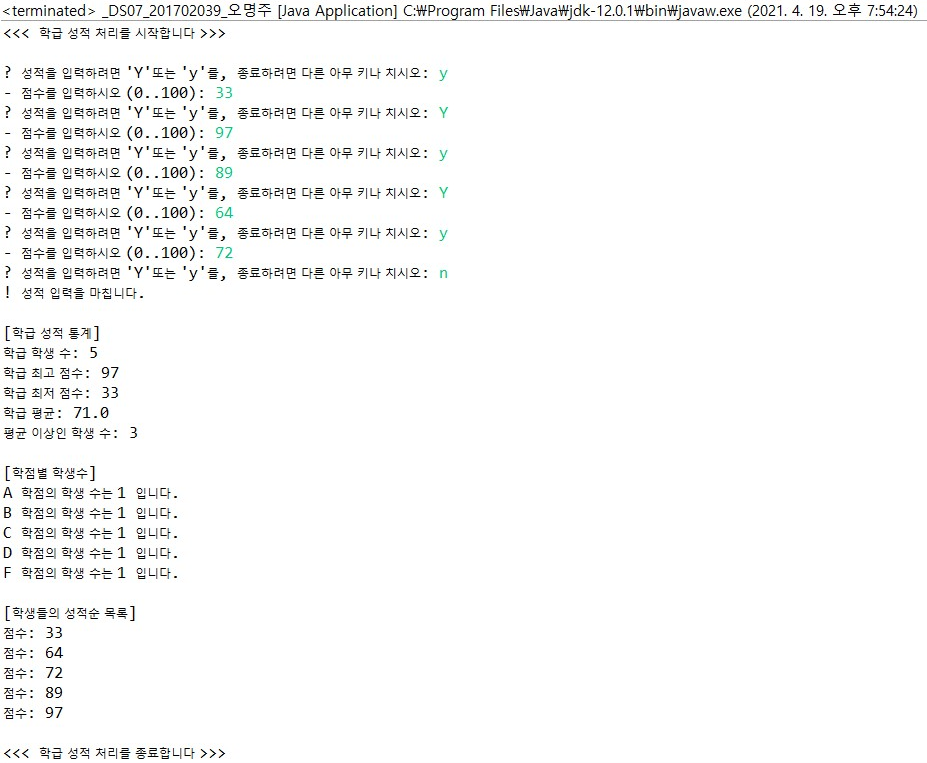
* 학생 성적을 순서대로 출력하는 함수에 대해서는 상단에 설명 되어있다.

1. **프로그램 장단점 / 특이점 분석**

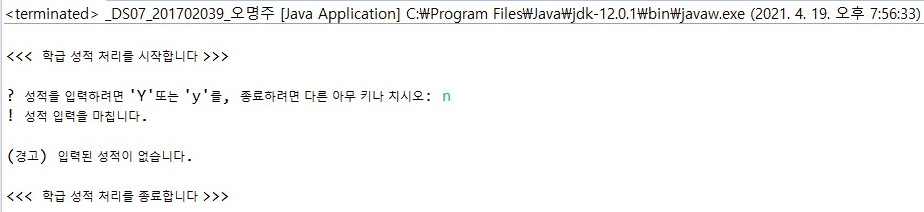
* **장점**
* MVC 모델을 이용하여 가독성과 생산성이 뛰어나다. 각 클래스, 함수의 역할이 분명해서 코드와 프로그램을 잘 이해할 수 있다.
* 설계가 잘 되어있다. 학번 등 학생 정보를 추가하여 더 입력 할 것이 있는 경우 프로그램에서 Student 클래스에 학번변수만 생성하여 사용하면 된다.
* 재귀를 사용하여 for, while문과 같은 반복문에 비해 코드가 간결하다. 다양한 방법을 통해 구현할 수 있었다.
* iterator을 이용한 반복자 개념을 활용할 수 있다. 반복문도 하나의 객체처럼 인터페이스를 정의하고 프로그램을 구현하니 여러 군데 활용할 수 있다.
* 구현 되어있는 UnsortedArrayList에 대한 재사용으로 코드를 구현하여 편리하였다.
* **단점**
* 반복문에 비해 이해하기가 어렵다. 탈출 조건과 어떻게 나누어 재귀적으로 구현할 지 설계하는데 시간이 많이 소요될 것 같다.
* UnsortedArrayList를 재사용하였는데 이번 과제와 관련이 없는 함수들도 그대로 구현되어 있어서 다소 지저분하다고 느낄 수 있다. 하지만 자료구조 특성상 감안해야 할 부분인 것 같다.

1. **실행 결과 분석**
2. **입력과 출력 (화면 capture하여 제출)**

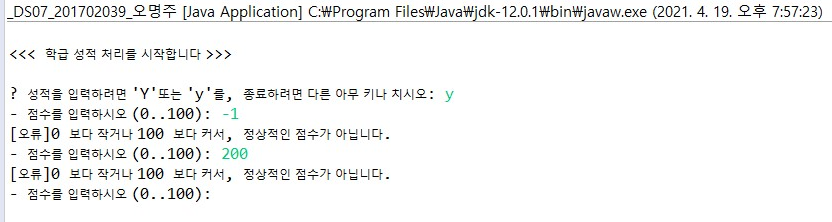
**[ 입출력 결과 ]**

****

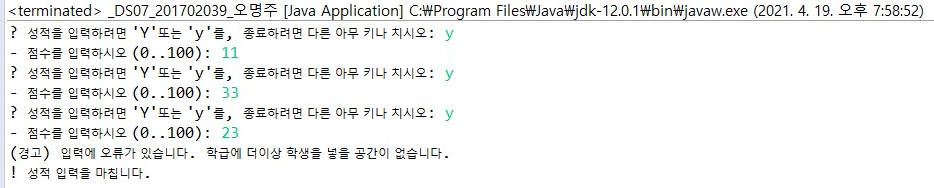
**[ 예외 처리 ]**

****

입력된 성적이 없는 경우

****

점수 범위가 0..100 사이가 아닌 경우

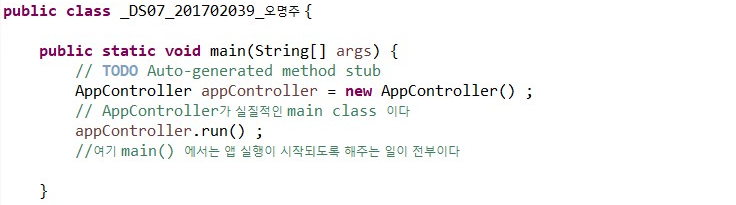
****

학생 수가 capacity(=10) 보다 많은 경우

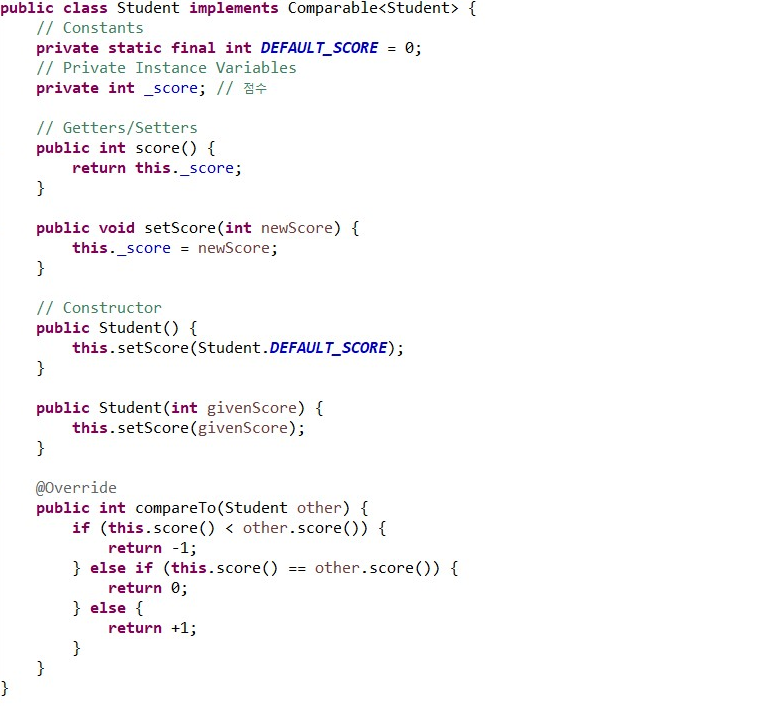
1. **결과 분석 (자신의 논리적 평가, 기타 느낀 점)**

* 학번 등 학생 정보를 추가하여 더 입력 할 것들이 있을 경우, 프로그램에서 바뀌어야 하는 부분은?
* Student 클래스에 학번을 나타내는 studentNumber 변수를 추가하고, 해당 정보에 해당하는 getter/setter을 추가하여 추가구현을 진행하면 된다.
* 재귀적이지 않은 문제 풀이에 비해 재귀적 문제풀이가 항상 좋은 성능을 낸다고 할 수 있을까?
* 코드의 가독성, 그리고 구현 면에서 복잡한 문제를 단순하게 접근한다는 점은 재귀의 장점이나 항상 좋은 성능을 낸다고 말 할 수는 없다.
* 피보나치 수열 같은 경우, 반복문이 훨씬 효율적인 구현이라고 할 수 있다. 하노이의 탑 역시 N이 커지면 결과가 나오는데 까지 시간이 오래 걸린다.
* 재귀함수는 기본적으로 스택 메모리를 사용하는데 재귀의 깊이가 깊어지면 stack overflow가 발생하기 쉽다. 스택 메모리를 초과하여 사용하는 문제이다.
* 느낀점
* 이해 면에서 반복문보다 재귀가 어려웠다. Lowest 값, Highest 값, Sum 등 재귀적으로 구할 때, 모두 3의 크기를 가진 배열이 있다고 가정하고 생각을 했던 것 같다.
* 주어진 코드를 이해하는 것도 쉽지 않았는데 구현해야 했다면 지금보다 더 오랜 시간을 투자해야 했을 것이라고 생각된다.

1. **소스코드**



[\_DS\_201702039\_오명주]



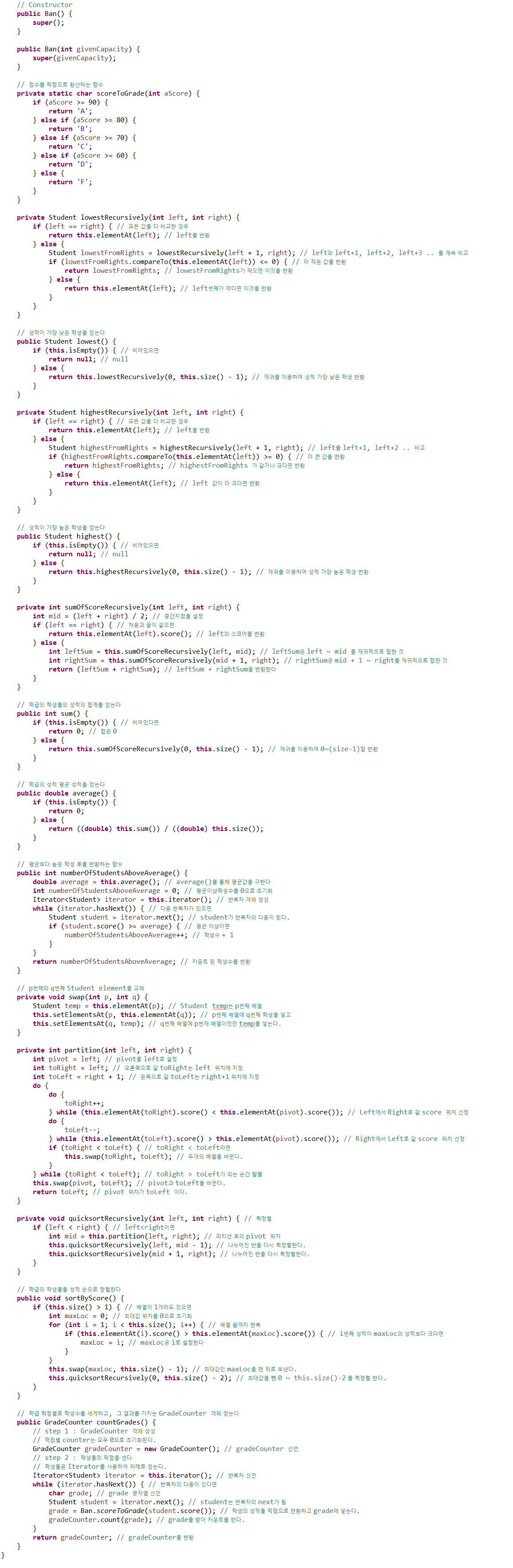
[Student]



[AppController]



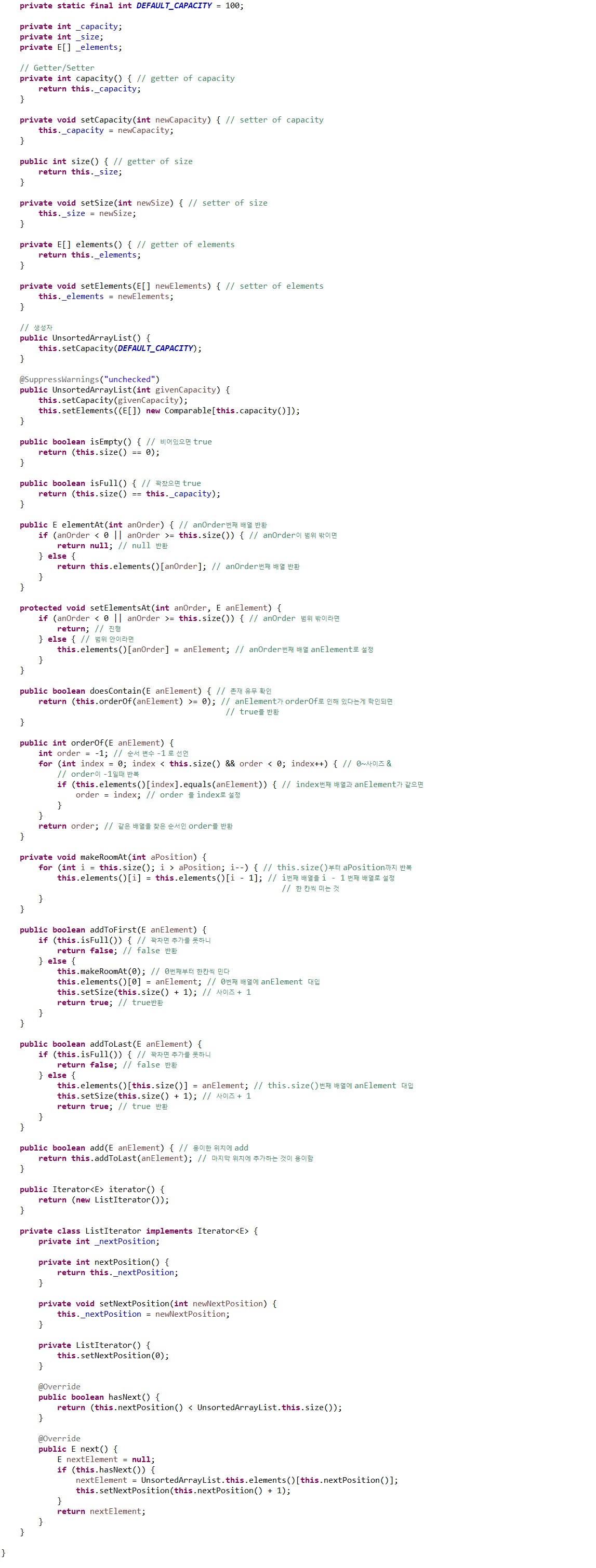
[AppView]



[Ban]



[GradeCounter]

[UnsortedArrayList]