**자료구조 실습 보고서**

[제12주] 정렬 성능 비교

**2021년 05월 24일**

**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.

* 정렬 결과 검증 프로그램에서의 각 클래스 별 MVC 구조 역할

**Model** :

* DataGenerator : 데이터를 생성하고 배열을 만든다.
* Sort<E> (Abstract) : 삽입정렬과 퀵정렬의 공통 기능.
* InsertionSort<E> : 삽입정렬의 기능을 구현한다.
* QuickSort<E> : 퀵정렬의 기능을 구현한다.
* ListOrder (Enum) : 오름차순/내림차순/랜덤값 세가지 방법에 대한 Enum 클래스.
* ExperimentManager : 실험 준비, 실험 시작을 진행한다.
* Experiment : 정렬 성능 측정 실험을 실행한다.
* ParameterSet : 단순히 실험에서 필요한 매개변수들 모아 놓는 역할.
* Timer : 실험 수행시간을 측정할 메소드를 구성한다.

**View** :

* AppView : 프로그램의 입/출력을 담당한다.

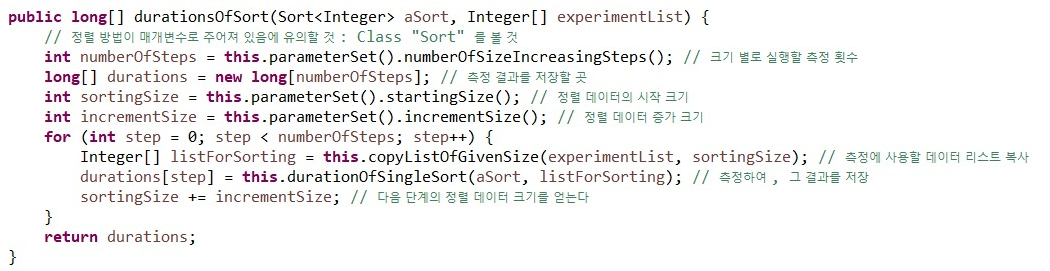
**Controller** :

* AppController : Model을 통해 생성된 결과물을 AppView를 통해 출력한다.

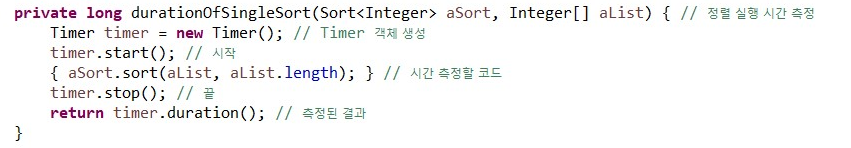
1. **함수 설명서**

* 주요 알고리즘

1. 리스트 정렬 시간 측정 후 측정값 배열 반환

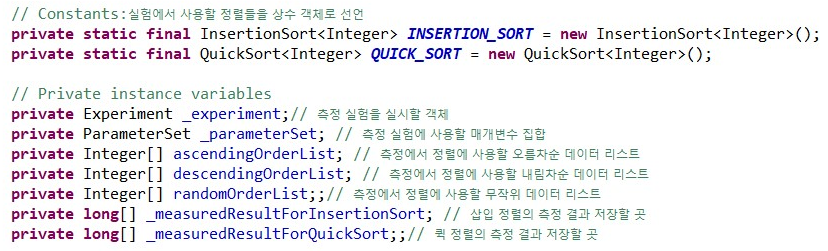


* 정렬 방법과 리스트를 매개변수로 받아와 측정 결과를 배열로 반환하는 함수이다.
* 측정 횟수, 시작크기, 증가크기를 parameterSet에서 받아온다.
* 측정 횟수가 저장된 numberOfSteps만큼 반복하여 for문을 반복한다.
* 측정에 사용할 데이터 리스트를 복사한 후, durationOfSingleSort 함수를 이용하여 측정한 후 그 결과를 durations 배열에 저장한다. 다음 단계의 정렬 데이터 크기를 얻은 후 모두 끝나면 durations 배열을 반환한다.

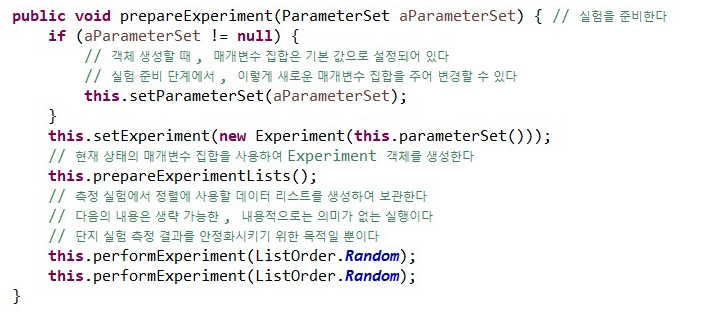


* 정렬 실행 시간을 측정하는 durationOfSingleSort 함수
* Timer 객체를 생성한다.
* start() 함수로 시간 측정을 시작하고 stop() 함수로 시간 측정을 종료하는데 사이에 시간 측정을 실행할 코드를 삽입한다.
* 측정 후, 측정 시간 값의 배열 (long) duration[] 이 반환된다.

1. Experiment Manager



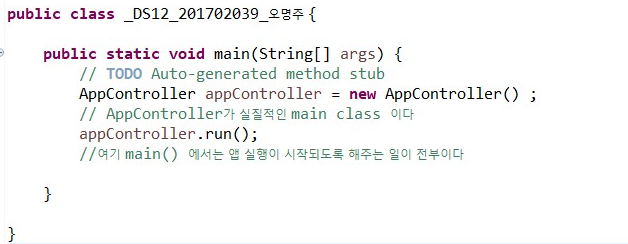
* 실험에 필요한 변수.
* 실험할 객체, 매개변수 집합, 오름차순/내림차순/무작위 데이터 리스트, 결과 저장 변수 등을 private으로 선언한다.



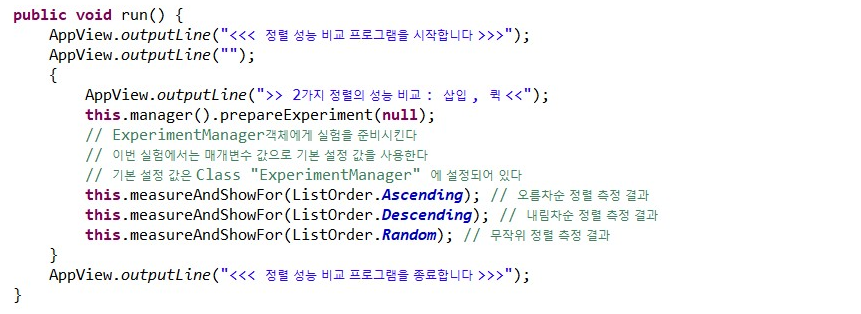
* 실험을 준비하는 prepareExperiment 함수
* 실험 준비 단계에서 새로운 매개변수 집합을 주어 변경한다.
* Experiment 객체를 생성하여 설정한다.
* 실험 측정 결과를 안정화하기 위해 performExperiment 를 통한 Random 리스트 실험을 2번 진행한다.

1. **종합 설명서**

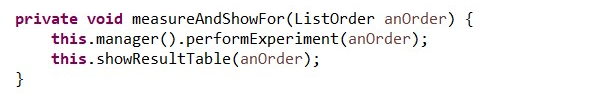
* 프로그램 실행 순서대로 설명해보자.



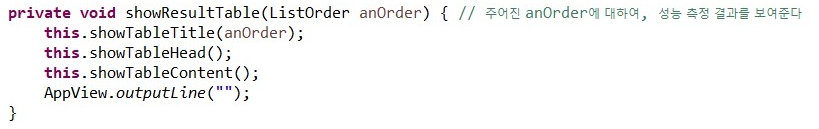
main에서 AppController 의 객체를 생성하여 run 한다. 프로그램을 실행한다.



* AppController의 run 함수이다. 프로그램 실행
* 오름차순, 내림차순, 랜덤 값 리스트에 대한 정렬 측정을 실행한다.



* 배열 생성 방법 anOrder을 인자로 받고 ExperimentManager()의 performExperiment를 실행한다.



* 결과를 출력하도록 하는 showResultTable() 함수이다. anOrder가 주어져 해당 결과를 반환한다.
* showTableTitle() 함수는 테이블 제목을, showTableHead() 함수는 테이블 헤드를, showTableContent() 함수는 측정 결과를 출력한다.

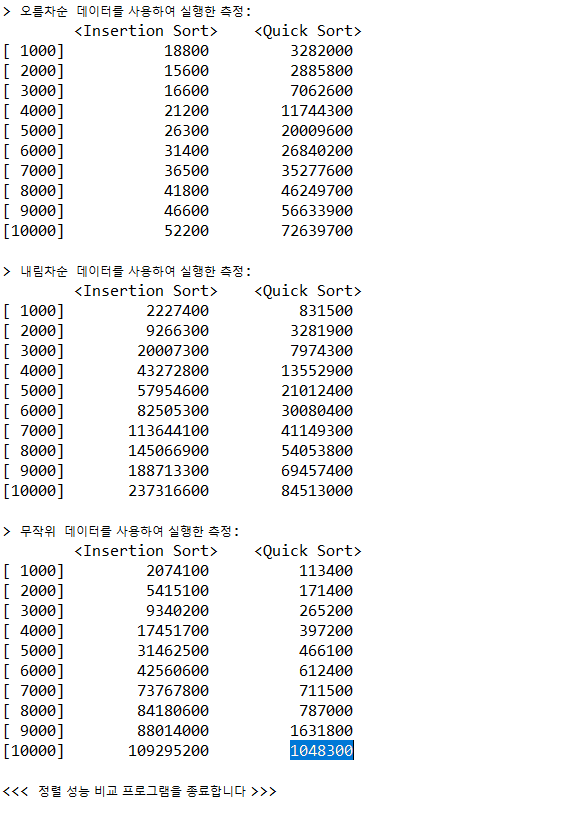


* 데이터 결과에 맞게 형식을 맞추어 출력하게 된다.
* parameterSet() 의 각종 매개변수를 받아와 진행한다. 횟수를 받아와 횟수만큼 for문을 반복한다.

1. **프로그램 장단점 / 특이점 분석**

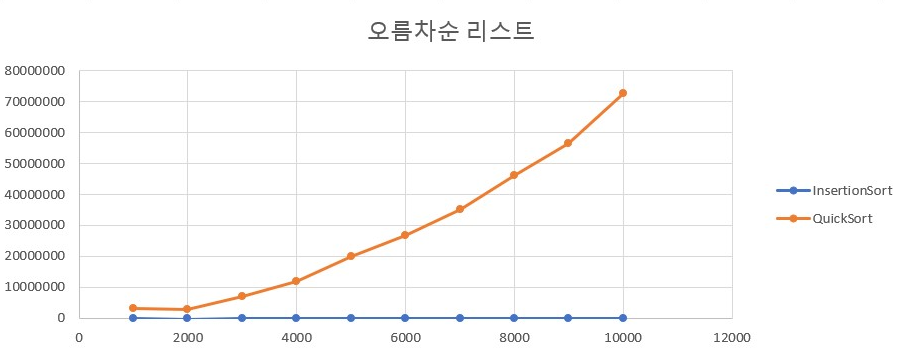
* **장점**
* 정렬에 공통부분을 추상클래스로 정의하여 새로운 정렬을 추가하거나 정렬을 수정하고 싶을 때, Sort나 Swap 함수에 대한 수정은 하지 않고 사용하여 구현할 수 있다.
* 각 정렬에 대한 검증을 해볼 수 있어서 구현한 정렬이 올바르게 작동하는지 확인할 수 있다.
* 실험, 실험준비, 실험에 사용되는 매개변수, 시간 측정 등 각각의 기능을 갖는 클래스들로 기능을 나누어 설계하였기 때문에 가독성이 높고 수정이 용이하다.
* **단점**
* 기능을 많이 나누어 헷갈릴 수 있다. 클래스가 많아지며 코드가 길어진다.

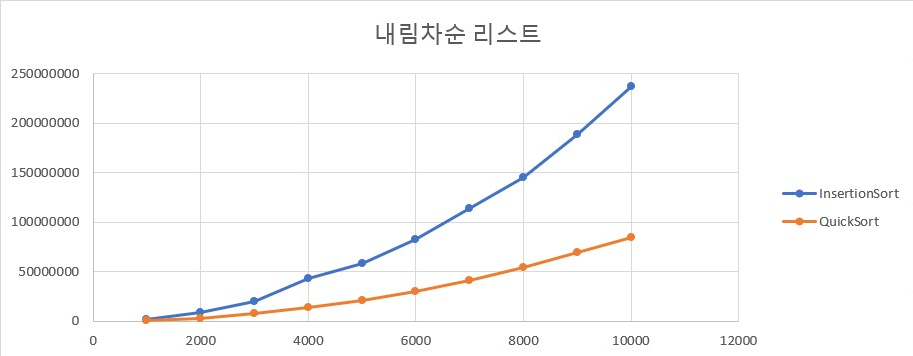
1. **실행 결과 분석**
2. **입력과 출력**

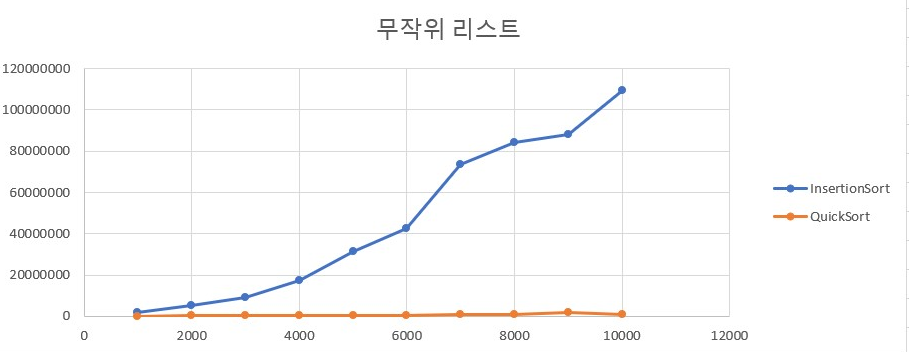


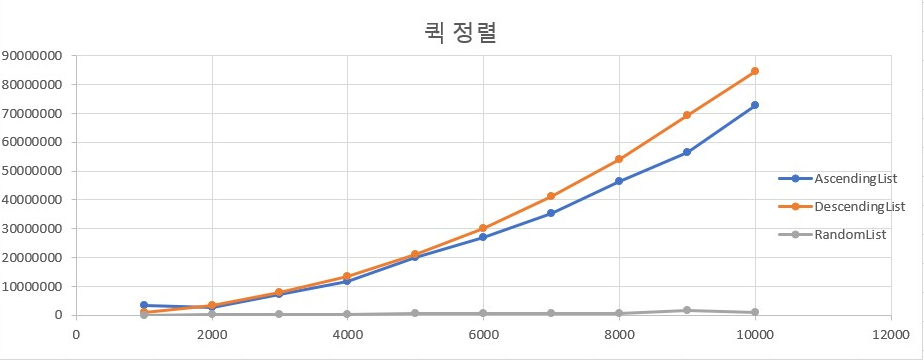
1. **결과 분석 (자신의 논리적 평가, 기타 느낀 점)**

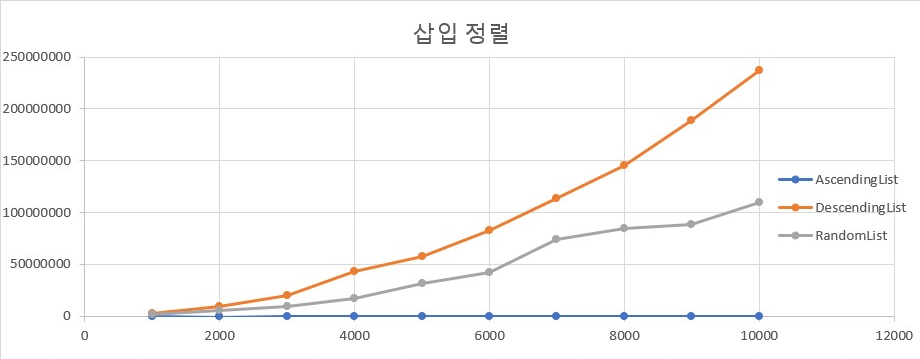
* 이론적 시간 복잡도
* 삽입 정렬 : 최선의 경우, 이미 정렬된 리스트에 대하여 O(N) 만큼이며, 최악의 경우 배열 전체를 비교해야 하므로 O(N^2)이다. 알고리즘이 간단하므로 배열의 크기가 작을수록, 배열이 이미 정렬되어 있을수록 효과적이다.
* 퀵 정렬 : 평균적으로 O(NlogN) 이고, 오름차순 일 때 최악의 경우 O(N^2)이다.





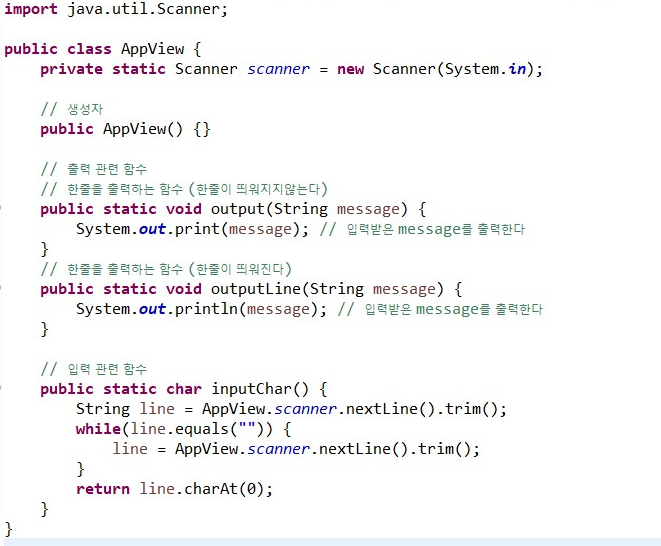






* 주어진 예시에 비해 성능 측정 값이 너무 높게 나와 잘 못 측정된 것은 아닌가 생각했는데 그래프로 나타내보니 맞게 측정되었음을 확인할 수 있었다.

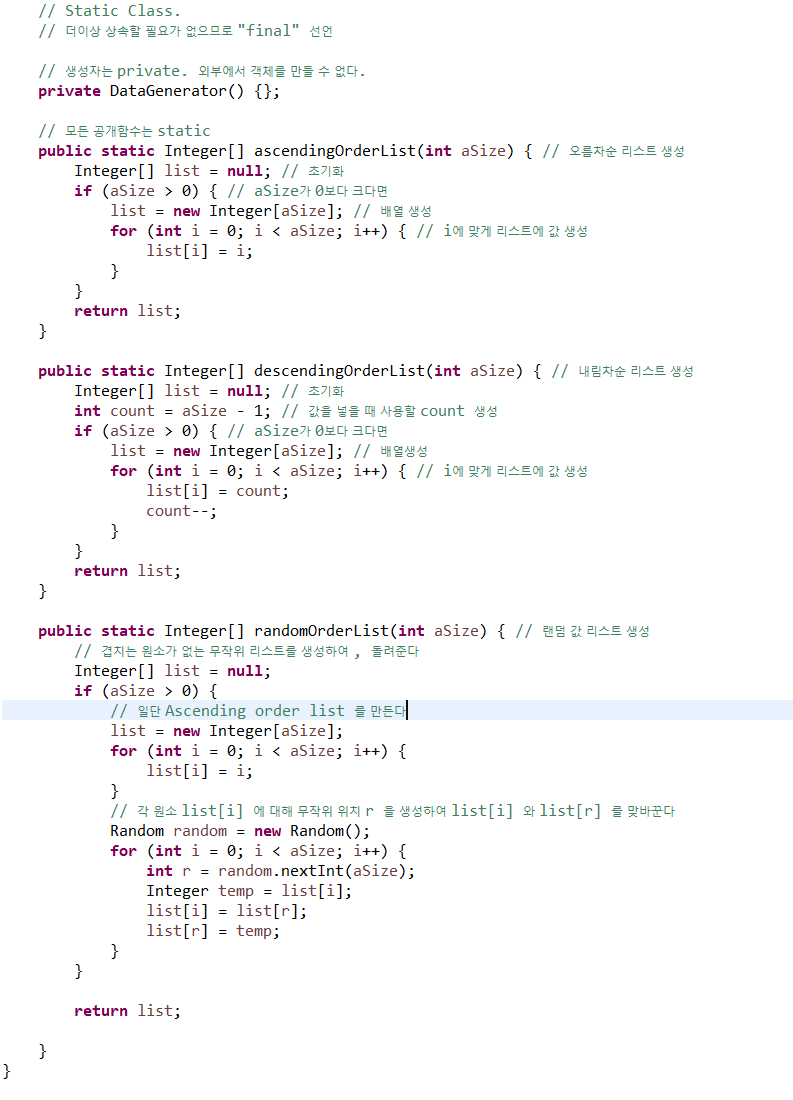
1. **소스코드**

****

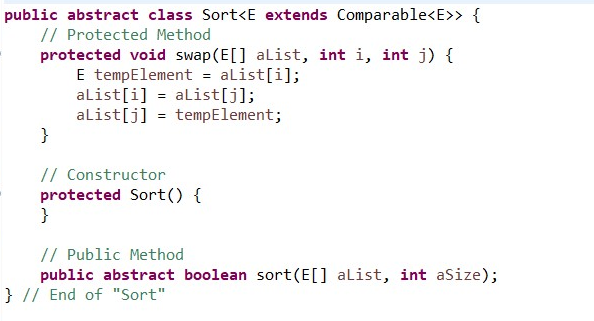
**[AppView]**

****

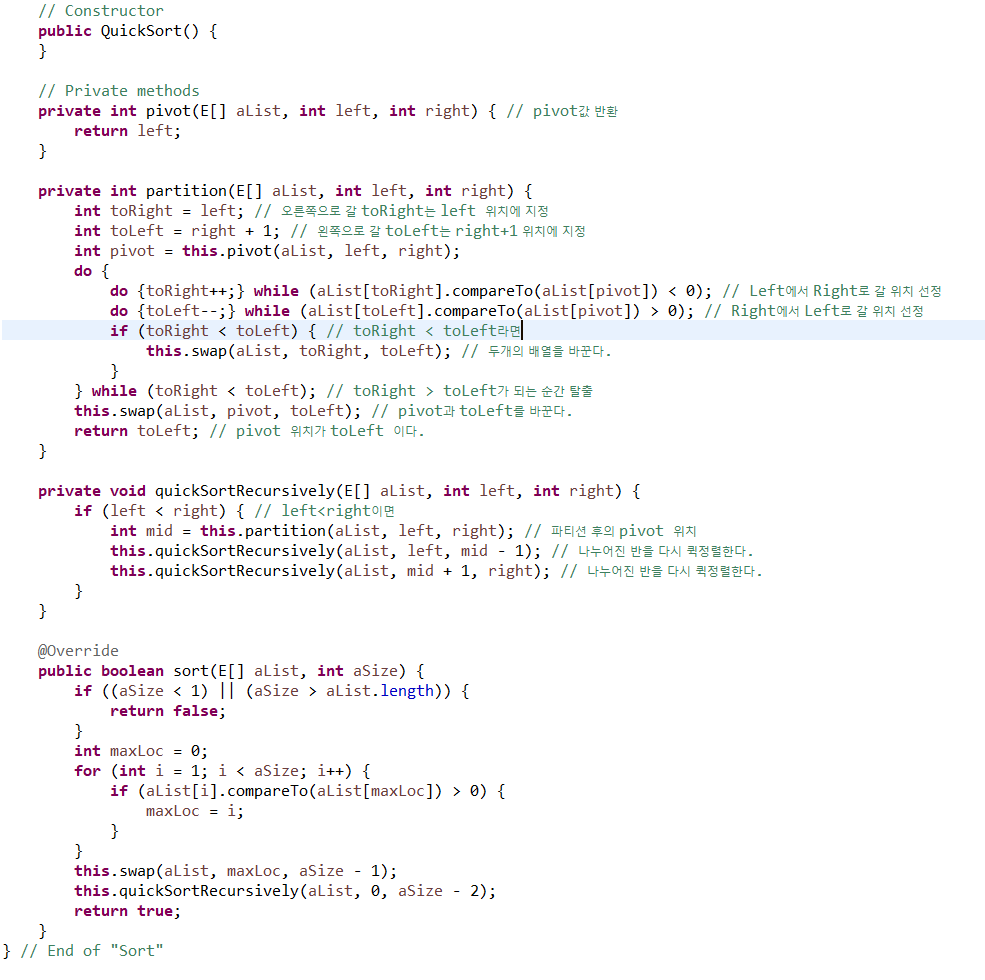
**[InsertionSort]**



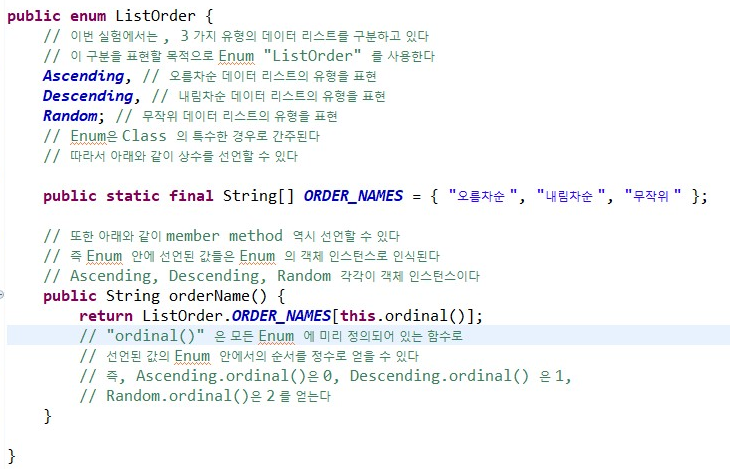
**[DataGenerator]**

****

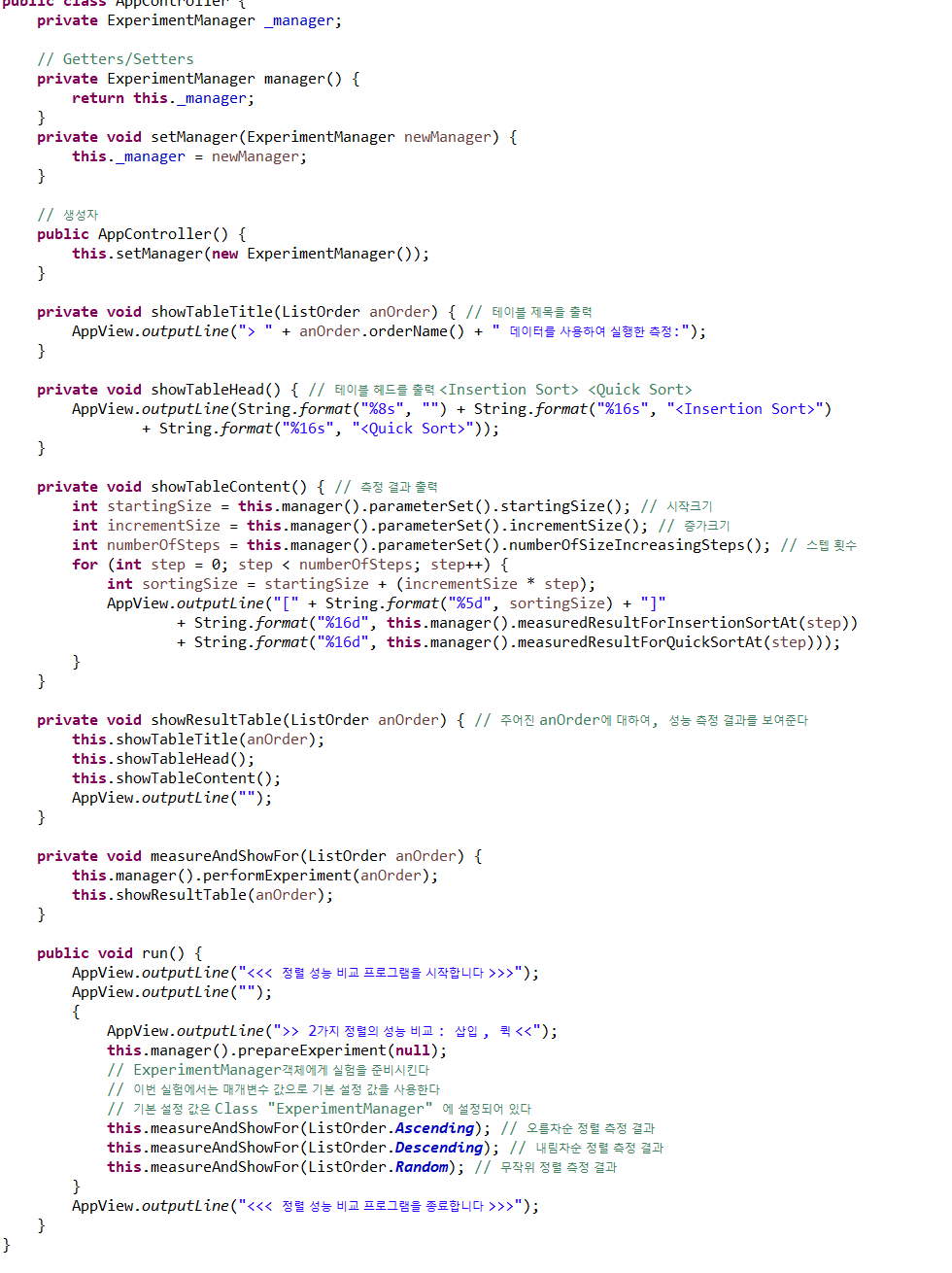
**[Sort]**



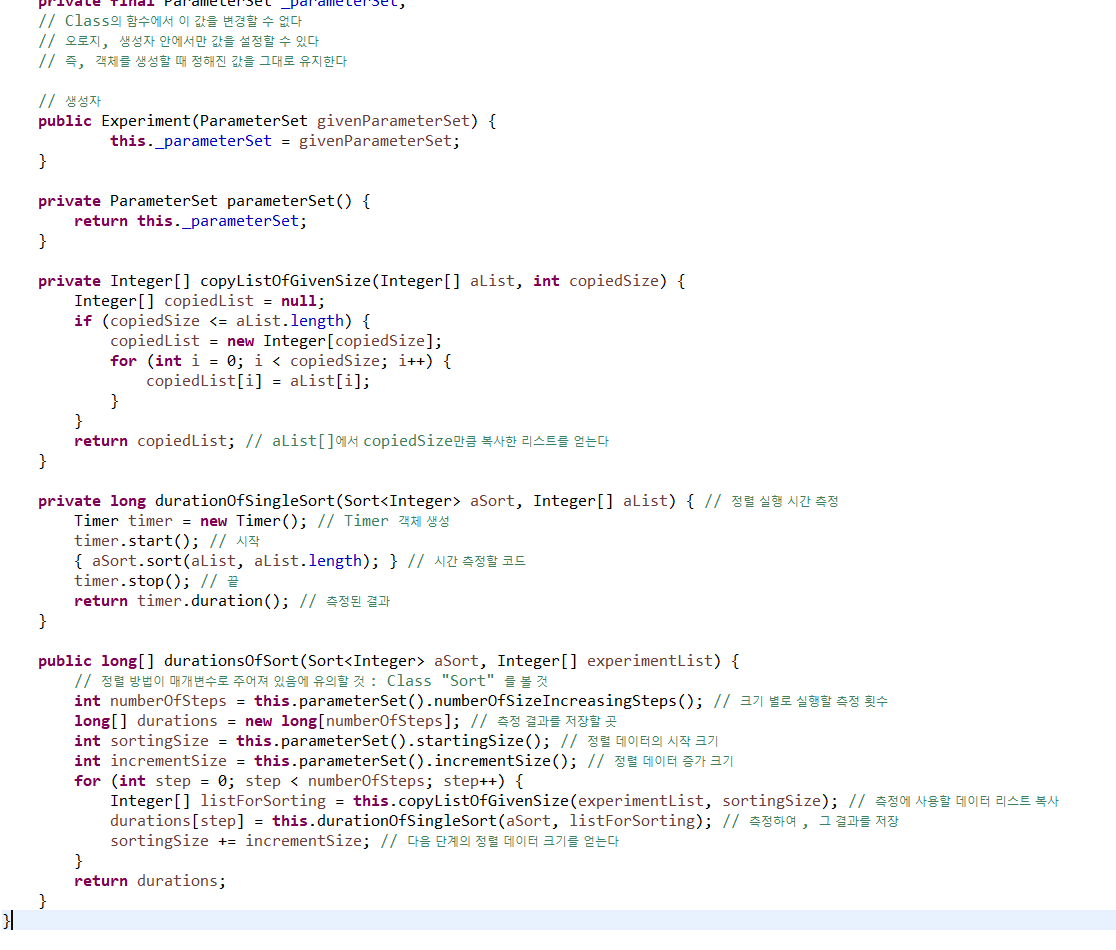
**[QuickSort]**

****

**[ListOrder]**



**[AppController]**



**[Experiment]**



**[ParameterSet]**



**[ExperimentManager]**