자료구조 실습 보고서

[제 11 주] 정렬 결과의 검증

제출일	2017/05/23
학 번	201000287
소 속	일어일문학과
이 름	유다훈

201000287 일어일문학과 유다훈

1 프로그램 설명서

1 주요 알고리즘 및 자료구조

- 알고리즘
  - 입출력
    - ◆ 필요한 데이터는 프로그램에서 생성
    - ◆ 성능 검증 결과를 출력
    - ◆ 삽입 정렬, 퀵정렬
- 자료구조
  - 오름차순 데이터 리스트
  - 내림차순 데이터 리스트
  - 무작위 데이터 리스트

1 함수 설명서

Class	AppController			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	AppController()	없음	없음	_appView 변수와 입력받은 문자와 추가된 문자, 무시된 문자의 수를 초기화하는 메소드를 실행하는 생성자 메소드
	run()	없음	없음	성능 검증프로그램을 실행하는 실행 메소드
	void validateWithAscedingOrderList()	없음	없음	오름차순 데이터 생성 및 출력
	void validateWithDescendingOrderList()	없음	없음	내림차순 데이터 생성 및 출력
	void validateWithRandomOrderList()	없음	없음	랜덤순 데이터 생성 및 출력
	void showFirstPartOfDataList()	없음	없음	생성한 데이터의 리스트 앞 5 원소 출력
	void validateSortsAndShowResult()	없음	없음	데이터를 정렬하고 잘 정렬되었는지 출력한다
	void validateSort()	없음	없음	데이터를 정렬한다
	Interger[] copyList(integer[] aList)	Integer 형 배열	Integer 형 배열	배열을 복사하여 리턴한다.
	boolean sortedListIsValid(integer[] aList)	integer 형 배열	잘 정렬되었으면 true, 아니면 false	전달받은 배열을 검사하여 정렬이 잘 되었는지 확인한다.
	void showValidationMessage(Sort<Integer> aSort, Integer[] aList)	Interger 형 Sort, Integer[] 배열	없음	정렬방법에 따라 정렬된 리스트를 전달받아 정렬이 잘 되었는지 확인

Class	AppView			
Method	메소드	파라미터 설명	리턴값	메소드 설명

	AppView()	없음	없음	생성자 메소드 값을 입력받는 스캐너 생성
	void output(String aString)	문자열	없음	전달받은 문자열을 출력한다.
	void outputLine(String aString)	문자열	없음	전달받은 문자열을 출력하고 행을 바꾼다.

Class	final class DataGenerater()			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	static Integer[] ascendingOrderList(int aSize)	배열의 최대값	배열	전달받은 수만큼 배열의 크기를 정하고 배열의 크기만큼 데이터를 만들어서 저장한다.
	static Integer[] descendingOrderList(int aSize)	배열의 최대값	배열	전달받은 수만큼의 배열의 크기를 정하고, 배열의 크기만큼 데이터를 만들어 거꾸로 저장한다.
	static Integer[] randomOrderList(int aSize)	배열의 최대값	배열	전달받은 수만큼의 배열의 크기를 정하고, 배열의 크기만큼 데이터를 만들어 랜덤하게 저장한다.

Class	abstract class Sort<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	protected void swap(E[] aList, int t, int j)	배열, 정수형 변수 2 개	없음	전달받은 정수형 변수위치에 저장된 데이터 위치를 서로 바꿈
	public abstract boolean sort(E[] aList, int aSize)	배열, 배열의 최대길이	없음	정렬을 담당하는 메소드. abstract 선언을 하여 자식 메소드에서 구현을 강제시킴.

Class	InsertionSort <E extends Comparable <E>> extends Sort<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	boolean sort(E[]aList, int aSize)	배열, 배열크기	잘 삽입되었으면 true, 아니면 false	삽입정렬의 개념을 이용하여 정렬하는 메소드

Class	QuickSort<E extends Comparable<E>> enxtends Sort<E>			
Method	메소드	파라미터 설명	리턴값	메소드 설명
	int pivot(E[] aList, int left, int right)	배열, 배열의 첫부분인덱스, 배열의 끝부분인덱스	피봇값	정렬 시 기준값이 되는 피봇값을 정하는 메소드

	int partition(E[] aList, int left, int right)	배열, 배열의 첫 부분 인덱스, 배열의 끝 부분 인덱스	중간값	재귀적 퀵정렬을 하기위해 필요한 중간값을 만든다.
	void quickSortRecursively(E[] aList, int left, int right)	배열, 배열의 첫 부분 인덱스, 배열의 끝 부분 인덱스	없음	파티션을 이용하여 배열을 두 군데로 나눈 후 나누어진 부분에 대하여 다시 퀵정렬을 재귀적으로 실행하는 메소드

2 종합 설명서

- 프로그램을 실행하면 코드의 짜여진 순서에 따라 오름차순, 내림차순, 무작위 리스트순으로 정렬을 실행한다
- 정렬은 insertion sort 와 quick sort 순으로 진행된다.
- 정렬해야하는 배열이 오름차순순으로 제대로 정렬되었다면 결과가 올바르다는 메세지를 출력한다
- 정렬해야하는 배열이 오름차순순으로 제대로 정렬되지 않았다면 결과가 올바르지 않다는 메세지를 출력한다.

2 프로그램 장단점 분석

- 장점
  - 데이터를 자동으로 생성하여 삽입정렬과 퀵정렬을 사용해볼 수 있다.
  - 오름차순, 내림차순, 무작위 로 생성된 데이터에 대해 어느 정렬이 더 효율적으로 정렬하는지 알 수 있다.
  - 같은 정렬 기능을 부모클래스로 선언하여 사용하므로 인해 불필요한 코드를 작성하지 않아도 된다.
  - 객체를 생성하지 않아도 되는 클래스와 메소드를 final 키워드와 static 키워드를 이용해 사용해 볼 수 있다.
  - enum 클래스에서 메소드와 static 객체를 사용할 수 있다.

- 단점
  - 삽입정렬의 경우는 내림차순, 퀵정렬의 경우 내림차순, 오름차순의 경우 정렬시간이 제일 길게 걸린다.
  - 상속 개념을 이해해야한다.

3 실행 결과 분석

1 입력과 출력

프로그램 실행

```
<< 정렬 알고리즘의 정렬 결과를 검증하는 프로그램을 시작합니다. >>

> 정렬 결과의 검증
```

입출력 1-1 - 오름차순 리스트 앞 부분 출력 및 정렬 결과 출력

```
[오름차순리스트] 의 앞 부분: 0 1 2 3 4
[오름차순리스트] 를 [InsertionSort] 한 결과는 올바릅니다.
[오름차순리스트] 를 [QuickSort] 한 결과는 올바릅니다.
```

입출력 1-2 - 내림차순 리스트 앞 부분 출력 및 정렬 결과 출력

```
[내림차순리스트] 의 앞 부분: 9999 9998 9997 9996 9995
[내림차순리스트] 를 [InsertionSort] 한 결과는 올바릅니다.
[내림차순리스트] 를 [QuickSort] 한 결과는 올바릅니다.
```

입출력 1-3 - 무작위 리스트 앞 부분 출력 및 정렬 결과 출력

```
[무작위리스트] 의 앞 부분: 447 8974 7445 7764 5784
[무작위리스트] 를 [InsertionSort] 한 결과는 올바릅니다.
[무작위리스트] 를 [QuickSort] 한 결과는 올바릅니다.
```

프로그램 종료

```
<< 정렬 알고리즘의 정렬 결과를 검증하는 프로그램을 종료합니다. >>
```

2 결과 분석

- 오름차순, 내림차순, 무작위 순으로 생성된 데이터 리스트에 대하여 각각 삽입 정렬과 퀵 정렬로 오름차순 순으로 정렬을 시도하였고, 올바른 결과를 출력하였다.
- 실행 결과 삽입정렬은 내림차순, 퀵 정렬은 오름차순, 내림차순에 대하여 이미 정렬이 되어져있는 데이터라면 좋지않은 시간복잡도라는 것을 알 수 있다.(프로그램 실행시 걸리는 시간도 퀵정렬이 오래걸림)
- 부모 클래스 Sort 를 선언하여 상속을 구현하였다

- Sort 를 상속받는 자식클래스는 부모클래스에서 이미 구현한 swap 메소드를 사용할 수 있음
- 또한 sort()메소드를 abstract 선언을 하여, 자식클래스에서 무조건 구현하게 강제함.
- validateSort(Sort<Integer> aSort)메소드는 Sort 클래스 타입의 객체를 넘겨받고, 내부에서 Sort 클래스의 sort()메소드를 실행하고 있음. 실제 호출할 때에는 InsertionSort, QuickSort 와 같은 자식클래스를 파라미터로 넘겨주고 있는데 이것은 부모클래스에서 이미 선언한 메소드라면, 부모클래스를 상속받는 어떠한 자식클래스를 받는다고 하여도 부모클래스에서 해당 기능을 실행하게 되어 있음. 입력할 코드를 절약하여 시간을 단축할 수 있음.

- 결론
  - 삽입정렬과 퀵정렬을 구현해볼 수 있다
  - 상속에 대해서 사용할 수 있다.