**자료구조 실습 보고서**

[제08주] 스택 : 기본기능

**2021년 04월 26일**

**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.

* 스택 프로그램에서의 각 클래스 별 MVC 구조 역할

**Model** :

* Stack(Interface) : 스택 변수와 추상 메소드를 구성한다.
* ArrayList<T> : 스택 배열 기능을 구성한다.

**View** :

* AppView : 프로그램의 입/출력을 담당한다.

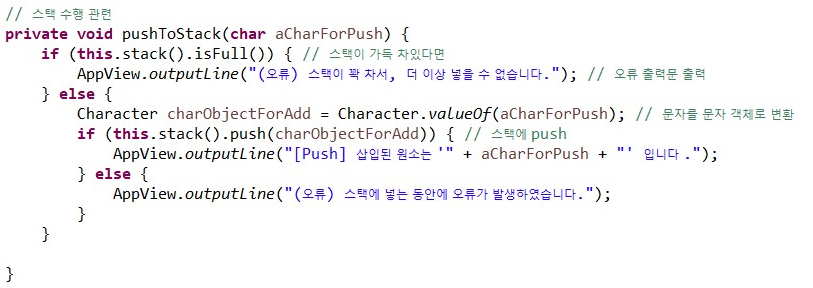
**Controller** :

* AppController : Model을 통해 생성된 결과물을 AppView를 통해 출력한다.

1. **함수 설명서**

* 주요 알고리즘

1. Push



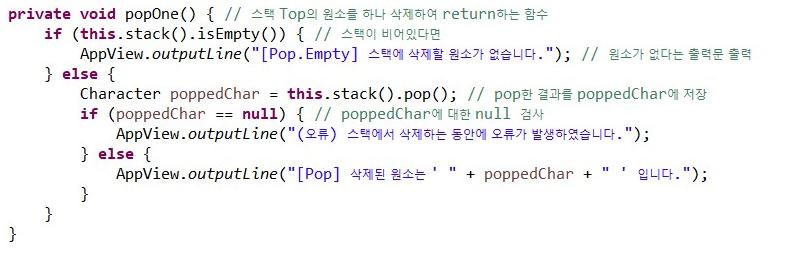
Stack Interface에 정의된 추상 메소드를 ArrayList class에서 구현한다. Stack의 Top에 원소를 삽입하는 형태이며, 배열에서는 인덱스 size부분에 삽입하는 것에 해당한다.

* 문자 추가를 위해 aCharForPush의 char 형 문자를 입력 받는다.
* Stack이 가득 차 있다면 오류 출력문을 출력한다.
* 그것이 아니라면, aCharForPush 문자를 문자 객체로 변환하여 저장한다.
* Stack의 push 함수를 이용하여 삽입한다. 성공하면 true를 false라면 오류를 출력한다.



배열의 마지막에 원소를 추가하는 Stack을 구현한 ArrayList의 push 함수이다. addToLast 함수를 이용하여 배열의 마지막에 원소를 삽입한다.

1. PopOne



스택 Top의 원소를 하나 삭제하여 반환하는 함수이다.

* 먼저 스택이 비어있다면 오류 출력문을 출력한다.
* 그게 아니라면, Stack의 pop 함수를 이용하여 반환 받은 Top 원소를 poppedChar에 저장한다.
* poppedChar이 null이라면 오류 출력문을 출력한다.
* null이 아니라면 삭제된 원소에 대한 정보를 출력한다.



ArrayList의 pop 함수이다. removeLast 함수를 이용하여 구현하였다. Stack의 Top의 인덱스가 배열에서는 size 부분이기 때문에 마지막 원소를 삭제하는 removeLast를 이용한다.

1. PopN



스택 Top의 원소를 입력 받은 수만큼 반복하여 삭제하여 반환하는 함수이다.

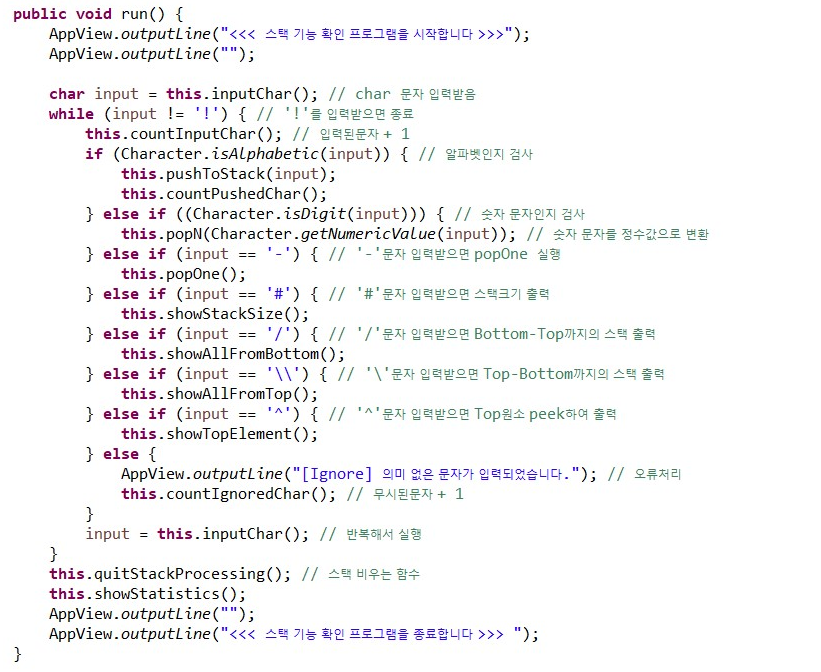
* 입력 받은 numberOfCharsToBePopped 만큼 반복하여 스택의 Top원소를 삭제한다.
* 인자가 0이면 예외처리한다.
* 입력 받은 수만큼 반복 && 스택이 empty가 아닌 경우 반복한다.
* Stack의 pop 함수를 이용하여 반환 받은 원소를 poppedChar 변수에 저장한다.
* poppedChar이 null이라면 오류문을 출력한다.
* 그게 아니라면 반복해서 poppedChar 변수에 대해 출력한다.
* 만약 입력 받은 수가 스택에 있는 원소의 수보다 크다면 오류문을 출력하기 위해 조건문을 이용하여 선언한다.
* pop에 대한 설명은 popOne 함수 설명을 참조

1. **종합 설명서**

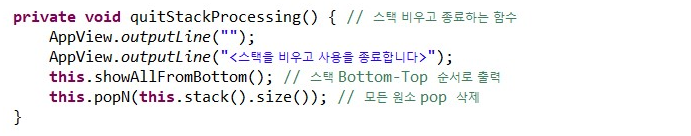
* 프로그램 실행 순서대로 설명해보자.



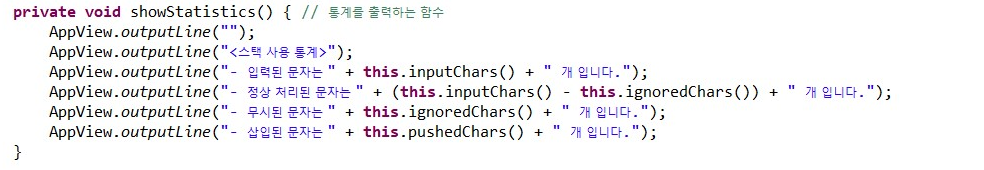
main에서 AppController 의 객체를 생성하여 run 한다. 프로그램을 실행한다.



* 먼저 수행할 문자를 inputChar 함수를 이용하여 입력 받는다.
* ‘!’ 를 입력 받으면 종료한다. (종료조건)
* 입력 받은 문자에 따라 각각 다른 수행 처리를 해준다. 문자를 입력 받으면 countInputChar를 +1 하여 입력된 문자를 count 한다.
* 입력 받은 문자가 알파벳이라면 스택에 삽입하는데 그 때 countPushedChar 변수를 +1 하여 삽입된 문자를 count 한다.
* 의미 없는 문자가 들어오면 (예:$,% 등) 예외처리를 해주는데 이 때 countIgnoredChar 변수를 +1 하여 무시된 문자를 count 한다.
* 만약 숫자 문자가 입력되었다면 숫자 문자를 정수값으로 변환하고 정수만큼 popN을 실행한다.
* 삽입된 문자가 ‘!’라면 quitStackProcessing 함수와 showStatistics 함수를 호출하고 프로그램을 종료한다.



스택을 비우고 종료하는 quitStackProcessing 함수이다. 스택을 비운다는 출력문을 출력하고 스택을 Bottom부터 Top까지 원소를 출력한다. 그리고 popN 함수에서 size만큼의 인자를 주어 size만큼 pop을 반복한다. 스택 모든 원소를 삭제하게 된다.



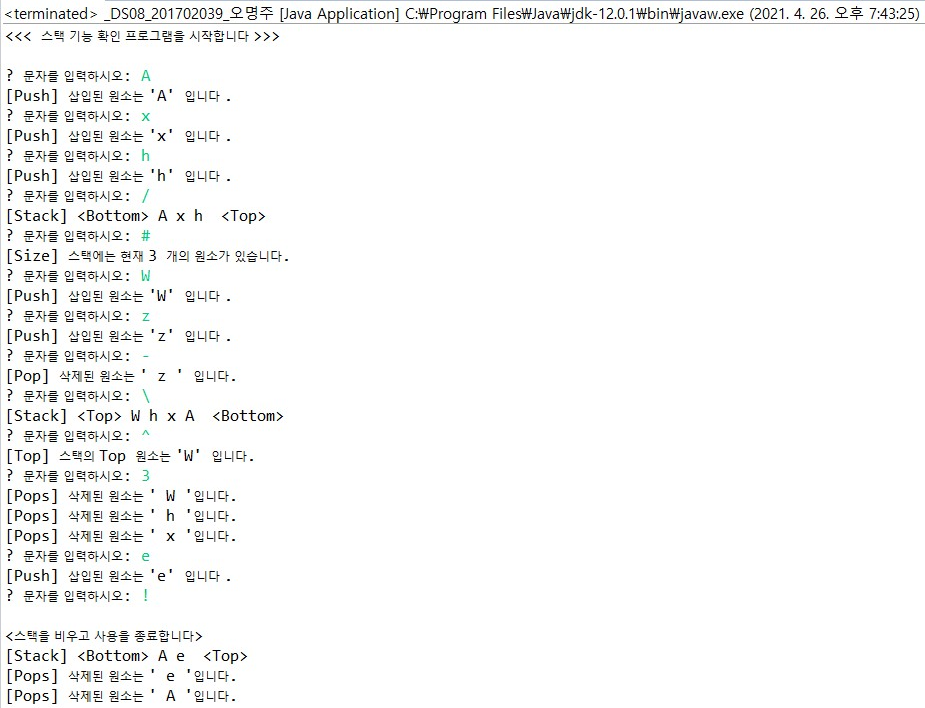
프로그램에 대한 통계를 출력하는 showStatistics 함수이다. 입력된 문자, 정상 처리된 문자, 무시된 문자, 삽입된 문자를 출력할 때는 inputChars, ignoredChars pushedChars를 이용한다. 프로그램 실행동안 count 하여 저장한다.

1. **프로그램 장단점 / 특이점 분석**

* **장점**
* 지난 주 과제를 통해 구현 되어있는 UnsortedArrayList에 대한 재사용으로 코드를 구현하여 편리하였다.
* 삽입도 addToLast, 삭제도 removeLast로 구현된 Last In First Out 구조여서 구현이 비교적 쉽다.
* 배열을 통해 구현하여서 인덱스를 통한 접근이 가능했다. 접근 가능성이 좋고 속도가 빠르다.
* ArrayList에 대한 정의를 하고 Stack으로 활용하니 구현이 편리하였다.
* LinkedList에 대한 Stack 구현도 쉽게 할 수 있다.
* **단점**
* 데이터에 대한 접근이 Top 부분을 통해서만 가능하기 때문에 Bottom 원소나, 중간 원소에 대한 접근이 불가능하다.

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

**[ 입출력 결과 ]**

****

****

**(한번에 캡처가 안되어 나누어서 캡처한 점 참고 바랍니다.)**

**[ 예외 처리 ]**

****

* 입력에 공백이 있는 경우 -> 공백을 제거하고 인식

****

* capacity 보다 더 입력한 경우

****

* 스택 원소 수보다 삭제할 수가 많은 경우

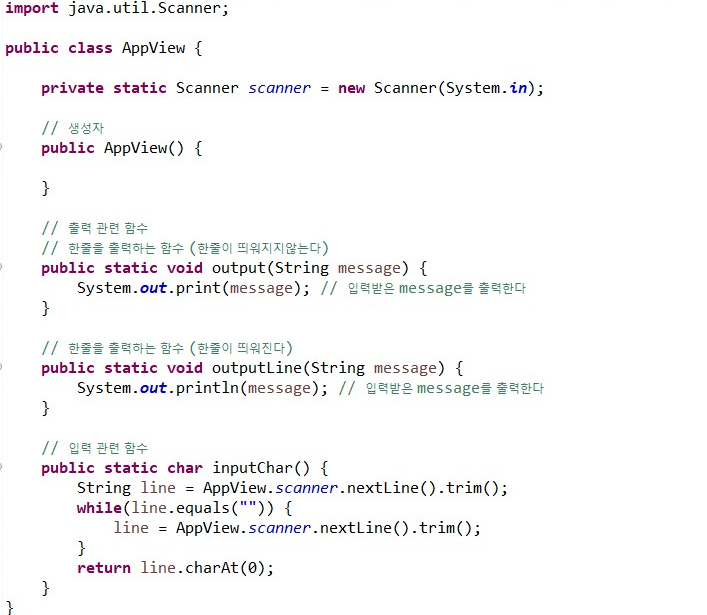


* 의미 없는 문자가 입력된 경우

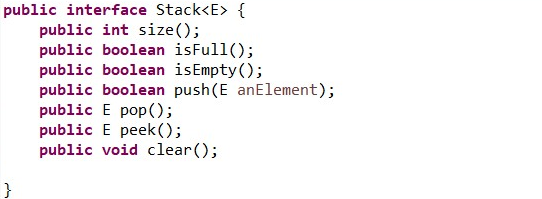
1. **결과 분석 (자신의 논리적 평가, 기타 느낀 점)**

* Stack은 Last In First Out 구조로, 마지막 입력이 먼저 출력되는 후입선출 구조이다.
* ArrayList를 이용하여 구현하였는데 인덱스를 이용한 접근이 가능하여 접근성이 좋았으나 만약 배열이 가득 차면 스택 오버플로우가 발생할 수 있으며 배열을 더 크게 만들어 옮겨야 한다는 단점이 있다.
* 반면, LinkedList를 이용하여 구현하면 스택의 크기를 자유롭게 구성할 수 있어 크기에 대한 제한이 없다.
* 느낀점
* 자료구조 중 스택을 이용하는 법을 배웠는데 먼저 구조를 Array를 통해 정의해놓고 스택을 구현하니 편리하였다. LinkedList를 이용하여 구현한다고 해도 구조를 먼저 정의하고 구현하면 쉽게 구현할 것 같다.
* 스택을 이용할 수 있는 다양한 프로그램에 사용될 것이라고 생각된다.

1. **소스코드**

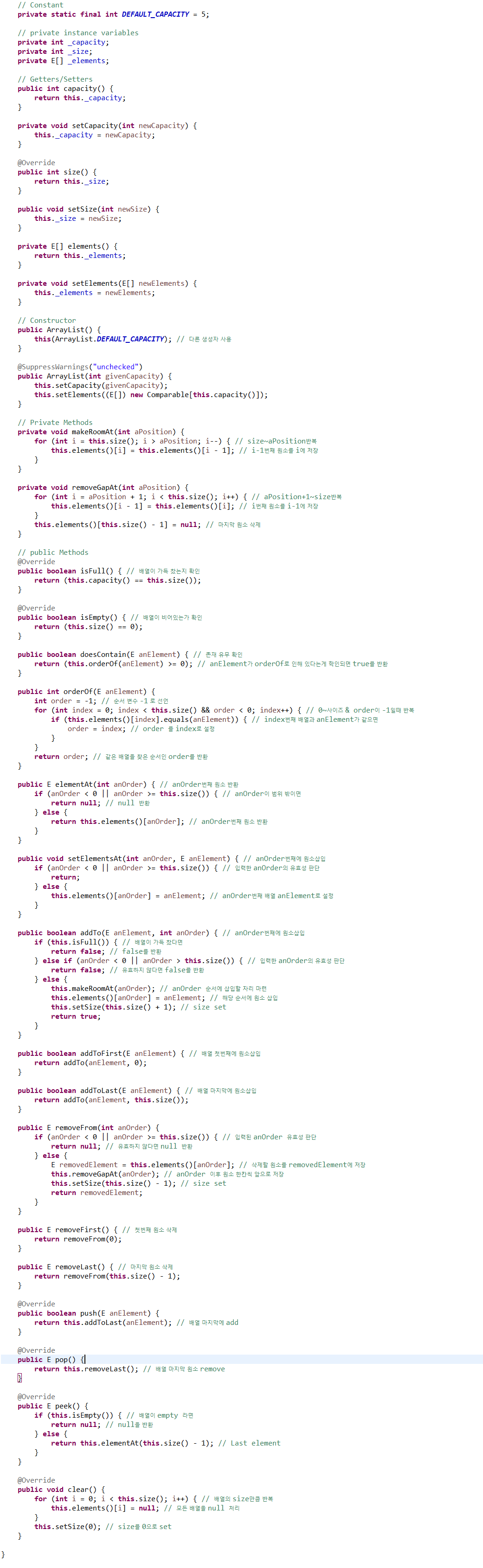


[AppView Class]



[Stack - interface]

[AppController]

[ArrayList]