**자료구조 실습 보고서**

[제01주] 마방진

**2021년 3월 14일**

**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.

* 마방진 프로그램에서의 각 클래스 별 MVC 구조 역할

Model :

* MagicSquare : 입력 받은 차수를 통해 마방진을 계산한다.
* Board : 계산된 마방진 판을 만들고 숫자를 넣는다.
* CellLocation : 마방진의 각 Cell 좌표를 구성한다.
* OrderValidity : 입력 받은 차수의 유효성을 검사하는 enum 이다.

View :

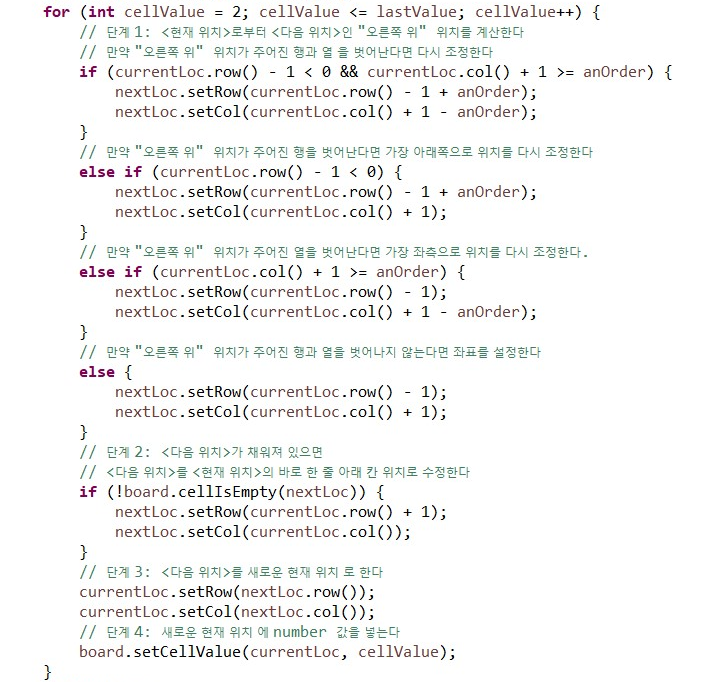
* AppView : 프로그램의 입/출력을 담당한다.

Controller :

* AppController : AppView를 통해 차수를 입력받고 Model 의 클래스에 전달하고 결과물인 마방진 판을 AppView를 통해 출력한다.

1. **함수 설명서**

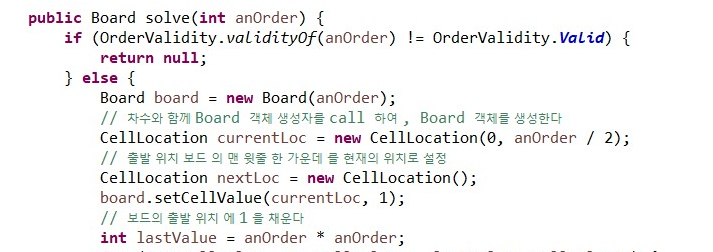
* 주요 알고리즘



마방진 프로그램에서 nextLoc Cell 위치 잡는 논리 : row – 1, col + 1 (현재 위치의 오른쪽 위 좌표로 설정)

1. 만약 nextLoc 위치가 row를 벗어난다면 row – 1에 입력 받은 차수만큼 더해준다.
2. 만약 nextLoc 위치가 col을 벗어난다면 col + 1에 입력 받은 차수만큼 빼준다.
3. 만약 nextLoc 위치가 row, col를 모두 벗어난다면 1)와 2) 모두 해준다.

* **이 클래스(MagicSquare)**의 전체적인 알고리즘은 다음과 같다.

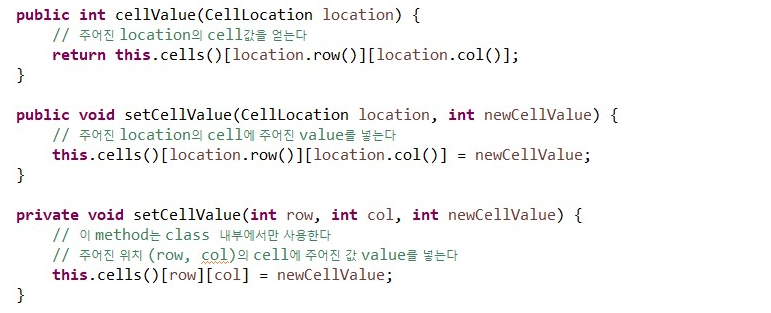


Board 클래스의 판을 입력 받은 차수 anOrder\*anOrder 크기만큼 생성한 후, 보드의 맨 윗줄 가운데를 시작 위치로 설정한다. 시작 위치인 currentLoc에 1을 넣은 후 2부터는 앞서 설명한 nextLoc 설정 알고리즘을 for문으로 반복한 후 nextLoc을 currentLoc으로 설정하고 board에 setCellValue 함수를 이용하여 값을 넣는다.

* **Board 클래스**는 다음과 같다.



마방진 보드 생성자가 있으며 입력 받은 차수 크기의 보드를 생성한다. 해당 Cell이 비었는지 안 비었는지 확인하는 cellIsEmpty 함수도 존재한다. 비었으면 true를, 채워져 있으면 false를 반환한다.



Cell에 값을 넣는 getter/setter 이 존재한다. 주어진 Location에 Cell값을 넣는 함수와 해당 Location에 주어진 value를 넣는 함수가 있다.

* 마방진에서 Cell 좌표를 구성하는 **CellLocation 클래스**는 다음과 같다.



기본 생성자는 다음과 같다. Cell의 좌표가 주어지지 않으면 상수 UNDEFINED\_INDEX (여기서는 -1로 미리 정의해 놓았다) 로 설정한다. 주어진 차수로 설정하는 생성자도 존재 한다.

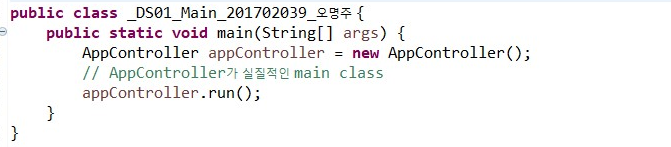


해당 클래스의 getter / setter 함수들이다. Row, col의 위치 좌표를 반환하고 해당 위치로 설정한다.

* 각 클래스와 함수마다 기능이 명확히 구분되어 있어 다소 복잡하지만 높은 생산성과 유연성을 가지고 있다.

1. **종합 설명서**

* 프로그램 실행 순서대로 설명해보자.

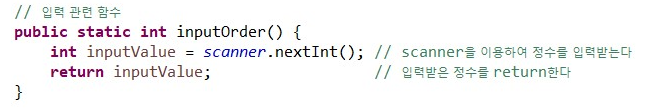
****

Main 클래스에서 AppController 객체 생성 후 appController의 run 메소드를 실행한다.



AppController의 run 메소드에서는 차수를 입력 받기 위한 출력문들과 입력 받은 차수의 유효성을 확인하는 함수를 호출한다. 이때, 입력 받을 때는 AppView 클래스의 함수를 이용한다. 만약 차수가 유효하다면 magicSquare 클래스에 전달하여 결과를 얻는다. showBoard 함수를 이용하여 결과 마방진을 출력한다. 이때, 마방진을 계산하는 알고리즘은 위 페이지를 참고한다.

* 이때, 입/출력을 담당하는 **AppView 클래스**는 다음과 같다.



입력 관련 함수에는 차수 입력 받는 inputOrder 함수가 존재한다. 자바 표준 입력 클래스 Scanner를 import하여 차수를 입력 받아 입력 받은 차수를 return 한다.

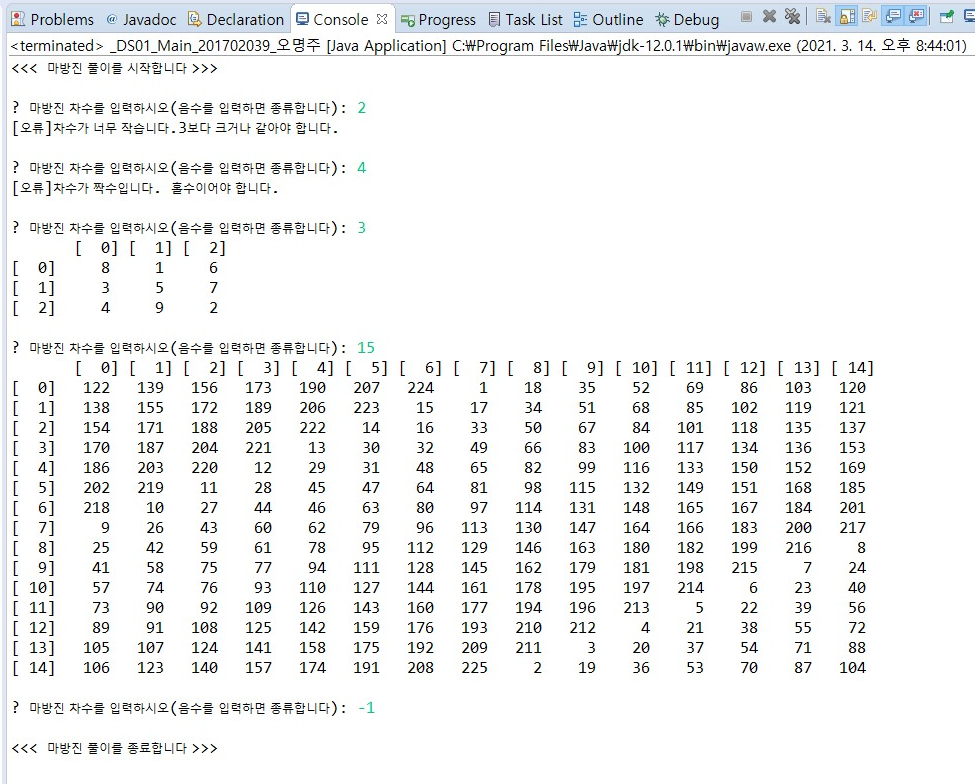


Console 창의 출력문을 출력하는 함수들과 마방진을 출력할 때 행과 열을 나타내는 숫자와 각 수를 주어진 위치에 출력하는 함수들이 있다. 각각의 상황에 맞게 print문의 구조를 다르게 설정한다. “%3d”는 3칸 중 뒤에서부터 값을 출력하는 형태이다.

1. **프로그램 장단점 / 특이점 분석**

* **장점**
* MVC 모델을 이용하여 가독성과 생산성이 뛰어나다. 각 클래스, 함수의 역할이 분명해서 코드와 프로그램을 잘 이해할 수 있다.
* 유사한 기능을 하는 다른 프로그램에도 재사용할 수 있다. 객체 지향 프로그램의 가장 큰 장점이라고 할 수 있다.
* 수정이 편리하다. 데이터나 기능을 수정하려고 하면 해당 메소드만 수정하면 되기 때문에 편리하다.
* **단점**
* 처음에 클래스와 함수 역할을 뚜렷하게 나누는 것이 쉽지않다. 객체 지향 프로그램 설계할 때 시간이 오래 걸린다.
* 입/출력까지 모두 분리 하다 보니 코드양이 많아지고 시간이 오래 걸린다.

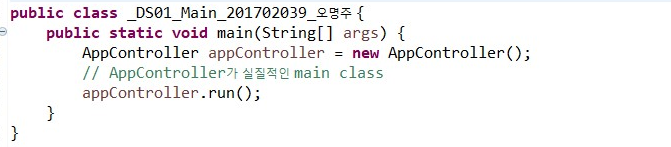
1. **실행 결과 분석**
2. **입력과 출력 (화면 capture하여 제출)**

****

1. **결과 분석 (자신의 논리적 평가, 기타 느낀 점)**

객체 지향 설계가 편리함은 이미 알고있었고, 가독성이나 재사용성이 높다는 것도 알고 있었으나 프로그램을 구현할 때 각각의 클래스나 함수가 어떠한 역할을 맡는지 명확하게 나누는 것이 나로서는 어려웠다. 하지만 주어진 클래스들을 분석하고 어떠한 역할을 하는지 알아보니 굉장히 잘 짜여진 프로그램이라는 생각이 들었고 그래서 분석하는데 어렵지는 않았던 것 같다. 여러가지 장점들 때문에 큰 프로그램을 설계할 때는 특히 더 MVC 모델을 이용하여 객체 지향 설계를 해야 할 것 같다는 생각이 들었다. 또한, 사전에 각 클래스와 함수 역할을 명확히 하여 구현을 해야 쉽게 할 수 있을 것 같았다. 마지막으로, Enum 클래스는 익숙하지 않아 잘 사용하지 않았는데 이번 기회에 다시한번 공부하게 되어 많이 알게 되어 유익했다.

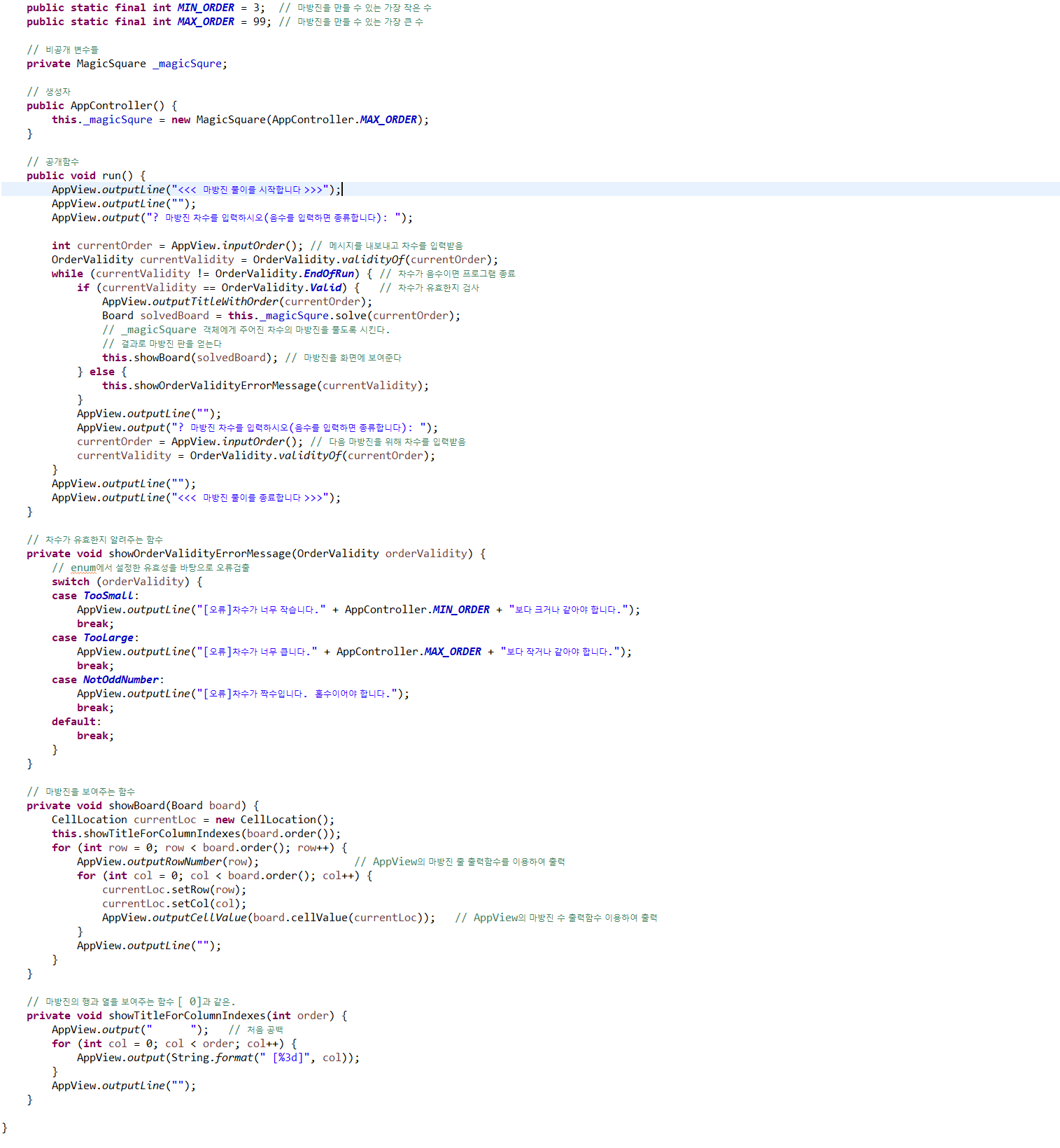
1. **소스코드**

****

[Main클래스]



[AppView 클래스]



[AppController 클래스]



[Board 클래스]



[CellLocation 클래스]



[OrderValidity 클래스]



[MagicSquare 클래스]