Debug

- 문제 풀 때마다 2차원 배열, 1차원 배열을 찍고 제출할 때 다시 지웠다가 제출하고 틀렸으면 다시 찍고 어쩔 때는 input파일 입력으로 넣었다가 파일 찾을 수 없다고 뜰 때마다 화가 나서 디버거용 클래스를 만들었습니다.
- 제가 사용하는 디버깅 용 프린트 경우의 수는 다 구현해 놓은 것 같습니다.

주요 기능

- `printArr()` : **배열을 출력해주는 메소드입니다.**
- `print()` : 문자열을 출력해주는 메소드입니다.
- `printHR()` : 특정 문자열로 구분선을 출력해주는 메소드입니다.
- `breakPoint()` : 아무런 기능도 없지만, debug 모드에서 브레이크 포인트를 잡기 위한 함수입니다.
- `set~()` : Debug클래스의 세팅을 하는 함수입니다.

사용

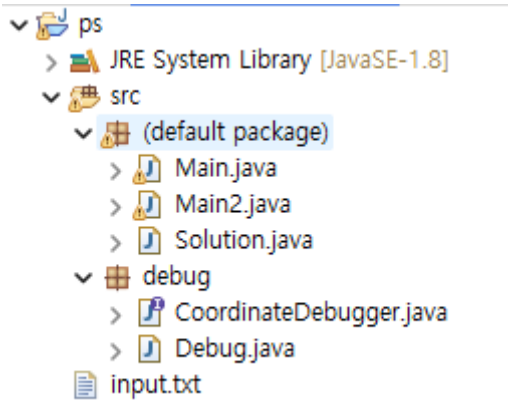
1. `debug.Debug` 를 import해줍니다.
2. `Debug` 클래스가 사용되기 전 먼저 `Debug.setDebug` 함수를 호출해줍니다.
    - `Debug.setDebug(boolean debug, [inputSetting])`
    - `boolean debug` 디버깅모드를 끄고 켤 수 있습니다. default : false
    - `inputSetting`
        - `boolean useInput` : 입력 파일로 입력을 받을지 설정합니다. default `false`
        - `String inputFile` : useInput을 true로 설정하고, inputFile 이름으로 가진 파일을 입력파일로 설정합니다. default `input.txt` (프로젝트 최상단 폴더)
3. 아래의 사용법을 참고하여 잘 디버깅합니다.
4. `setDebug()` 메서드를 활용해 디버깅을 껐다 켰다할 수 있습니다.
5. `ctrl+F` 를 눌러 `Find/Replace` 창을 열고, debug(대소구분 x)문자열이 들어간 열을 모두 공백으 대체하면 디버깅과 관련된 라인은 모두 사라집니다.

```
import debug.Debug;
```

```
...
public static void main(String[] args){
        Debug.setDebug(true, true);
        //Debug.setDebug(true, false);
        //Debug.setDebug(true, "input/customInput.txt");
        ...
}
```

설치

- `debug` 라는 패키지에 `Debug.java, CoordinateDebugger.java` 파일을 넣어주세요.
- inputfile을 사용하려면 프로젝트 최상단 폴더에 input.txt파일을 만들고 사용하면됩니다.



프로젝트 예시

printArr() 사용법

- `printArr` 함수는 다양한 배열과 출력방식을 지원합니다.
- `PRINT_WITH_HR` (default true) 값이 true일때 구분선으로 구분됩니다.
- `printArr(arr,[rowSize,colSize], [coordinate, chkChar], [ignore], [startText...])`

arr

- arr 매개변수는 출력할 대상이 될 매개변수입니다.
- 출력할 배열은 항상 가장 맨 앞에 순서로, 필수적으로 들어가야 합니다.
- printArr함수는 `int[][], int[], boolean[][], boolean[], char[][], char[]` 을 지원합니다.

size

- 출력할 배열의 크기를 지정할 수 있는 매개변수입니다.
- size는 `row, col` 순으로 같이 입력 돼야 합니다.
- 생략할 수 있는 변수이며 생략 시에는 arr의 전체를 출력합니다.
- 아래의 코드와 같다고 생각하면 됩니다.
- 2차원 배열의 경우,

```
for(int i=0; i<row; i++)
        for(int j=0; j<col; j++)
                print(arr[i][j]);
```

- 1차원 배열의 경우,

```
for(int i=start, i<end; i++)
        print(arr[i])
```

coordinate

- 특정 행, 열 위치의 출력을 대체하는 매개변수입니다.
- coordinate는 `coordinate, chkChar` 순으로 같이 입력 돼야 합니다.
- 생략할 수 있는 변수입니다.
- 좌표를 입력할 수 있는 방법은 총 4가지입니다.
  - (단일) 직접 좌표 : `int row, int col` 형태로 단 하나의 좌표를 chkChar로 대체합니다.
  - (단일) 클래스 좌표 : `CoordinateDebugger cor` 형태로 단 하나의 좌표를 chkChar로 대체합니다.
  - (복수) 배열 좌표 : `int[][] cors` 형태로 `[idx][row = 0, col = 1]` 저장된 배열의 위치들을 전부 chkChar로 대체합니다.
  - (복수) 클래스 배열 좌표 : `CoordinateDebugger[] cors` 형태로 좌표 클래스 내부의 좌표들을 전부 chkChar로 대체합니다.

coordinate 사용 예시

- 원본 배열

```
SIZE = 4
arr =
{
0 0 0 0
0 0 1 0
0 0 0 1
1 1 0 0
}

(단일) 가장 왼쪽 위에 존재하는 1을 *로 출력
0 0 0 0
0 0 * 0
0 0 0 1
1 1 0 0
(복수) (0,0) 좌표와, 1을 모두 X로 출력
X 0 0 0
0 0 X 0
0 0 0 X
X X 0 0
```

직접 좌표

- (단일) 직접좌표 :

```
// (단일) 가장 왼쪽 위에 존재하는 1을 *로 출력
int row = -1, col = -1;
for(int i=0; i<SIZE && row == -1; i++){
        for(int j=0; j<SIZE && col == -1; j++){
                if(arr[i][j] == 1){
                        row = i; col = j;
                }
        }
}
Debug.printArr(arr,row,col,'*');
Debug.printArr(arr,1,2,'*');
```

- (복수) 배열좌표 :

```
// (복수) (0,0) 좌표와, 1을 모두 X로 출력
int[][] cors = new int[5][2];
int cnt = 1;
for(int i=0; i<SIZE && row == -1; i++){
        for(int j=0; j<SIZE && col == -1; j++){
                if(arr[i][j] == 1){
                        cors[cnt][0] = i;
                        cors[cnt][1] = j;
                        cnt++;
                }
        }
}
Debug.printArr(arr,cors,'X');
```

클래스 좌표

```
// 예시 클래스
class CustomClass
implements CoordinateDebugger{ // CoordinateDebuger를 상속받아야합니다.
        int y, x;
        public CustomClass(int y, int x){
                this.y = y;
                this.x = x;
        }
        // Debuger에게 넘겨줄 행과 열을 제공할 함수를 오버라이드합니다.
        @Override
        public int getRow() {
                return y;
        }

        @Override
        public int getCol() {
                return x;
        }
}
```

- (단일) 클래스 좌표 :

```
// (단일) 가장 왼쪽 위에 존재하는 1을 *로 출력
int row = -1, col = -1;
for(int i=0; i<SIZE && row == -1; i++){
        for(int j=0; j<SIZE && col == -1; j++){
```

```
                if(arr[i][j] == 1){
                        row = i; col = j;
                }
        }
}
Debug.printArr(arr, new CustomClass(row, col),'*');

CustomClass cor = new CustomClass(row, col);
Debug.printArr(arr, cor,'*');
```

- (복수) 클래스 배열 좌표 :

```
// (복수) (0,0) 좌표와, 1을 모두 X로 출력
CustomClass[] cors = new CustomClass[5];
cors[0] = new CustomClass(0,0);

int cnt = 1;
for(int i=0; i<SIZE && row == -1; i++){
        for(int j=0; j<SIZE && col == -1; j++){
                if(arr[i][j] == 1){
                        cors[cnt++] = new CustomClass(i,j);
                }
        }
}
Debug.printArr(arr,cors,'X');
```

ignore

- 무시할 값을 지정합니다. 배열에 ignore과 같은 값이 있다면 배열의 값 대신 0을 출력합니다.
- 이 매개변수는 출력할 배열의 자료형이 `int` 일때만 사용할 수 있습니다.
- 생략가능하며 생략시 `IGNORE_MIN_MAX_VAL` (default true) 값이 true일때, `Integer.MAX_VALUE, MIN_VALUE`가 0으로 출력 됩니다.
- `setIgnoreMaxMin(boolean set)` 메서드로 `IGNORE_MIN_MAX` 값을 설정할 수 있습니다.

startText

- 배열을 출력하기 전에 출력할 문구입니다.
- 이 매개변수는 0개 이상 입력으로 사용될 수 있습니다.

- 2개 이상 입력될 경우, 공백문자로 구분되어 출력됩니다.

printHR() 사용법

- `printHR([char c], [int cnt])` 메서드는 특정 문자로 구분선을 출력해주는 메소드입니다.
- `char c` 는 생략 가능 하며, 생략시 `DEFUALT_HR_CHAR` (default `'*'`)이 출력됩니다.
- `int cnt` 는 생략 가능하며, 생략시 `DEFAULT_HR_CNT` (default `45`)개의 `c` 가 출력됩니다.
- `DEFAULT_HR_~` 변수들은 public이므로 직접 수정할 수 있습니다.