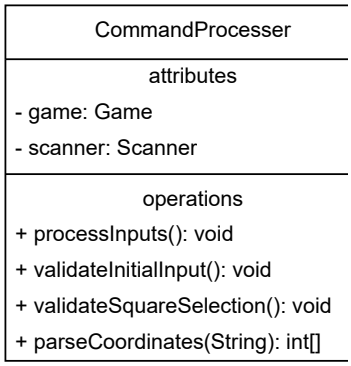
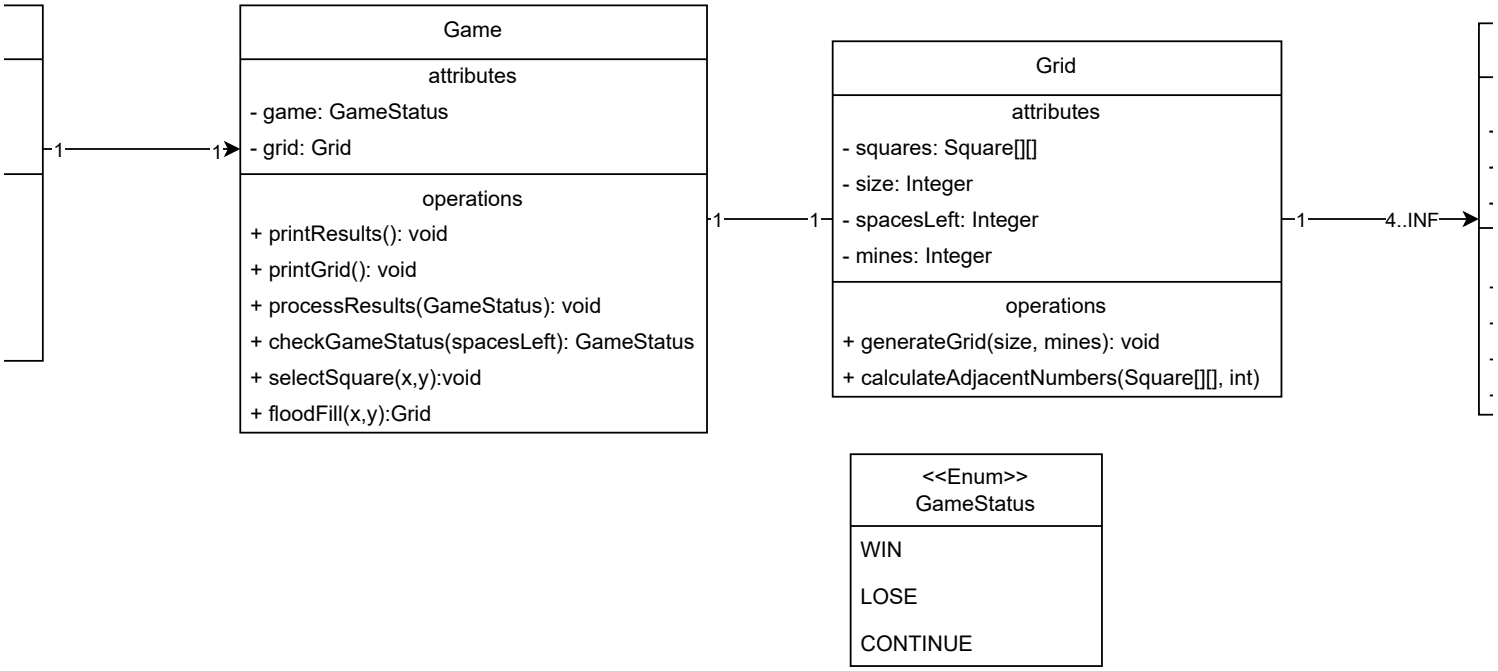
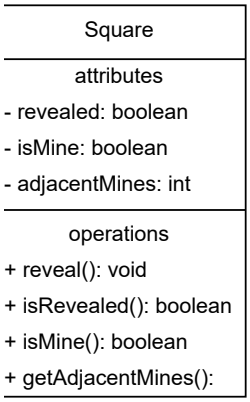

CommandProcesser
attributes
- game: Game
- scanner: Scanner
operations
+ processInputs(): void
+ validateInitialInput(): void
+ validateSquareSelection(): void
+ parseCoordinates(String): int[]


1
1
Game
attributes
- game: GameStatus
- grid: Grid
operations
+ printResults(): void
+ printGrid(): void
+ processResults(GameStatus): void
+ checkGameStatus(spacesLeft): GameStatus
+ selectSquare(x,y):void
+ floodFill(x,y):Grid
1
1
Grid
attributes
- squares: Square[][]
- size: Integer
- spacesLeft: Integer
- mines: Integer
operations
+ generateGrid(size, mines): void
+ calculateAdjacentNumbers(Square[][], int)
1
4..INF

<<Enum>>
GameStatus
WIN
LOSE
CONTINUE


Square
attributes
- revealed: boolean
- isMine: boolean
- adjacentMines: int
operations
+ reveal(): void
+ isRevealed(): boolean
+ isMine(): boolean
+ getAdjacentMines():

Problem statement: MineSweeper App
Write a program that simulates a Minesweeper game on a square grid.

- The game should begin by prompting the user for the grid size and the number of mines to be randomly placed on the grid.

- The program should then generate the grid and randomly place the specified number of mines on the grid.

- The user should then be prompted to select a square on the grid to uncover.
  - If the selected square contains a mine, the game is over and the user loses.
  - Otherwise, the selected square is uncovered and reveals a number indicating how many of its adjacent squares contain mines.
  - If an uncovered square has no adjacent mines, the program should automatically uncover all adjacent squares until it reaches squares that do have adjacent mines.

- The game is won when all non-mine squares have been uncovered.

- The program should display the game grid and allow the user to input their choices through a command line interface.

- Additionally, the program should track the user's progress throughout the game, displaying the minefield after each user input.

Game play

Sucess example
```
Welcome to Minesweeper!

Enter the size of the grid (e.g. 4 for a 4x4 grid):
4
Enter the number of mines to place on the grid (maximum is 35% of the total squares):
3

Here is your minefield:
  1 2 3 4
A _ _ _ _
B _ _ _ _
C _ _ _ _
D _ _ _ _
```

```
Select a square to reveal (e.g. A1): D4
This square contains 0 adjacent mines.

Here is your updated minefield:
  1 2 3 4
A _ _ 2 0
B _ _ 2 0
C _ 2 1 0
D _ 1 0 0

Select a square to reveal (e.g. A1): B1
This square contains 3 adjacent mines.

Here is your updated minefield:
  1 2 3 4
A _ _ 2 0
B 3 _ 2 0
C   2 1 0
```

#Requirements rewritten

1.Application is a minesweep
2.Application should be able
3.The user will be freely ab
4.The application should gen
5.The user will be able to i
6. The field will be filled
7. When the user selected a
8. When the user selected al
9. When the user selected a
10. Display the minefield af
11. Display only the reveale

er game.
to be read user inputs.
le to input grid sizes and mines count
erate a 2d mine field.
nput a X and a Y dimension to specify a location on the field
with mines and numbers.
mine on the field, the game is lost
l the grids, the game is won
'0' the surrounding 0 will be revealed + 1 in the border (Flood fill algo)?
ter every selection.
d fields.