GIN314 Project 1 – Group 4 – Snake Game

Presented to: Pascal DAMIEN

Presented by: Madeleine FARAH

Maher HALABI

Elias-Charbel SALAMEH

Github Repository: https://github.com/eliascharbelsalameh/SnakeGame-OOP

N.B: Kindly find in Page 5 the **UML Diagram**

Information:

Snake Game

This is a console-based implementation of the classic Snake Game in Java. The game is designed to provide a simple yet engaging experience by combining object-oriented principles with clean design. It includes features like food, obstacles, a movable snake, and collision detection.

Features

- Dynamic Game Map: A customizable grid-based environment.
- Snake Mechanics: The snake moves, grows when eating food, and loses upon collision.
- Interactive Gameplay: Players can control the snake using keyboard inputs (WASD).
- Collision Detection: The game detects interactions between the snake and food, obstacles, or itself.
- Win/Loss Conditions: The game ends when the snake consumes all food (win) or collides with an obstacle or itself (loss).

Class Descriptions

Core Classes

1. **Cell (Abstract Class):**
   Represents a generic pair of coordinates with x, y coordinates and overlap checking.
   - **Subclasses:** EmptyCell, Food, Obstacle, SnakeHead, SnakeBodyCell.

2. **EmptyCell:**
   Represents an empty space on the map ( ).
3. **Food:**
   Represents food items (F) that the snake consumes to grow.
4. **Obstacle:**
   Represents obstacles (O) that end the game upon collision.
5. **SnakeHead and SnakeBodyCell:**
   Represent the snake's head (S) and body (s).

Gameplay Classes

1. **Map:**
   Represents the game grid with:
   - Dynamic Grid Generation: Initializes an empty grid of dimensions (height, width).
   - Cell Manipulation: Allows setting and retrieving grid cells.
   - Game Display: Prints the map with updated snake positions based on the updated (override) toString() function.
2. **Snake:**
   - Controls Movement: Moves in four directions (UP, DOWN, LEFT, RIGHT).
   - Grows on Eating Food: Expands when consuming F.
   - Detects Self-Collision: Ends game if it collides with its own body.
3. **Game:**
   - Handles Player Input: Uses Scanner to read WASD commands.
   - Checks Collisions: Detects interactions between snake and obstacles, food, or boundaries.
   - Manages Game Flow: Starts, updates, and terminates the game based on win/loss conditions.
4. **Driver:**
   - Initializes Game Objects: Sets up the Map, Snake, and Game.
   - Starts the Game: Calls game.play() to begin the gameplay loop.

How to Run

1. **Requirements:**
   - **Java Development Kit (JDK) installed.**
   - **A terminal or IDE to run the program.**
2. **Steps:**
   - **Compile all Java files:**

     ```
     javac *.java
     ```
   - **Run the Driver class:**

     ```
     java Driver.java
     ```

How to Play

- **Use WASD keys to control the snake:**
  - W - Move Up
  - A - Move Left
  - S - Move Down
  - D - Move Right
- **Consume food (F) to grow.**
- **Avoid obstacles (O) and self-collision.**
- **The game ends when:**
  - The snake consumes all food (You Win!).
  - The snake collides with itself or an obstacle (Game Over!).

Future Enhancements

- Add a graphical user interface (GUI) for a better visual experience.
- Improve input validation for smoother gameplay.

- Implement a score tracking system.
- Enhance collision detection.
- Add exception handling.

Project Files

- **Core Classes:**
    - Cell.java - Base abstract class for all cells.
    - EmptyCell.java - Represents an empty cell.
    - Food.java - Represents food cells.
    - Obstacle.java - Represents obstacle cells.
    - SnakeHead.java - Represents the snake's head.
    - SnakeBodyCell.java - Represents the snake's body.
- **Gameplay Classes:**
    - Game.java - Manages gameplay and game states.
    - Map.java - Handles grid initialization and rendering.
    - Snake.java - Implements snake movement and growth.
- **Driver:**
    - Driver.java - Initializes and starts the game.

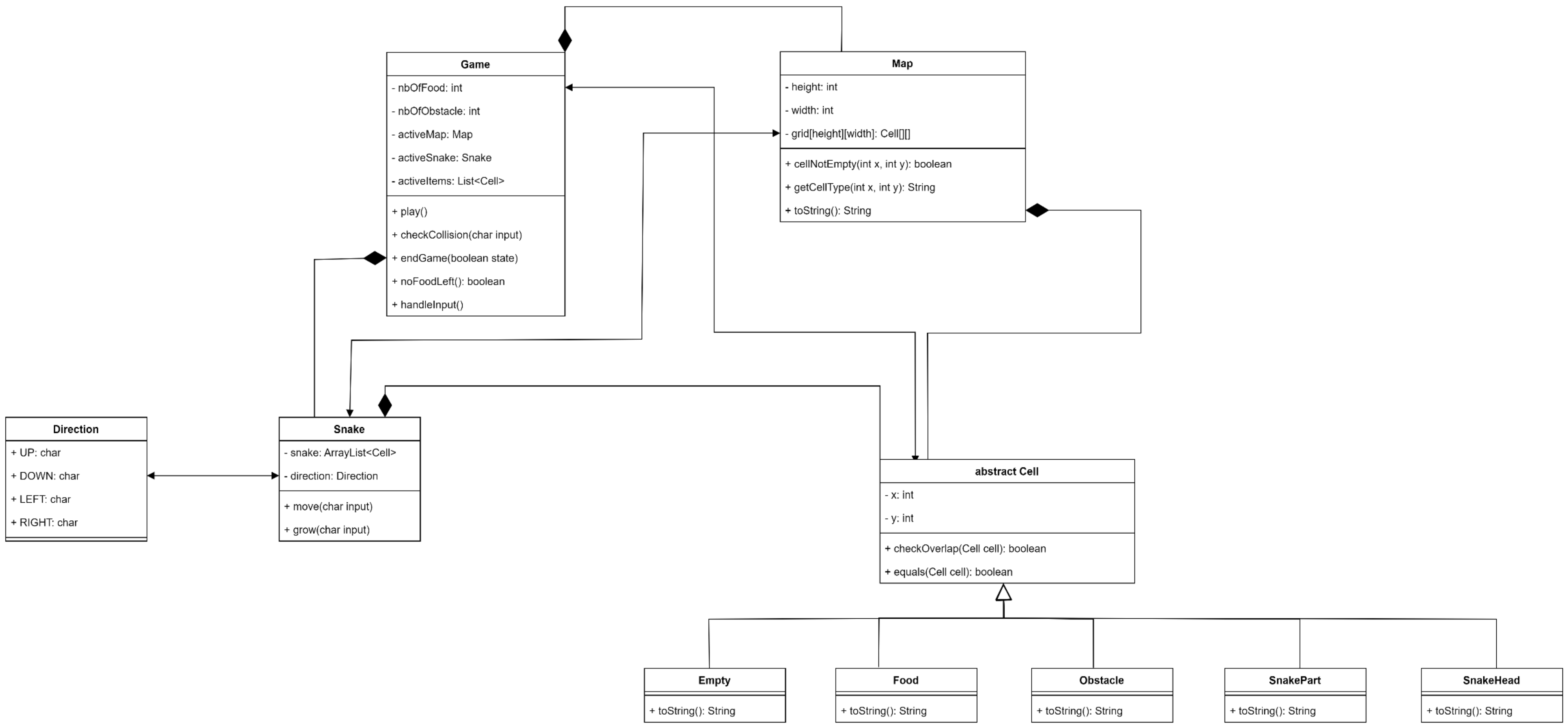
Authors

Developed as part of an educational project in the Object Oriented Analysis and Design class at the Holy Spirit University of Kaslik - USEK - in Lebanon.

- Madeleine Farah
- Maher Halabi
- Elias-Charbel Salameh

Feel free to contribute or modify the game as needed!

UML Diagram:


Game
- nbOfFood: int
- nbOfObstacle: int
- activeMap: Map
- activeSnake: Snake
- activeItems: List<Cell>
+ play()
+ checkCollision(char input)
+ endGame(boolean state)
+ noFoodLeft(): boolean
+ handleInput()
Map
- height: int
- width: int
- grid[height][width]: Cell[][]
+ cellNotEmpty(int x, int y): boolean
+ getCellType(int x, int y): String
+ toString(): String
Direction
+ UP: char
+ DOWN: char
+ LEFT: char
+ RIGHT: char
Snake
- snake: ArrayList<Cell>
- direction: Direction
+ move(char input)
+ grow(char input)
abstract Cell
- x: int
- y: int
+ checkOverlap(Cell cell): boolean
+ equals(Cell cell): boolean
Empty
+ toString(): String
Food
+ toString(): String
Obstacle
+ toString(): String
SnakePart
+ toString(): String
SnakeHead
+ toString(): String