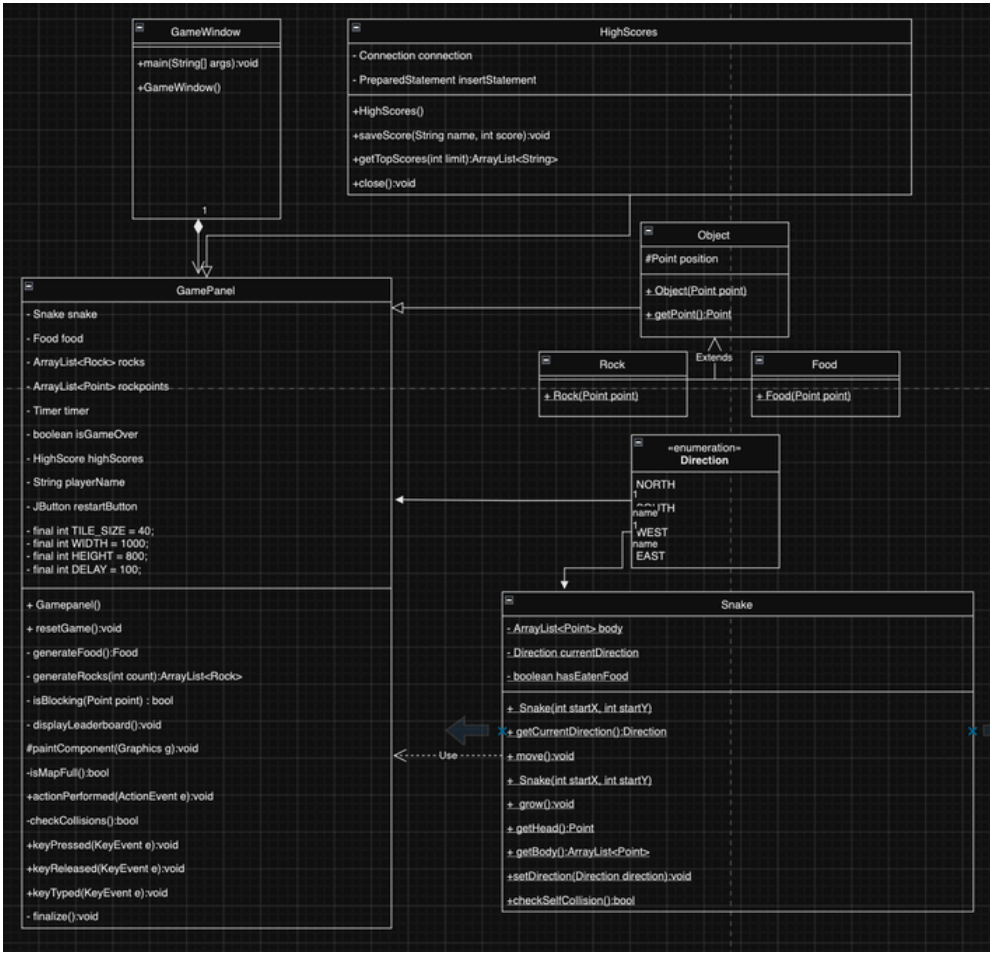
Name : Hajiyev Hajiaga **3 assignment 2nd task** **11 December 2024**

Naptun code: C499UD

Task

We have a rattlesnake in a desert, and our snake is initially two units long (head and rattler). We have to collect with our snake the foods on the level, that appears randomly. Only one food piece is placed randomly at a time on the level (on a field, where there is no snake).The snake starts off from the center of the level in a random direction. The player can control the movement of the snake's head with keyboard buttons. If the snake eats a food piece, then its length grow by one unit. It makes the game harder that there are rocks in the desert. If the snake collides with a rock, then the game ends. We also lose the game, if the snake goes into itself, or into the boundary of the game level.
In these situations show a popup messagebox, where the player can type his name and save it together with the amount of food eaten to the database. Create a menu item, which displays a highscore table of the players for the 10 best scores. Also, create a menuitem which restarts the game.

UML Class Diagram



Description of major methods

The methods in GameWindow

1. GameWindow()

Initializes the game window with a title, adds the game panel, prevents resizing, centers the window, and adds a confirmation dialog for exiting.

2. public static void main(String[] args)

Launches the game window on the Event Dispatch Thread using SwingUtilities.invokeLater.

The methods in GamePanel

1. GamePanel()

Sets up the game panel's size, background, and layout, initializes game elements (snake, food, rocks), and adds a restart button with a confirmation dialog.

2. resetGame()

Resets the game state, including the snake's position, food, rocks, and game-over status. Restarts the game timer and prepares for a new game.

3. generateFood()

Creates a new food object at a random position that does not overlap with the snake or rocks.

4. generateRocks(int count)

Generates a specified number of rocks at random positions, avoiding collisions with the snake and ensuring rocks do not block the snake's immediate path.

5. isBlocking(Point point)

Checks if a given point obstructs the snake's movement within five tiles in the current direction.

6. displayLeaderboard()

Retrieves and displays the top 10 high scores in a dialog box using the HighScores class.

7. paintComponent(Graphics g)

Renders the game, including the snake, food, and rocks. If the game is over, it stops the timer, saves the player's score, and shows the leaderboard.

8. isMapFull()

Checks if the entire map is filled with the snake, rocks, and food, indicating a win condition.

9. actionPerformed(ActionEvent e)

Updates the game state at each timer tick, moves the snake, checks collisions, and handles interactions like eating food or game-over conditions.

10. checkCollisions()

Detects collisions with walls, the snake itself, or rocks.

11. keyPressed(KeyEvent e)

Changes the snake's direction based on player input (WASD keys).

12. keyReleased(KeyEvent e)

Not used in this implementation.

13. keyTyped(KeyEvent e)

Not used in this implementation.

14. finalize()

Ensures proper closure of resources, including the high scores database connection.

The methods in Object

1. Object(Point point)

Initializes the object's position using the given point.

2. getPoint()

Returns the current position of the object.

The methods in Food

1. Food(Point point)

Initializes the food's position by inheriting the behavior of the Object class.

The methods in Rock

1. Rock(Point point)

Initializes the rock's position by inheriting the behavior of the Object class.

The methods in Snake

1. ArrowKeyListener(BoardGUI boardGUI): Constructor that initializes the key listener with a reference to the main game board GUI class.
2. keyPressed(KeyEvent e): Handles key presses, checks the current spaceship position, calculates movement direction based on arrow key input, and calls moveSpaceship() to update the position if the move is valid. Updates the game state by incrementing the turn and refreshing the display.
3. keyReleased(KeyEvent e): Empty method, required by KeyListener, but not used in this implementation.
4. keyTyped(KeyEvent e): Empty method, required by KeyListener, but not used in this implementation.

Connections between the events and event handlers.

Player Movement
Event: Keyboard key press (W, A, S, D).
Handler: keyPressed(KeyEvent e) (in GamePanel).
Method: snake.setDirection(Direction direction)
Updates the snake's movement direction based on key input.

Exit Confirmation
Event: User closes the game window.
Handler: windowClosing(WindowEvent e) (in GameWindow).
Method: Prompts the user to confirm exit before closing the game.
.

Food Consumption
Event: Snake's head collides with the food's position.
Handler: actionPerformed(ActionEvent e) (in GamePanel).
Method: snake.grow()
Increases the snake's length and generates a new food object.

Game Rendering
Event: Timer triggers every 100ms.
Handler: actionPerformed(ActionEvent e) (in GamePanel).
Method: paintComponent(Graphics g)
Renders the game elements, including the snake, food, and rocks, updating the screen.

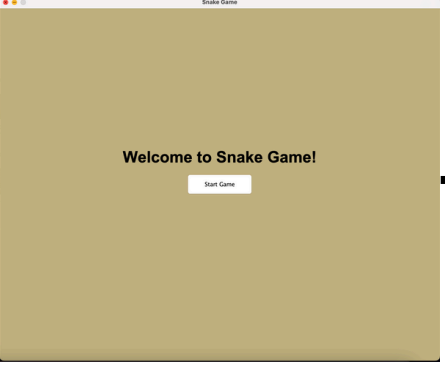
Collision Detection
Event: Timer triggers every 100ms.
Handler: checkCollisions() (in GamePanel).
Method: Checks for collisions with walls, rocks, or the snake itself to determine if the game is over.

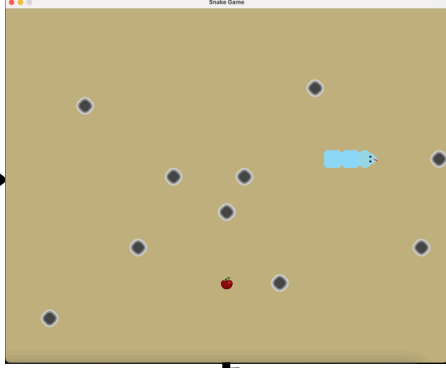
Rock Generation
Event: Game reset or initialization.
Handler: generateRocks(int count) (in GamePanel).
Method: Adds random rock positions on the grid, ensuring they do not block the snake's path or overlap with food.

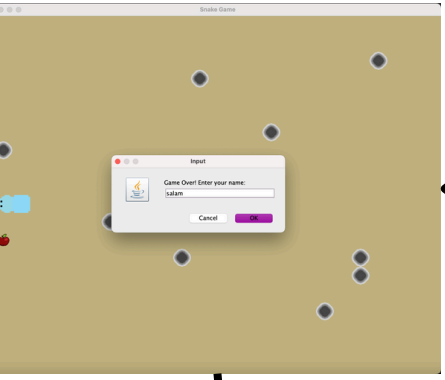
Leaderboard Display
Event: Game over after collision or map completion.
Handler: displayLeaderboard() (in GamePanel).
Method: Retrieves and shows the top scores using HighScores.getTopScores().

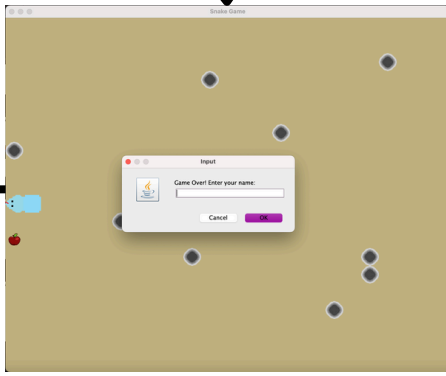
Game Reset
Event: User clicks "Restart Game" button.
Handler: resetGame() (in GamePanel).
Method: Resets snake position, food, rocks, and game state, restarting the game.

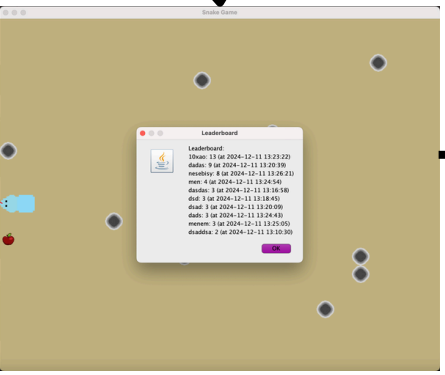
High Score Save
Event: Game over and player inputs name.
Handler: saveScore(String name, int score) (in HighScores).
Method: Stores the player's name and score in the database.

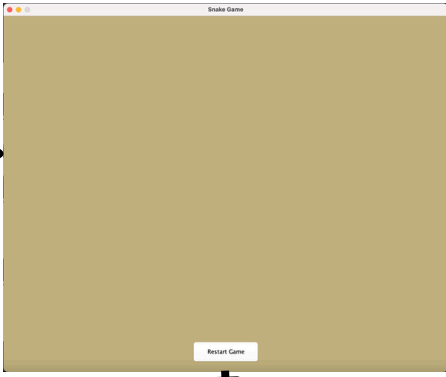

Snake Game
Welcome to Snake Game!
Start Game

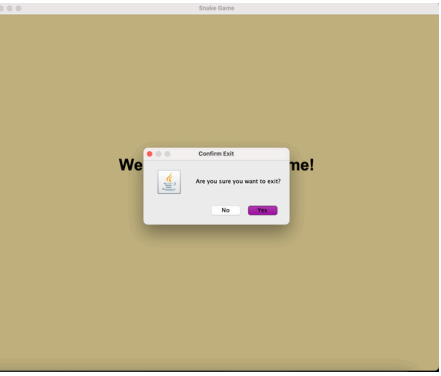

Snake Game

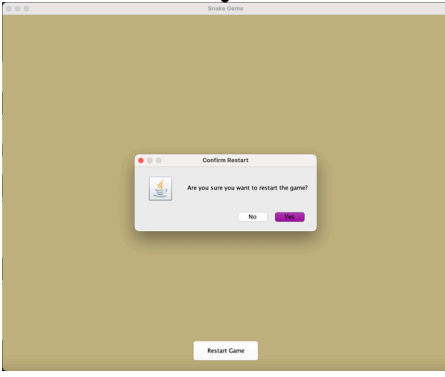

Snake Game
Input
Game Over! Enter your name:
salam
Cancel
OK


Snake Game
Input
Game Over! Enter your name:
Cancel
OK


Snake Game
Leaderboard
Leaderboard:
10xao: 13 (at 2024-12-11 13:23:22)
dadas: 9 (at 2024-12-11 13:20:39)
neseblsy: 8 (at 2024-12-11 13:26:21)
men: 4 (at 2024-12-11 13:24:54)
dasdas: 3 (at 2024-12-11 13:16:58)
dsd: 3 (at 2024-12-11 13:18:45)
dsad: 3 (at 2024-12-11 13:20:09)
dads: 3 (at 2024-12-11 13:24:43)
menem: 3 (at 2024-12-11 13:25:05)
dsaddsa: 2 (at 2024-12-11 13:10:30)
OK


Snake Game
Restart Game


Snake Game
Confirm Exit
Are you sure you want to exit?
No
Yes


Snake Game
Confirm Restart
Are you sure you want to restart the game?
No
Yes
Restart Game