

Rattlesnake adventure

Documentation

Made by: Illia Takhtamyshev

Contents

Task	5
Class diagram	6
Short description of important methods.....	10
Snake	10
1. Method “public BufferedImage getIconForPart(int partInd)”	10
2. Method “public void move(int cellSize)”	10
GameManager	10
1. Method “private void initFruits()”	10
2. Method “public Boolean checkCollision()”	10
Coordinate	10
1. Constructor “public int getX()”	10
2. Method “public void setX(int x)”	10
GameObject	11
1. Method “public void genRandCoord(int cellSize, int width, int height, ArrayList<Coordinate> snakeParts, int areaSize)”	11
2. Method “private Boolean checkSnakeOverlap(int cellSize, ArrayList<Coordinate> snakeParts, int areaSize)”	11
Apple (same for Banana, Cherry, Rock).....	11
1. Construcor “public Apple(int cellSize, int width, int height, ArrayList<Coordinate> snakeParts, int areaSize)”	11
Window	11
1. Construcor “public Window()”	11
MainWindow	11
1. Construcor “public MainWindow(int cellSize, int width, int height, int rocksCount)”	11
Top10Window	12
1. Construcor “public Top10Window(ArrayList<Window> windows, DatabaseManager databaseManager)”	12

2. Method “private DefaultTableModel createTableModel()”	12
GameWindow	12
1. Construcor “public GameWindow(ArrayList<Window> windows, DatabaseManager databaseManager, int cellSize, int width, int height, int rocksCount)”	12
2. Method “public void displayGameEnd()”	12
MenuBar.....	12
1. Construcor “public MenuBar(ArrayList<Window> windows, DatabaseManager databaseManager)”	12
2. Method “private ActionListener showTop10List()”	12
GamePanel	13
1. Construcor “public GamePanel(GameManager gameManager, GameWindow gameWindow)”	13
2. Method “public void drawBoard(Graphics g)”	13
DatabaseManager.....	13
1. Method “public void saveScore(String playerName, int playerScore)” ..	13
2. Method “public ArrayList<String> readTop10Players()”	13
Connections between the events and event handlers	14
MainWindow	14
Event: Button Click.....	14
MenuBar.....	14
Event: Menu Item Click.....	14
Event: Menu Item Click.....	14
Event: Menu Item Click.....	14
GamePanel	15
Event: Timer Tick	15
Event: Key Press	15
Testing.....	16
1. Test eating fruits	16
Description:	16

Result:	16
2. Test board collision	17
Description:	17
Result:	17
3. Test rock collision	18
Description:	18
Result:	18
4. Test snake collision	19
Description:	19
Result:	19
5. Test restart button	20
Description:	20
Result:	20
6. Test writing to the database	21
Description:	21
Result:	21
7. Test listing top 10 players	22
Description:	22
Result:	22

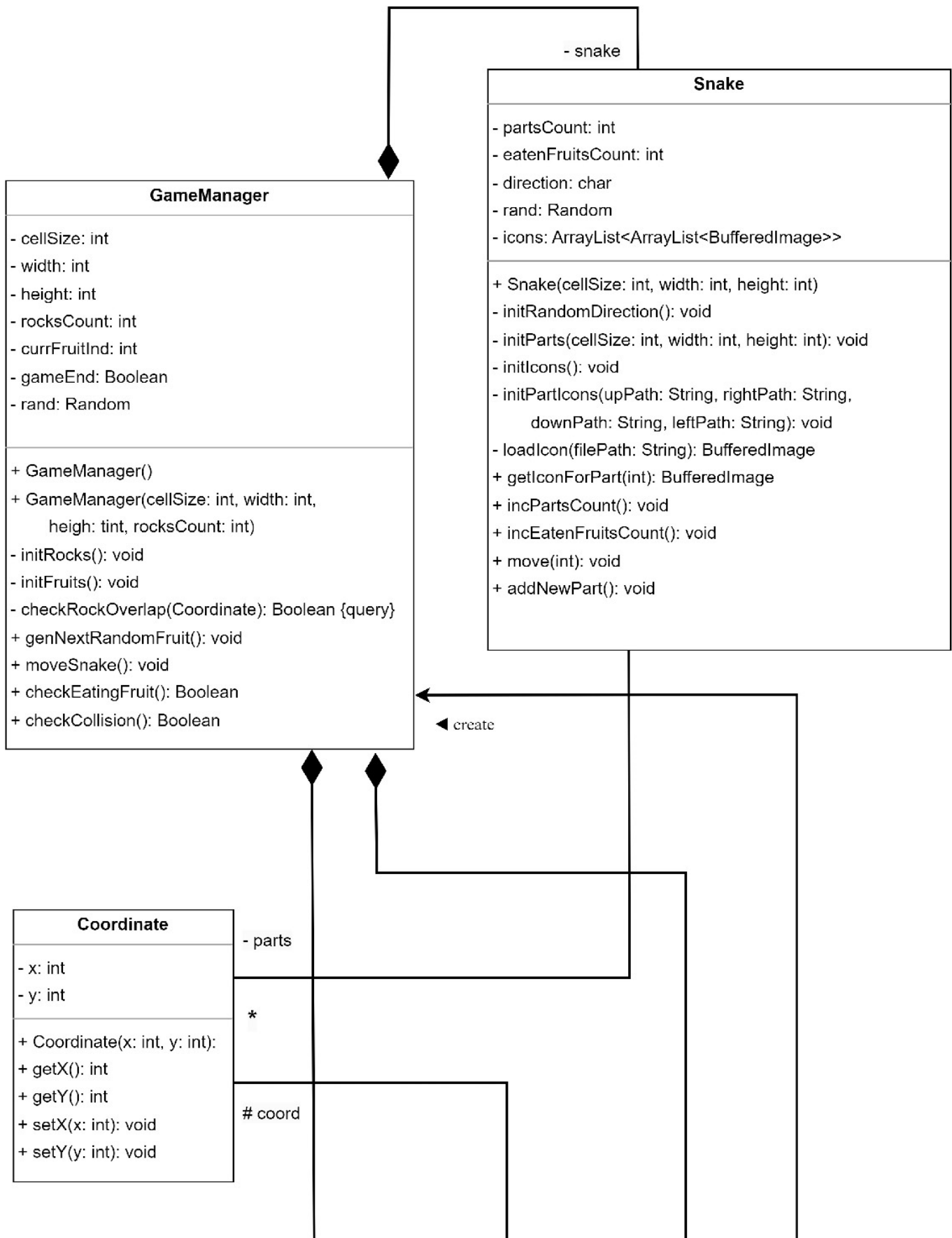
Task

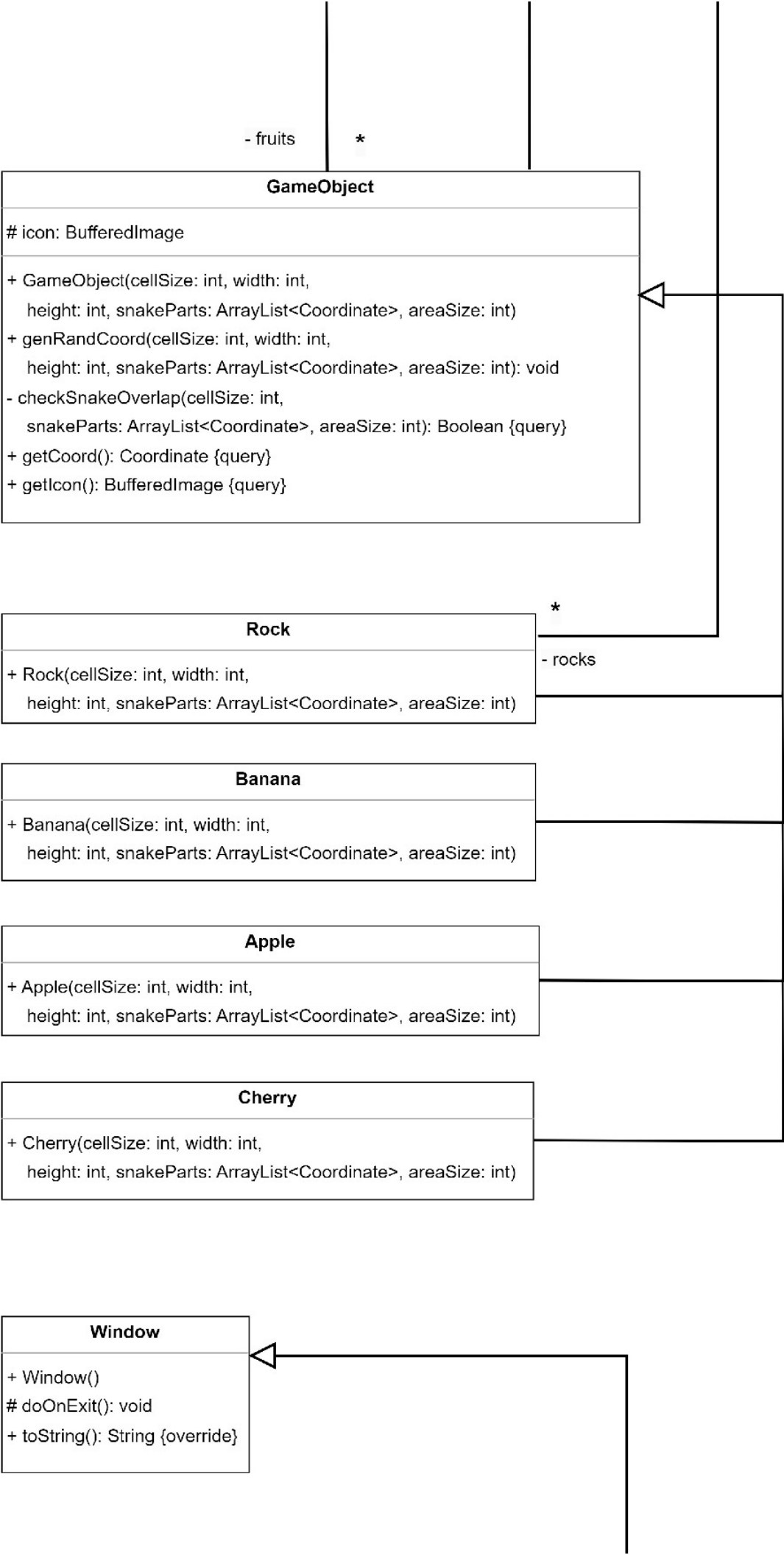
There is a rattlesnake in a desert, and the snake is initially two units long (head and rattler). You have to collect with the snake the foods on the level, that appears randomly. Only one food piece is placed randomly at a time on the level (on a field, where there is no snake). The snake starts off 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 grows by one unit.

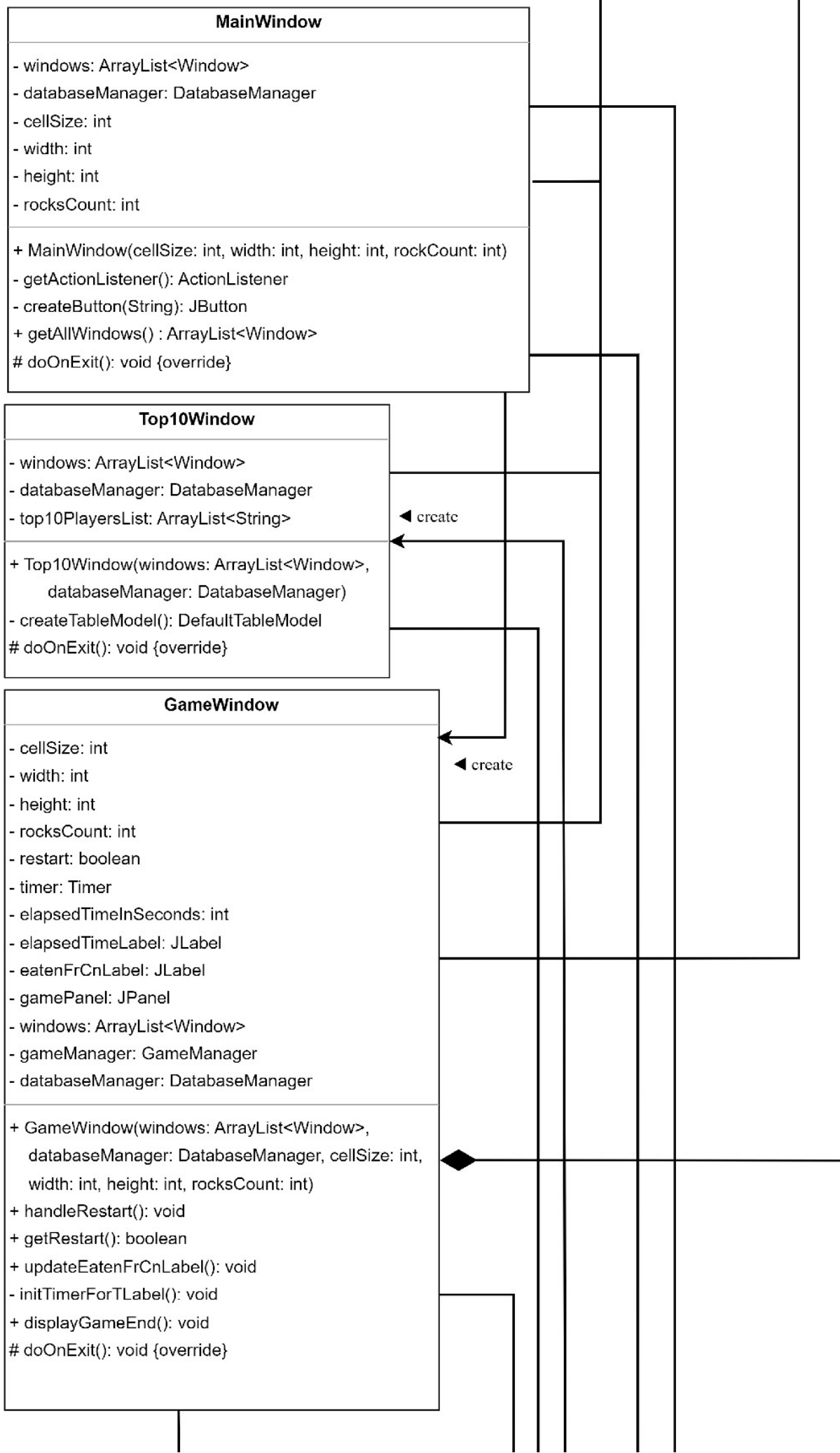
It makes the game harder that there are rocks in the desert. If the snake collides with a rock, then the game ends. We also lose the game, if the snake goes into itself, or into the boundary of the game level.

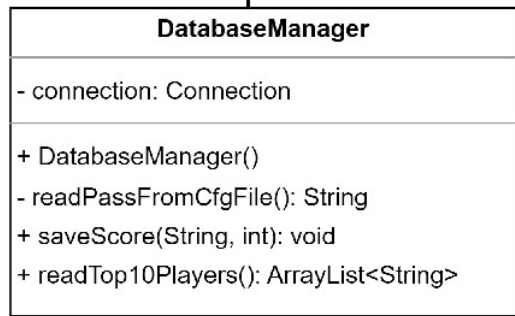
In these situations shows a popup message, where the player can type his name and save it together with the amount of food eaten to the database. There is a menu item, which displays a highscore table of the players for the 10 best scores. Also, a menu item which restarts the game.

Class diagram

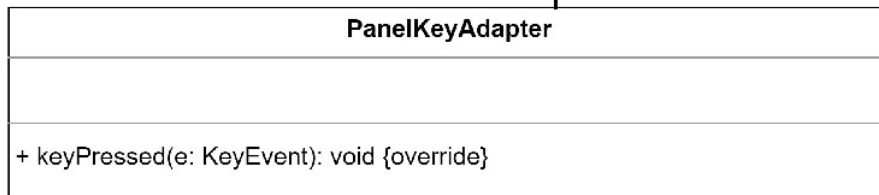
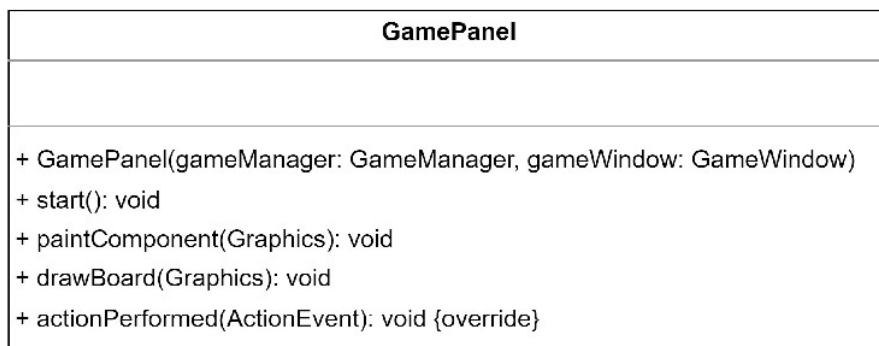
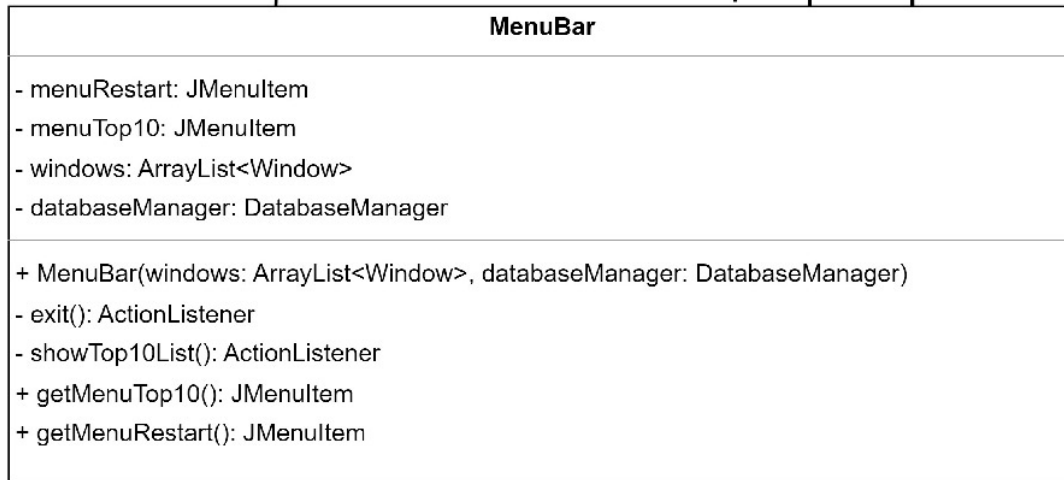








- menuBar



Short description of important methods

Snake

1. Method “**public BufferedImage getIconForPart(int partInd)**”

- This method returns the icon associated with a specific part of the snake based on its index. It allows different parts of the snake to be represented by distinct images.

2. Method “**public void move(int cellSize)**”

- It is responsible for updating the position of the snake based on its current direction. It moves the snake forward by one cell size in the direction it is facing, facilitating its continuous motion in the game.

GameManager

1. Method “**private void initFruits()**”

- This method initializes the fruits in the game. It generates instances of different fruit objects (Apple, Banana, Cherry) at random positions on the game board, ensuring they do not overlap with rocks or the snake. The selected current fruit index determines the type of fruit currently available for the snake to eat.

2. Method “**public Boolean checkCollision()**”

- This method is responsible for determining if a collision has occurred in the game. It checks for collisions with the game board boundaries, rocks, and the snake itself. If any collision is detected, it sets the game's end state to true, indicating that the game should conclude.

Coordinate

1. Constructor “**public int getX()**”

- This method returns the current x-coordinate value of the coordinate object.

2. Method “**public void setX(int x)**”

- This method sets the x-coordinate value of the coordinate object to the specified integer value x.

GameObject

1. Method “**public void genRandCoord(int cellSize, int width, int height, ArrayList<Coordinate> snakeParts, int areaSize)**”
 - This method generates a random coordinate for the game object, taking into consideration the cell size, game board dimensions (width and height), a list of occupied coordinates (to avoid overlap with other objects), and a minimum distance to ensure a safe placement.
2. Method “**private Boolean checkSnakeOverlap(int cellSize, ArrayList<Coordinate> snakeParts, int areaSize)**”
 - Method is responsible for determining whether the current game object overlaps with any part of the snake. It takes the snake's parts as input and returns a boolean value indicating whether there is overlap or not

Apple (same for Banana, Cherry, Rock)

1. Construcor “**public Apple(int cellSize, int width, int height, ArrayList<Coordinate> snakeParts, int areaSize)**”
 - Initializes Apple object by calling the constructor of the base class. It attempts to load the apple image from a specified file path. In case of the image not being found or an IOException, an error message is printed to the standard error.

Window

1. Construcor “**public Window()**”
 - Sets up a basic JFrame for the game application, configuring the title, size, and location, and handling window-closing events.

MainWindow

1. Construcor “**public MainWindow(int cellSize, int width, int height, int rocksCount)**”
 - Initializes the main window for the game, creating a JFrame with a menu bar, welcome label, and a "Start a new game" button. It also associates action listeners to respond to user interactions.

Top10Window

1. Constructor “**public Top10Window(ArrayList<Window> windows, DatabaseManager databaseManager)**”
 - Initializes the window for displaying the top 10 scores. It sets up the window properties, such as size and layout, and initializes components like the table and buttons. Additionally, it establishes connections to the database manager to fetch and display the top 10 scores.
2. Method “**private DefaultTableModel createTableModel()**”
 - Responsible for generating the table model to populate the top 10 scores. It retrieves score data from the database manager and structures it into a format suitable for display in a table.

GameWindow

1. Constructor “**public GameWindow(ArrayList<Window> windows, DatabaseManager databaseManager, int cellSize, int width, int height, int rocksCount)**”
 - Sets up the game window, including the menu bar, top panel with labels, and the game panel. It also initializes essential game components such as the GameManager, timer, and database manager.
2. Method “**public void displayGameEnd()**”
 - Handles the game-over scenario by stopping the timer, prompting the user to enter their name, saving the score to the database, and finally disposing of the window.

MenuBar

1. Constructor “**public MenuBar(ArrayList<Window> windows, DatabaseManager databaseManager)**”
 - Initializes the menu bar and its components, including menu items like "New Game," "Top 10 Scores," and "Exit." It sets up action listeners for these items to respond to user interactions.
2. Method “**private ActionListener showTop10List()**”
 - Is responsible for displaying the top 10 scores. It retrieves the scores from the database manager, creates a dialog box to show the scores in a user-friendly format, and then displays the dialog to the user.

GamePanel

1. Constructor “**public GamePanel(GameManager gameManager, GameWindow gameWindow)**”
 - Initializes an instance of the game panel. It configures the panel's properties, sets up the layout, and initializes various components required for the game display.
2. Method “**public void drawBoard(Graphics g)**”
 - Is responsible for rendering the game board. It takes care of drawing the snake, fruits, and other game objects on the panel based on their current positions and states. This method is crucial for updating the visual representation of the game during its execution.

DatabaseManager

1. Method “**public void saveScore(String playerName, int playerScore)**”
 - Is responsible for storing the player's score in the database. It takes the player's name and score as parameters and updates the database with this information.
2. Method “**public ArrayList<String> readTop10Players()**”
 - Returns the top 10 players' scores from the database. It returns a data structure, such as a list or an array, containing information about the highest-scoring players, allowing the application to display or utilize this data, such as showing a leaderboard.

Connections between the events and event handlers

MainWindow

Event: Button Click

- Buttons: “startButton”
- Event Handler: “private ActionListener getActionListener()”
 - When the "Start a new game" button (startButton) is clicked, it triggers the actionPerformed method associated with the ActionListener returned by the getActionListener() method.

MenuBar

Event: Menu Item Click

- Buttons: ”menuRestart”
- Event Handler: “public void handleRestart()” from GameWindow
 - It is given in GameWindow
 - The method sets the restart flag to true, stops the timer, and creates a new instance of the GameWindow to restart the game.

Event: Menu Item Click

- Buttons: “menuTop10”
- Event Handler: ”private ActionListener showTop10List()”
 - When the "Top10 scoreboard" menu item (menuTop10) is clicked, it triggers the actionPerformed method associated with the ActionListener returned by the showTop10List() method.

Event: Menu Item Click

- Buttons: “menuExit”
- Event Handler: ”private ActionListener exit()”
 - When the menu item is clicked, exit action is performed

GamePanel

Event: Timer Tick

- Timer: "timer"
- Event Handler: "public void actionPerformed(ActionEvent e)"
 - The timer (timer) in the GamePanel class triggers the actionPerformed method at regular intervals (controlled by the delay value).
 - In the actionPerformed method, the game logic is updated, including moving the snake, checking for collisions, and updating the display.
 - If the game ends, it stops the timer and, if not in restart mode, displays the game over screen.

Event: Key Press

- KeyAdapter: "PanelKeyAdapter"
- Event Handler: "public void actionPerformed(ActionEvent e)"
 - The GamePanel class has a KeyAdapter (PanelKeyAdapter) that listens for key presses.
 - When a key is pressed, the corresponding keyPressed method is called.
 - Depending on the pressed key (W, A, S, D, Arrow keys), the snake's direction in the game is updated accordingly.

Testing

1. Test eating fruits

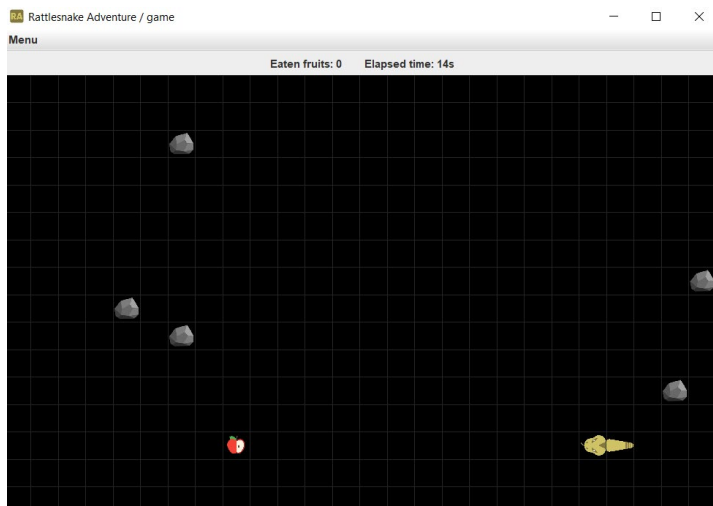
Description:

When snake eats the fruit, it is expected that the snake will get one more body part and fruit count will increment by one.

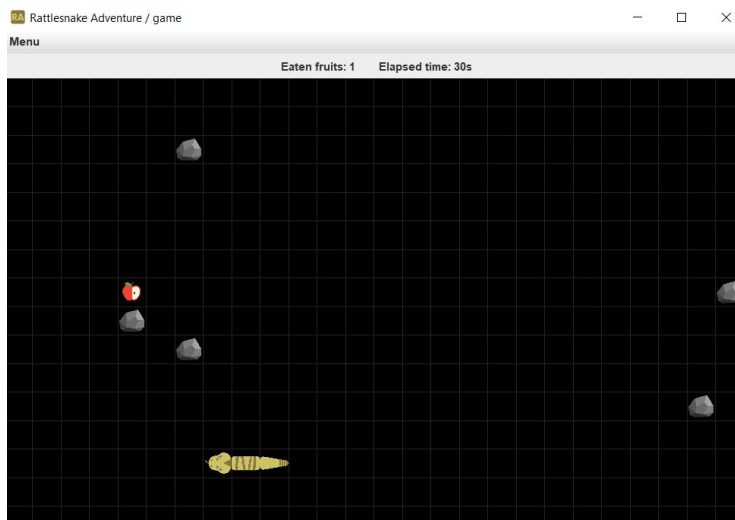
Result:

After eating the fruit, snake added one more body part and fruit count incremented by one as expected.

Before eating the fruit:



After eating the fruit:



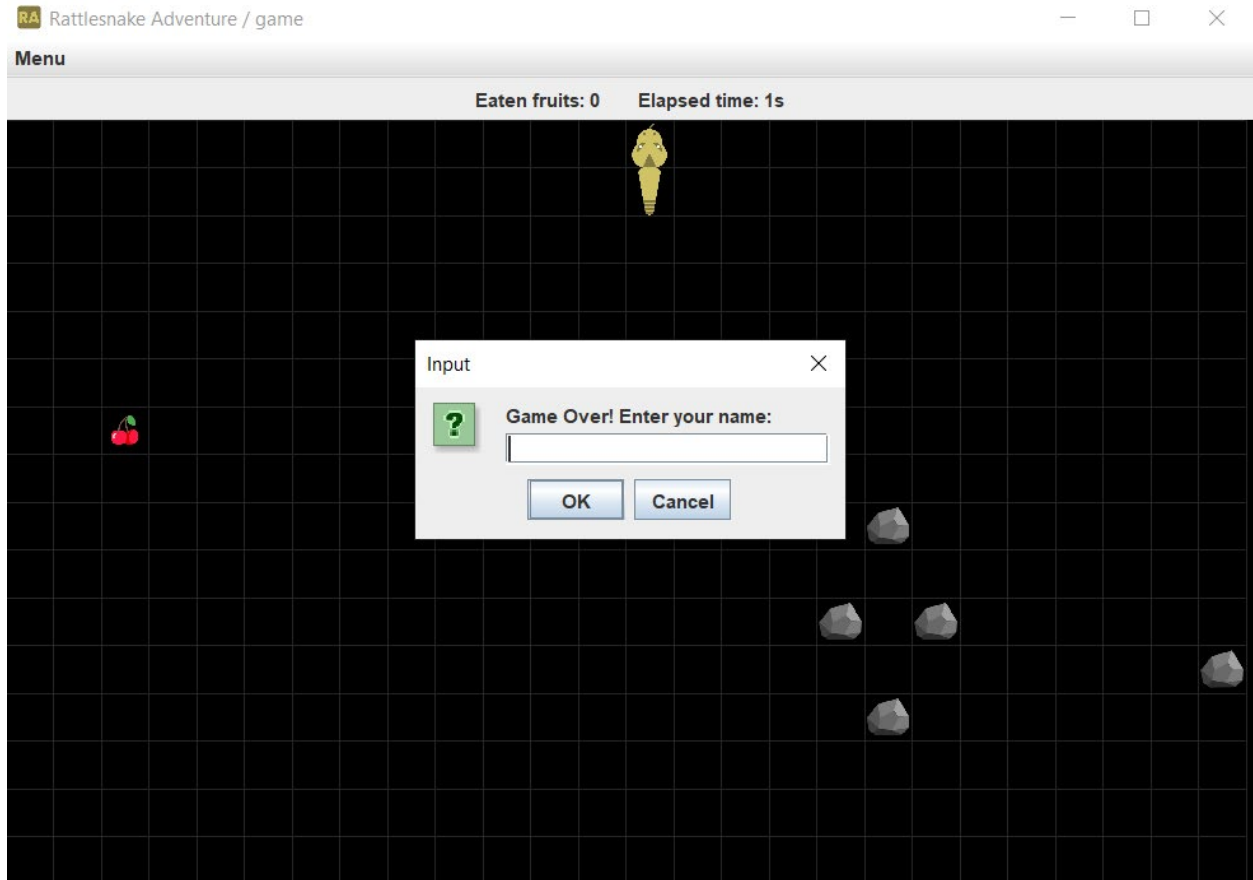
2. Test board collision

Description:

When snake bumps into the board end it is expected that the game will end.

Result:

After bumping into the board end, the game ended as expected.



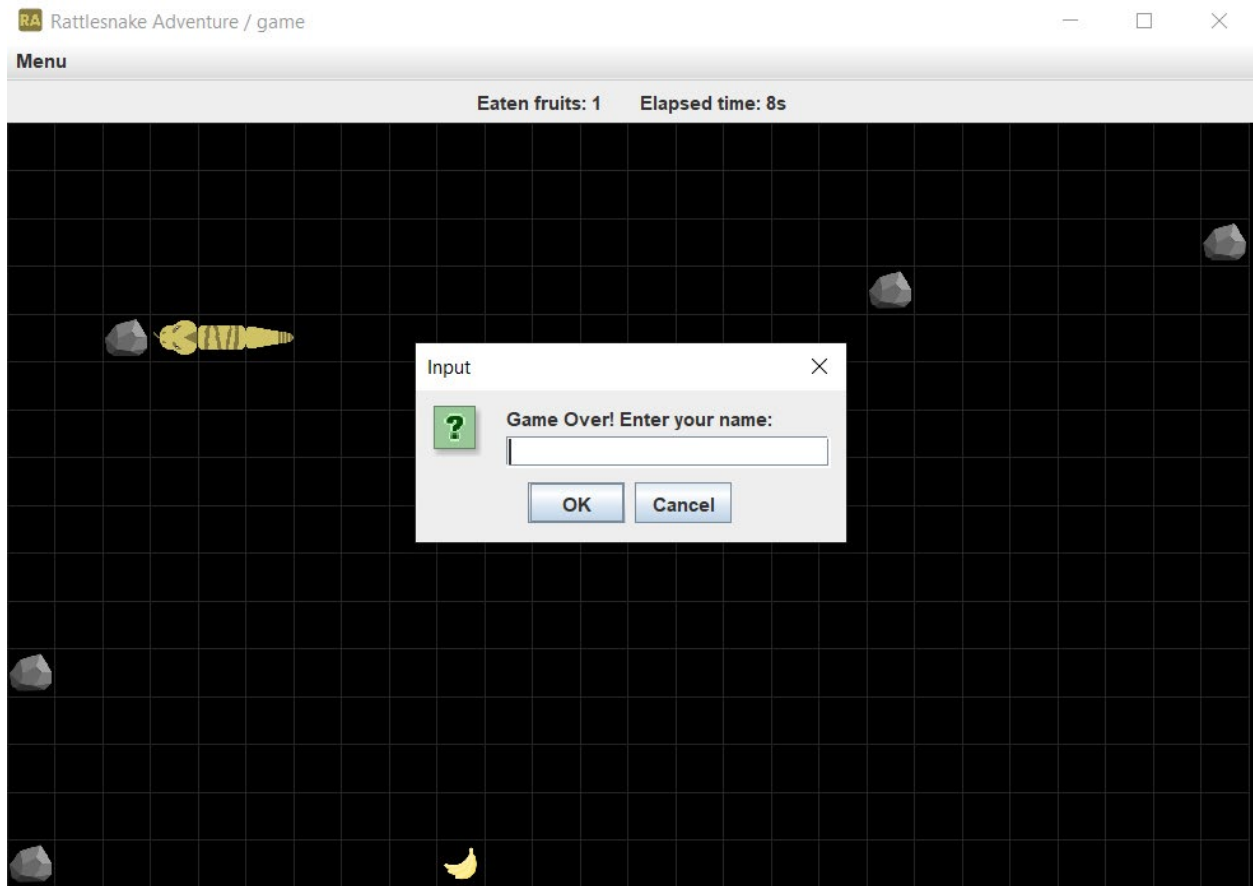
3. Test rock collision

Description:

When snake bumps into the rock it is expected that the game will end.

Result:

After bumping into the rock, the game ended as expected.



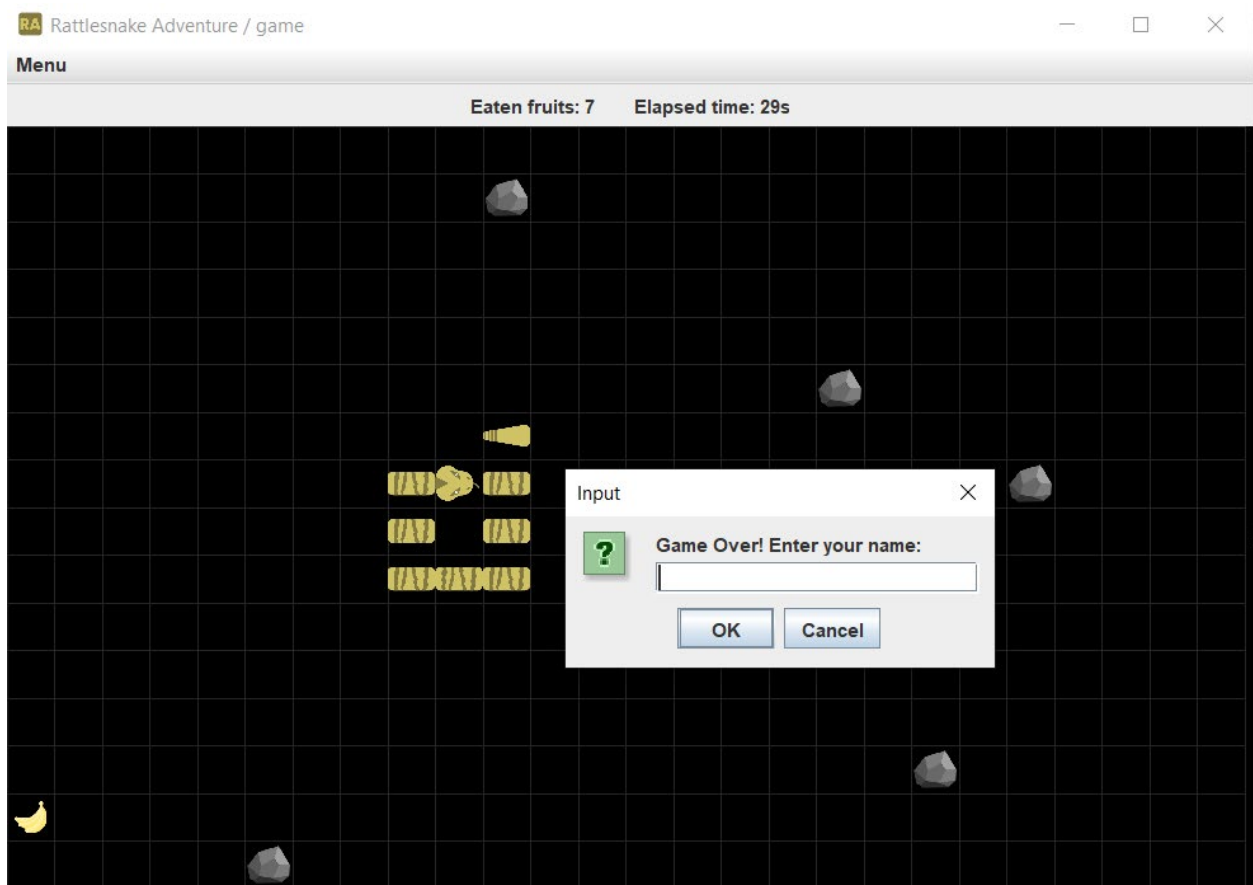
4. Test snake collision

Description:

When snake bumps into itself it is expected that the game will end.

Result:

After bumping into one of the snake parts, the game ended as expected.



5. Test restart button

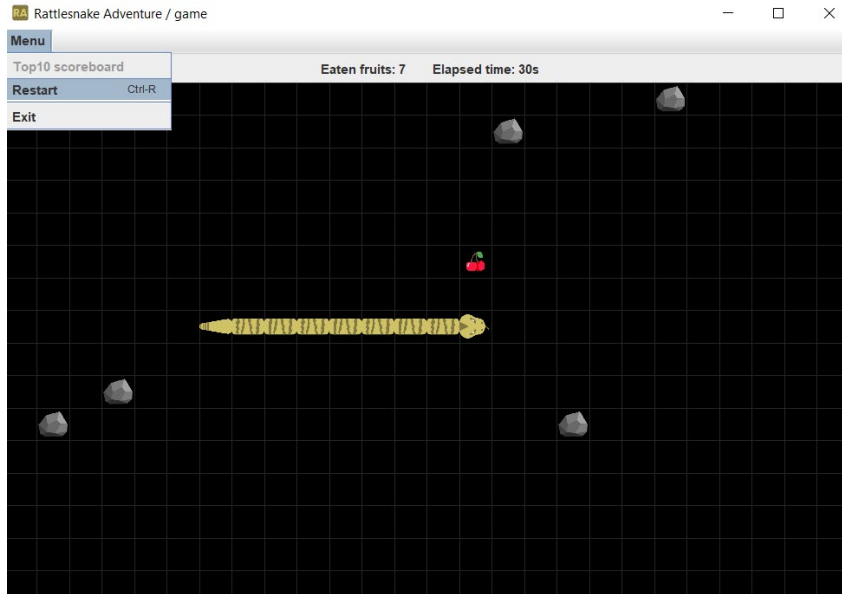
Description:

When restart button in the menu is clicked, it is expected that the game will restart.

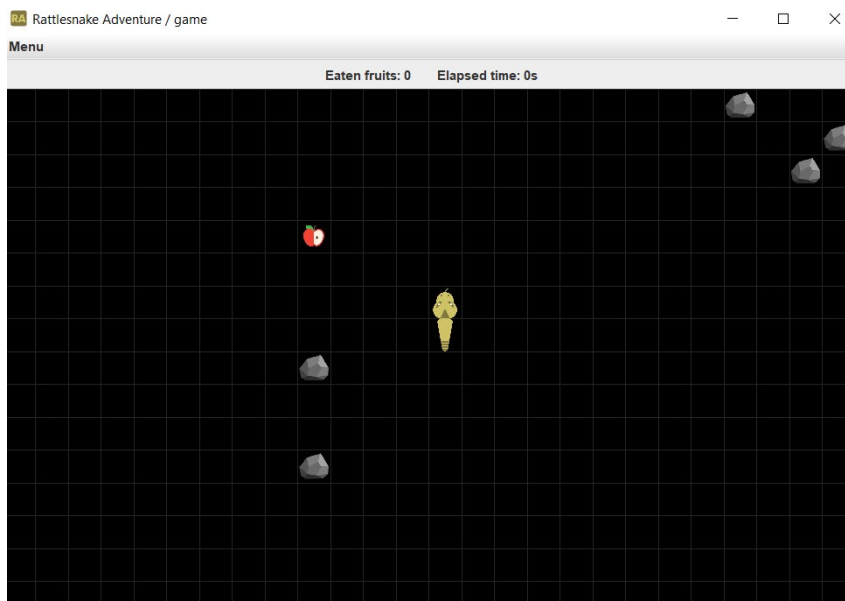
Result:

After clicking the restart button, game restarted as expected.

Before clicking restart:



After clicking restart:



6. Test writing to the database

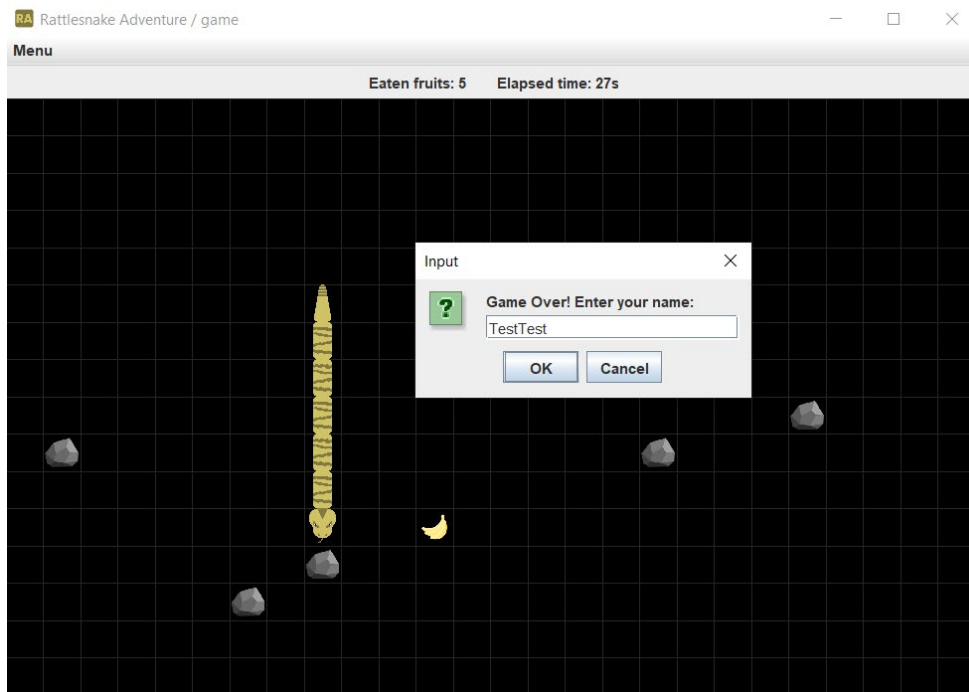
Description:

When game ends and player enters his name, his name and eaten fruits count are added to the database.

Result:

After writing player name and clicking “OK”, player name and eaten fruits count were added to the database.

Game window:



Database:

Result Grid			
Filter Rows:			
	idplayers_score	playername	playerscore
	7	Test3	11
	8	Test4	8
	9	Test5	0
	10	Test6	5
	11	Test5	2
	12	Test7	11
	13	SnakeGam...	29
	14	444	1
	15	2	0
	16	Test4	7
	17	TestTest	5
	NULL	NULL	NULL

7. Test listing top 10 players

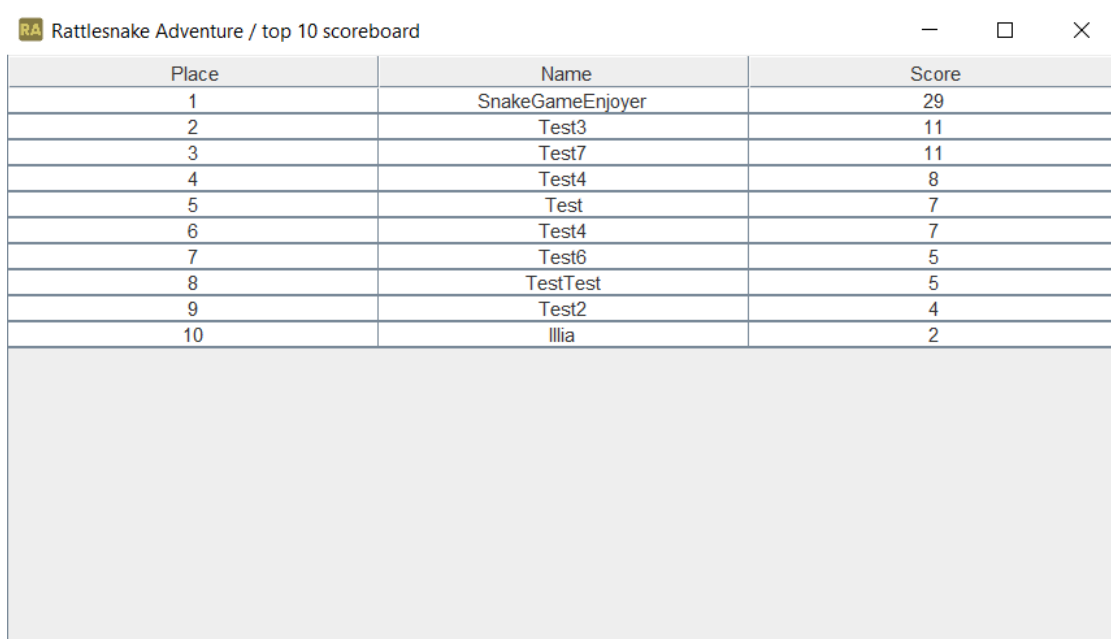
Description:

When menu item “Top10 scoreboard” is clicked, it is expected that new window will open and show top 10 players from the database.

Result:

After clicking menu item “Top10 scoreboard”, new window was opened and top 10 players were listed.

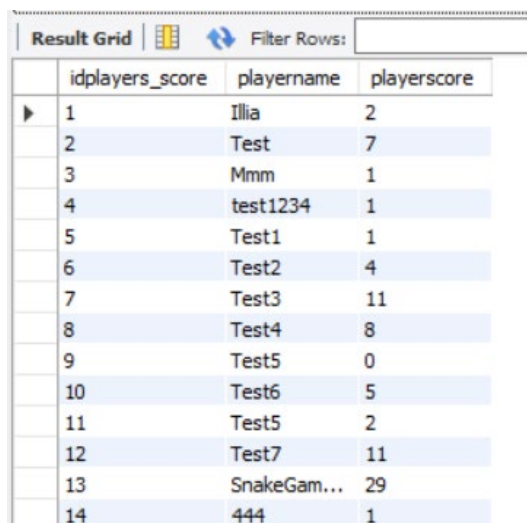
Top 10 window:



The screenshot shows a window titled "Rattlesnake Adventure / top 10 scoreboard". Inside the window is a table with three columns: "Place", "Name", and "Score". The table lists the top 10 players based on their score. Below the table is a large, empty light gray rectangular area.

Place	Name	Score
1	SnakeGameEnjoyer	29
2	Test3	11
3	Test7	11
4	Test4	8
5	Test	7
6	Test4	7
7	Test6	5
8	TestTest	5
9	Test2	4
10	Illia	2

First 14 rows in the database:



The screenshot shows a database result grid with a toolbar at the top containing "Result Grid", a grid icon, a refresh icon, and a "Filter Rows:" input field. The table has four columns: "idplayers_score", "playername", and "playerscore". The first 14 rows are displayed, showing a mix of player names and scores.

	idplayers_score	playername	playerscore
▶	1	Illia	2
	2	Test	7
	3	Mmm	1
	4	test1234	1
	5	Test1	1
	6	Test2	4
	7	Test3	11
	8	Test4	8
	9	Test5	0
	10	Test6	5
	11	Test5	2
	12	Test7	11
	13	SnakeGam...	29
	14	444	1