COIS 2240H Final Project Code Documentation

Group 1: Joey Luyben, David St. Pierre, Colin Crow, William Wilkins

Class: Main

The main class acts as the initializer for the program. All actions, scenes, and options are initialized from within this class. The program itself is also launched from this class. This method defines and calls loaders to communicate with controllers that handle the actual program functionality.

Methods:

1. main

The main method only contains a single command to launch the game.

2. public void start(Stage primaryStage)

The start method is used to initialize the program window within the program. This method is defaulted to when the program is launched from the main method. It initializes the main menu of the program, initializes the database used to store scores, and names the program window.

3. size

The size method calculates the size of the blocks used in the game window

4. private void start(Button button)

This start method is called through the pushing of the “Start” button on the main menu. This method contacts the game loader and initializes the game controller. Using the controller, it sets the default values for the color, speed, block size and window settings. The method then launches the game component of the program.

5. private void options(Button button)

Similarly to the start method, the options method is called through pushing the “Options” button on the main menu. This method contacts the options loader and initializes the options controller. Using the options controller, this method simply displays the different mechanics that can be altered by the player.

6. private void scores(Button button)

The scores method is called through the pushing of the “Scores” button on the main menu. This method calls the scores loaders and initializes the scores controller. It then sets the known scores using the controller and changes the scene to the scores scene.

7. public void handle

The handle method is called when a button is pressed on the main menu of the game. It simply calls another method based on which button was pressed and serves to organize the different button options into a single method.

Class: PlayController

This class servers to define the mechanics of the snake game itself. This controller is used in accordance with the main method and is only utilized if the “Start” button is pressed on the main menu. This class communicates with the GUI and the Main class through the game loader.

Methods:

1. public void launch

This method launches the game itself once notified by the controller. It creates the scene for the game, creates the game itself, initializes the animation within the game, and then starts the game.

2. public void animate

This method defines the animation of the snake itself within the snake game. This method will only activate is the game is considered running by a Boolean flag, otherwise it simply exits the method. This method defines the position of the tail of the snake with respect to the rest of the body. It also defines how the snake and the snake components move when a specific command is interpreted by the keyboard. It will then begin the animation once movement and position are defined. The animate method also defines what happens when the snake reaches the border of the game window, either ending the game or spawning the snake on the opposite side of the screen depending on user preferences. It is also responsible for ending the game if the snake hits itself. Lastly, this method is responsible for ensuring the snake grows once food is eaten, prompting an increase in score as well.

3. private Parent createGame

This method servers to define the game specifications and build the scene. It creates the pane that the scene is displayed in, the snake, the food, the animation time line and finishes by showing all of these elements in the game window. This method is called by the launch method when initializes the scene.

4. private void startGame

This method simply initializes the game variables and begins the animation timeline. It is called from the launch method and the restartGame method.

5. private void restartGame

This method ends the game using the stopGame method and then starts the game again using the startGame method.

6. private void stopGame

This method simply stops the animation timeline and clears the body of the snake.

7. private void moveSnake

This simple method defines the key command for the game window, including which keys change direction, and the restart button.

8. private void endGame

This method begins by ending the current instance of the game using the stopGame method. It then makes use of the scores loader and scores controller to print out the top 5 scores of the game.

9. private void randomizeFood

This method is called by the animate method, and defines a random position for the food to be spawned at during the beginning of the game and every time the food is eaten. It also prevents the food from spawning inside the snake.

Class: Entity

This simple class is just used to create rectangles used as food and the components of the snake. It defines the characteristics of these rectangles based on how it is called when the food or a snake component is defined.

Constructors:

Entity(double xPosition, double yPosition, String colour)

This constructor defines the x and y positions of the entity created, and the color of the entity based on the specifications of the object definition.

Class: OptionsController

This class is used to create and provide functionality to the “Options” screen of the program when the options button is pressed.

Methods:

1. protected void buttonClicked

This method simply calls another method based on which button on the screen was clicked.

2. private void back

This method is called by the buttonClicked method if the “Back” button was pressed on the screen. This method will update any game settings that were altered while in the options screen and then return to the main menu of the game through the use of the start loader.

3. private void reset

This method is called by the buttonClicked method if the “Reset” button was pressed on the screen. This method will reset all options on the options screen to their default values.

4. private void random

This method is called by the buttonClicked method if the “Random” button was pressed on the screen. This method randomly selects values in a specified range and applies these values to the settings of the game.

5. public void fillComboBox

The fillComboBox method fills the combo box of the program.

6. public void setSlider

This small method simply sets the slider value as the user chooses.

7. public void setColourBox

This method sets the color choice of the user.

8. setSizeBox

This method defines the size of the combo box

9. setCheckBox

This method sets a value for the check box of the combo box

Class: ScoresController

This class is responsible for creating and providing functionality to the “Scores” screen of the program when the “Scores” button is pressed.

Methods:

1. protected void buttonClicked

This method simply calls another method based on which button on the screen was clicked.

2. public void setTop5

This method creates labels for the top five score holders of the game and then assigns their respective scores to each label, showing them on screen.

3. public void start

This method is called by the buttonClicked method when the “Start” button is pressed. This method uses the game loader and game controller to initiate the settings of the game and then launch it.

4. private void back

This method is called by the buttonClicked method if the “Back” button was pressed on the screen. This method will update any game settings that were altered while in the options screen and then return to the main menu of the game through the use of the start loader.

5. private void submit

This method receives input from the player in the form of their name and then takes the score of the player’s most recent game and adds it to the database to be stored.

Class: DatabaseUse

This class is responsible for initializing and contacting the database for use with the game's scores. This class both sends and receives data from the database when utilized.

Methods:

1. public static int add

This method establishes a connection to the database. It then adds a record to the data base depending on the properties of the record. The connection to the database is then closed.

2. public static String getRecord

This method establishes a connection to the database, retrieves a record from the database if it is found, and then closes the connection to the database.

3. public static void initialize

This method simply defines a source of connection between the database and the program , it then creates the database itself if it does not already exist, based on the name of the database.