Name: An Nguyen

Course: CSC 143

Quarter: Fall 2018

REPORT

❖ GameWindow class

The class extends Application.

- start (Stage stage): create a Controller object to hold graphic (snake, food, bad food, button, score and game over text). Then create a Scene object (600 x 600) to hold Controller and to control key pressed. Finally set scene to stage and show.
- main method: pass the command line args to the javafx.application.Application.launch

❖ Point class

The Point object helps to create Snake and Food objects.

The Point class has 2 data fields: double x and double y.

Constructor: Point(double x, double y): set data fields

Accessors:

- getX(): return x as double
- getY(): return y as double

Mutators:

- setX(): modify x
- setY(): modify y

❖ Direction class

This object helps to determinate Snake's direction while moving.
The Direction have 4 enumerated constants: `UP (0, -1), DOWN (0, 1), LEFT (-1, 0), RIGHT (1, 0).`

Data fields: double x, double y
Constructor: Direction (double x, double y)
Accessors:

- getX(): return x as double
- getY(): return y as double

❖ Snake class

Data fields:

- ArrayList <Rectangle> snakeBody: hold the body of the Snake as 6x6 square
- Direction direction: direction of the snake when moving
- boolean isMoving: state of the snake (true: can move, false: can't move)
- boolean elongate: if the snake can grow (true: can grow, false: can't grow)
- int score: score of the game

Constructor: `Snake(Direction direction, boolean isMoving)` sets direction and isMoving data fields. Then create the snakeBody with 20 squares (6*6, Black and White stroke) in the middle of the screen. Set elongate = false.

Accessors:

- getDirection() to get direction as Direction (enum)
- getX() to get X point of snake's head as double
- getY() to get Y point of the snake's head as double
- isMoving() to get isMoving state (true or false)
- getScore() to get score as int

Mutators:

- setDirection(Direction direction) to modify direction
- setMoving(boolean isMoving) to modify isMoving.

Methods:

- `checkFoodCollision (Rectangle food, boolean elongate)` checks when snake eats the foods, then return true or false.
- checkSelfCollision() checks if the snake hit itself by comparing the head to every square-body of the snake. If snake hits itself, set game constance in Controller class false.
- draw() return the snakeBody as ArrayList <Rectangle>
- grow() create one more square at the end of the snake and return the tail as Rectangle.
- make() delete old data and create a new snake in the middle of the screen. Set direction = RIGHT, isMoving = false.
- move(): check if isMoving is true, then move the snake in the direction in the data field (move each part of it).
- shrink() check if snake's size is smaller than 2. If true, set isMoving to false, set game constance in Controller class to false. If false, remove the tail of the snake and change its color to black. (tail is at index 0)

❖ Food class

Data field: Rectangle food

Constructor: Food (Snake snake) create a random food and check to make sure that the snake does not contain the food. If contain, create a new random food.

Accessor: getFood(): return the food as Rectangle

Methods:

- checkSnakeCollision (Snake snake): call method checkFoodCollision() from Snake class to check if snake collides the food. If true, randomly relocate the food.
- createFoodLocation (double x, double y): modify food's location
- draw(Color color): return the food with the color in the parameter (RED for bad food and GREEN for food)

❖ Controller class

Data fields:

- Timeline animation: control the animation for the game
- Food badFood
- Line boundary: the bottom boundary of the game
- Snake snake
- Food food
- Text gameBoard: show score of the game
- Button gameButton: to create new game when pressed
- Text gameOver: show "Game Over" and score when snake dies
- boolean paused: if true, paused the game.
- Text rule: print the rule of the game
- int speed: speed of the snake.
- Stage stage: stage of the game

Contants:

- boolean game: if true, game runs. Otherwise, game's over. (can be changed in Snake class)
- String RULE: determine the rule of the game

Constructor: Controller (Stage stage): create food, and bad food, set stage (data field), call createGame() method (to set gameButton, gameBoard, gameOver, rule, boundary), call speedup() method to set animation with initial speed 100 millisecond.

Methods:

- animate(): if isMoving or game is false, stop. Otherwise, move the snake (after check if it hits itself), move food and bad food if snake hits. If snake hits food, increase speed and snake's length. If hits bad food, shrink. Finally, update score.
- checkCollisions(): check if snake hit the wall X(0-588) and Y(0-438). If true, set isMoving and game false.
- control(KeyEvent): can't change direction opposite with the snake's moving-direction
    - Arrow Key(DOWN, UP, RIGHT, LEFT): change direction
    - Space: pause game
- createGame ():
    - boundary: a while line (0, 450, 600, 451),
    - gameBoard: front 25, position (12, 530), Color WHITE
    - gameOver: front 50, position (200, 230), Color RED
    - rule: front 17, position (130, 472), Color WHITE, text RULE
    - gameButton: name "NEW GAME," position (503, 510), action createNewGame()
    - paused = false(game can run)
    - game = true (game can run)
- createNewGame(ActionEvent event): clear the stage, call draw() to add food, badFood and snake, then add all rule, doundary, gameBoard, gameOver, gameButton. Close the odd stage and create a new stage.
- draw(): make new snake, add all food, badFood and snake.
- setScoreBoard(): set score (in gameBoard) and show "Game Over" if game = false
- speedUp (int second): create KeyFrame holding Duration with millisecond from parameter. Create Timeline animation to hold KeyFrame. Set cycle to indefinite and let the animation play.
- update(): call 2 methods checkCollisions () and animate()