PACMAN

JAVA NEWBIES TEAM

this project has costed java newbies team the most important thing in the world it's their time.

10

PREPARED BY:

- MAHMOUD ELBAZ
- MAZEN MOHAMED
- MAHMOUD HABLASS
- ABDALLAH HUSSIEN
- IBRAHIM AYMAN
- MAHMOUD HAMDY
- OSMAN ELKENANY
- MOHAMED WAGEH
- OMAR TARTOUR
- ABDELRAHMAN HOSSIENY

MAY 21 // 2024

PRESENTED FOR DR: HITHAM M.ABO BAKR.

UML design

the next UML illustrates our project structure and the most important methods and attributes.

GamePane

- gameScene: Scene
- ghosts: Ghost[]
- BOARD_WIDTH: int
- BOARD_HEIGHT: int
- pacman: PacMan
- positionChecker: Timeline
- death: boolean
- mainScene: Scene
- sound: GameSounds
- stage: Stage
- CELL_SIZE: int

+ GamePane(mapNumber: int, sound : GameSounds,level: int, pacmanGifnum: int,stage: Stage, scene: Scene)
- createBackground(width: double, height: double,mapNumber: int): ImageView
+ createGhosts(mazeView: MazeView, level: int): Ghost[]
+ movePacman(): void
- startEndScene(isDeath; boolean): void
- handleGameWin(ImageView[], Text[], FadeTransition[], StrokeTransition[], ScaleTransition[]): void
- initializeEndGameTexts(StrokeTransition[], FadeTransition[]): Text[]
- checkLife(): void
- handleGameOver(ImageView[], Text[], FadeTransition[], StrokeTransition[], ScaleTransition[]): void

Maze

- adjMatrix: boolean[][]
- inGates: int[][]
- mazeNum: int
- outGateDireciton: Direction[]
- grid: Cell[][]
- ROWS: int
- COLS: int
- outGates: int[][]
- pellets: int
- Cell :enum

+Maze(width :int, height :int, cellSize: int, maxeNum: int)
+ getRows(): int
+findShortestPath(int,int): List<Integer>
+ cellToIndices(int): int[]
+ getInGates(): int[][]
+ setEmptySpace(int, int): void
- constructAdjMatrix(): void
+ setPellets(int): void
+ isGateIn(int, int): boolean
+ isFinishedMap(): boolean
+ getPellets(): int
+ isPowerPellet(int, int): boolean
+ isWall(int, int): boolean
+ isGateOut(int, int): boolean
+ getCols(): int
- generateInteriorWalls(int): void
- generateMaze(int): void
+ getOutGates(): int[][]
+ getMazeNum(): int
+ getOutGateDirection(): Direction[]
+ indicesToCell(int, int): int
- placePellets(): void
+ isPellet(int, int): boolean

Pane

MazeView

- CELL_SIZE: int
- maze: Maze
- gatesUrls: String[]

+ MazeView(maze: Maze)
- drawMaze(int): void
+ getMaze(): Maze

UML design

PacMan

+ PacMan(int, int, MazeView, int):
- maze: Maze
- mazeView: MazeView
- autoMovement: Timeline
~ sound: GameSounds
- positionTl: Timeline
- score: int

+ moveRight(): void
- setPosition(): void
+ getMazeView(): MazeView
+ getAutoMovement(): Timeline
+ moveUp(): void
- updateScore(): void
- setPositionTl(): void
+ moveDown(): void
+ moveLeft(): void

Pane

Character

- moverTl: Timeline
- reflected: boolean
currentRow: int
currentColumn: int
nextRow: int
nextColumn: int
- counter: int
- dx: double
- dy: double
CELL_SIZE: int
direction: Direction
- gif: ImageView

+ Character()
- setMoverTl(): void
+ reflectVerticallyToLeft(): void
+ rotateToTop(): void
+ setDirection(Direction): void
+ getCurrentRow(): int
- move(): void
setPosition(int, int): void
+ rotateToBottom(): void
+ setGif(ImageView): void
+ reflectVerticallyToRight(): void
+ getGif(): ImageView
+ getCurrentColumn(): int

Ghost

- moves: int
- ghostMovement: Timeline
- steps: int[][]
- pacMan: PacMan
- counter: int
- maze: Maze
- mode: int

+ Ghost(startingI int,startingJ: int ,mazeView; MazeView, level: int)
+ setMode(int): void
- moveDirectedGhost(): void
+ moveRight(): void
- setPath(): void
+ moveDown(): void
- setPosition(): void
- moveGhostRandomly(): void
+ moveLeft(): void
+ moveUp(): void
+ isStuck(int, int): boolean

GameSounds

~ deathSound: MediaPlayer
~ btnSound: MediaPlayer
~ winSound: MediaPlayer
~ eatPellet: MediaPlayer
~ start_sound: MediaPlayer

+ GameSounds():

UML design

Main

- level: int
- stage: Stage
- introFinished: boolean
- CELL_SIZE: int
- BOARD_HEIGHT: int
- introPlayer: MediaPlayer
- levelScene: Scene
- charactersScene: Scene
- mapsScene: Scene
- infoScene: Scene
- pacmanGifNum: int
- mainScene: Scene
- mainMenueBtns: Button[]
- mapNumber: int
- sound: GameSounds
- BOARD_WIDTH: int

+ start(Stage): void
- addButtonEffect(button: Button, before: Color, after: Color): void
- setMapScene(): void
- setMainScene(): void
- setCharactersScene(): void
- setInfoScene(): void
- charactersGifsPane(): StackPane
- charactersBtnsPane(charactersPane :StackPane): VBox
- setLevelScene(): void
- mainMenuBtnsPane(): Pane
- setImgDimensions(ImageView, width: double,height: double): void

→ **Application**

Classes’ usages

this part will shows classes’ explanations , and clarifies the most important attributes and methods

Maze class

this class represents the backend for our maps and bases for characters’ movement ,The maze serves as the environment in which Pacman and the ghosts interact, determining their movement, collision detection, and overall gameplay dynamics.

member	clarification
- grid: Cell[][]	the maze is represented in 2D array, each element in the array represents a cell which might be a wall, empty space, pellet , power pellet, gate in or gate out
- adjMatrix: boolean[][]	matrix is used to represent the grid as a graph DS
+Maze(width:double,heiht :double, cellSize: double, maxeNum:int)	Constructor for the Maze class, initializes the maze with the rows and columns based on the specified dimensions and maze number.
-generateInteriorWalls (mazeNum: int): void	Generates the interior walls of the maze based on the given maze number.
+ indicesToCell(row: int, column: int): int	Converts the row and column indices to a cell identifier.(2D to 1D)
+ setEmptySpace(row: int, column: int): void	Sets the cell at the given coordinates as an empty space (removes the pellets).
- constructAdjMatrix(): void	Constructs the adjacency matrix based on the maze's grid.
+ findShortestPath(start: int, end: int): List<Integer>	Finds the shortest path from the start cell(ghost’s cell) to the end cell(pacman’s cell) using a common graph algorithm called breadth-first search BFS

Classes’ usages

MazeView class

it’s a class represents the maze and view it as pane

member	clarification
-maze: Maze	the maze which will be represented
+ MazeView(maze Maze)	Initializes the MazeView object with the given Maze and set the appropriate background based on the maze number
drawMaze(int level)	Draws the maze on the screen based on the provided level.
+ getMaze()	Returns the current Maze object associated with the MazeView.

Game Sound class

it’s a class represents the game sounds

member	clarification
deathSound: MediaPlayer	Handles the sound played when Pac-Man dies.
btnSound: MediaPlayer	Handles the sound played when a button is pressed.
winSound: MediaPlayer	Handles the sound played when the player wins a level.
eatPellet: MediaPlayer	Handles the sound played when Pac-Man eats a pellet.
start_sound: MediaPlayer	Handles the sound played at the start of the game
GameSounds()	Initializes the **GameSounds** object and sets up the media players for various game sounds.

Classes’ usages

Character class

this class provides various functionalities for managing the movement, rotation, and position of the character. They allow setting the character's direction, position, and GIF image, as well as performing rotations and reflections.

member	clarification
# currentColumn: int # currentRow: int	Fields for detecting the current position for the character
# currentColumn: int # currentRow: int	Fields for detecting the next position for the character will be in, after the completion of the movement
- moverTl: Timeline	invoke move(): void every 10.1ms
+ Character()	Constructor that initialize the animations used in characters’ movement by invoking { setPositionTl(), setMoverTl()}
+ rotateToTop(): void + rotateToBottom(): void	Rotates the character to face upward and downward.
+ reflectVerticallyToRight(): void + reflectVerticallyToLeft(): void	Rotates the character to face right and left.
# setPosition(row: int, column: int): void	Sets the position of the character to the specified row and column by initializing (dx, dy) and play the moverTl animation
- move(): void	Changes Character Gif Position 2 Pixels For The Cycle using dy and dx to get the right direction
- setPosition(): void	tricky function used to set delayed position by playing setPositionTl animation

Classes’ usages

pacman class

This class provides functionalities specific to the Pacman character in the game. The constructor initializes the Pacman character with its starting position, associated MazeView, and GIF number.

member	clarification
- mazeView: MazeView	used to stick the pacman with the right maze
- autoMovement: Timeline	animation used to move the pacman automatically
- positionTl: Timeline	Changes Character Gif Coordinates Through a Certain Amount of Time (190ms)
+PacMan(startingI int, startingJ int, mazeView MazeView, gifNumber int)	Initializes the PacMan object with starting positions, the maze view, and a gif number.
+moveLeft()	Moves the Pac-Man character to the left in the maze.
+moveRight()	Moves the Pac-Man character to the right in the maze.
+moveUp()	Moves the Pac-Man character upwards in the maze.
-updateScore()	Updates the score of the game based on the eaten pellet and also update the position if the current cell is a gate
+moveDown()	Moves the Pac-Man character downwards in the maze
+getAutoMovement()	Returns the Timeline object that handles the automatic movement of Pac-Man.
- setPosition(): void	tricky function used to set delayed position by playing setPositionTl animation

Classes’ usages

ghost class

This class provides functionalities specific to the ghost character in the game, control the movement based on a specific mode and level

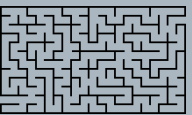
member	clarification
- ghostMovement: Timeline	used to move the repeat ghost movement every specific period based on the level
- steps: int[][]	steps for the directed ghost to follow
+ Ghost(startingI int, startingJ int, mazeView MazeView, level int)	Initializes the Ghost object with starting positions, the maze view, and the game level
+ setMode(int mode)	Sets the mode (flag for random movement) of the ghost
- moveDirectedGhost(): void	Moves the ghost in a directed manner towards Pac-Man
- setPath(): void	Sets the path for the ghost to follow .
- setPosition(): void	Sets the position of the ghost in the maze directly.
- moveGhostRandomly(): void	Moves the ghost in a random direction.
+ isStuck(int, int): boolean	Checks if the ghost is stuck at the given position (x, y).

Classes’ usages

GamePane class

This class is responsible for game creation as a pane and put it in a game scene

member	clarification
- gameScene: Scene	the scene in which the game pane will be added
+ GamePane(mapNumber: int, sound : GameSounds,level: int, pacmanGifnum: int,stage: Stage, scene: Scene)	Initializes the GamePane object with the map number, sound, level, Pac-Man GIF number, stage, and scene.
- createBackground(width: double, height: double, mapNumber: int):	Creates and returns the background image view based on the provided dimensions and map number
+ createGhosts(mazeView MazeView, level int)	Creates and returns an array of Ghost objects based on level and stick each ghost to the maze view.
+ movePacman()	Handles the key event for movement of Pac-Man within the game.
- startEndScene(isDeath; boolean): void	Starts the end scene, either for game over (death) or game win.
- handleGameWin(ImageView[], Text[], FadeTransition[], StrokeTransition[], ScaleTransition[]): void	Handles the logic for when the player wins the game, including animations.
- initializeEndGameTexts(StrokeTransition[], FadeTransition[]): Text[]	Initializes and returns the texts to be displayed at the end of the game.
+ checkLife()	Checks the life status of Pac-Man using positionChecker and updates the game state accordingly.
-handleGameOver(ImageView[] ghosts, Text[] texts, FadeTransition[] fades, StrokeTransition[] strokes, ScaleTransition[] scales)	Handles the logic for when the game is over, including animations

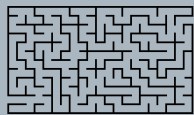


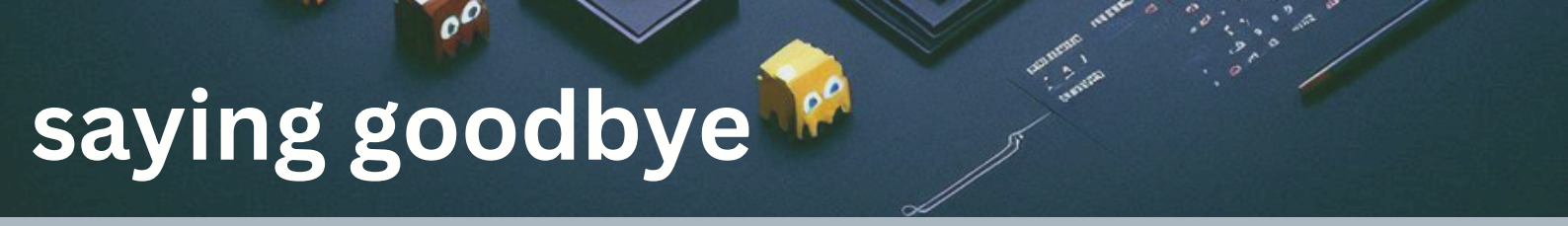
Classes’ usages

Main class

it’s the game master class which invokes methods for setting all scenes and creates all the necessary objects

member	clarification
+ start(Stage): void	Initializes the main application and sets the primary stage.
- addButtonEffect(button: Button, before: Color, after: Color): void	Adds a visual effect to a button that changes its color on interaction.
- setMapScene(): void	Sets the scene for map selection in the game.
- setMainScene()	Sets the main scene of the game, including the main menu.
- setCharactersScene(): void	Sets the scene for character selection.
- setInfoScene(): void	set scene for information about the project creators
- charactersGifsPane(): StackPane	Creates and returns a StackPane containing character GIFs for selection.
- charactersBtnsPane(charactersPane :StackPane) : VBox	Creates and returns a VBox containing buttons for character selection, based on the provided StackPane.
- setLevelScene(): void	Sets the scene for level selection.
- mainMenuBtnsPane(): Pane()	Creates and returns a Pane containing the main menu buttons.





saying goodbye

At the end of those long nights of searching, brainstorming and coding, now we get our target by completing this project with the desired manner.

This project was more than just a technical achievement; it was a testament to the power of teamwork and friendship. Working together, we not only honed our Java skills but also forged strong bonds that will last far beyond this project. The experience has shown us that with collaboration, and a bit of fun along the way, we can tackle any challenge that comes our way.

We learned not just from the problems we encountered, but from each other, growing both as programmers and as a team.

gameplay_video

explore the code