# Brick Breaker Game Documentation

# 

# M . Deepak Das July- 2024

## Overview

This project is a simple Brick Breaker game developed using Java. The game includes features such as multiple levels, sound effects, and a graphical user interface for user interaction. The primary objective of the game is to break all the bricks using a ball and a paddle.

## Project Structure

The project consists of the following main components:

* **Ball**: Represents the ball in the game.
* **Brick**: Represents the bricks that the player needs to break.
* **Paddle**: Represents the paddle controlled by the player.
* **Gameplay**: Manages the main game logic and user interaction.
* **GamePanel**: Sets up the game window.
* **LevelManager**: Manages the levels in the game.
* **MapGenerator**: Generates the map of bricks for each level.
* **SoundManager**: Handles the sound effects in the game.
* **BrickBreakerGame**: The main class to start the game.

## Class Descriptions

### Ball

The Ball class represents the ball in the game.

#### Fields

* int x, y: The x and y coordinates of the ball.
* int diameter: The diameter of the ball.
* int xDir, yDir: The direction of the ball's movement.
* Color color: The color of the ball.

#### Methods

* **Ball(int x, int y, int diameter, int xDir, int yDir, Color color)**: Constructor to initialize the ball.
* **void move()**: Moves the ball according to its direction.
* **void draw(Graphics g)**: Draws the ball on the screen.
* **Rectangle getBounds()**: Returns the bounding rectangle of the ball.
* **int getY()**: Returns the y-coordinate of the ball.

### Brick

The Brick class represents a brick in the game.

#### Fields

* int x, y: The x and y coordinates of the brick.
* int width, height: The width and height of the brick.
* Color color: The color of the brick.
* boolean isBroken: Indicates if the brick is broken.

#### Methods

* **Brick(int x, int y, int width, int height, Color color)**: Constructor to initialize the brick.
* **void draw(Graphics g)**: Draws the brick on the screen.
* **Rectangle getBounds()**: Returns the bounding rectangle of the brick.

### Paddle

The Paddle class represents the paddle controlled by the player.

#### Fields

* int x, y: The x and y coordinates of the paddle.
* int width, height: The width and height of the paddle.
* int xDir: The direction of the paddle's movement.
* Color color: The color of the paddle.

#### Methods

* **Paddle(int x, int y, int width, int height, Color color)**: Constructor to initialize the paddle.
* **void move()**: Moves the paddle according to its direction.
* **void setXDir(int xDir)**: Sets the direction of the paddle's movement.
* **void draw(Graphics g)**: Draws the paddle on the screen.
* **Rectangle getBounds()**: Returns the bounding rectangle of the paddle.

### Gameplay

The Gameplay class manages the main game logic and user interaction.

#### Fields

* boolean play, gameStarted, showNextLevelMessage: Flags to manage game states.
* int score: The player's score.
* int totalBricks: The total number of bricks in the current level.
* LevelManager levelManager: Manages the game levels.
* Timer timer: The game timer.
* int delay: The delay for the timer.
* int playerX: The x-coordinate of the player's paddle.
* int ballposX, ballposY: The coordinates of the ball.
* int ballXdir, ballYdir: The direction of the ball's movement.
* MapGenerator map: Generates the map of bricks for the current level.
* SoundManager soundManager: Manages the sound effects.

#### Methods

* **Gameplay()**: Constructor to initialize the gameplay.
* **void paint(Graphics g)**: Draws the game components on the screen.
* **void actionPerformed(ActionEvent e)**: Handles the game logic for each timer tick.
* **void keyPressed(KeyEvent e)**: Handles the key press events.
* **void keyReleased(KeyEvent e), void keyTyped(KeyEvent e)**: Handles key release and key typed events.
* **void mouseClicked(MouseEvent e), void mousePressed(MouseEvent e), void mouseReleased(MouseEvent e), void mouseEntered(MouseEvent e), void mouseExited(MouseEvent e)**: Handles mouse events.
* **void moveRight()**: Moves the paddle to the right.
* **void moveLeft()**: Moves the paddle to the left.

### GamePanel

The GamePanel class sets up the game window.

#### Methods

* **GamePanel()**: Constructor to initialize the game panel and set up the game window.

### LevelManager

The LevelManager class manages the levels in the game.

#### Fields

* int currentLevel: The current level of the game.
* int totalLevels: The total number of levels in the game.

#### Methods

* **LevelManager(int totalLevels)**: Constructor to initialize the level manager with the total number of levels.
* **int getCurrentLevel()**: Returns the current level.
* **void nextLevel()**: Advances to the next level.
* **Brick[][] getBricks()**: Returns the bricks for the current level.
* **int getBallSpeed()**: Returns the speed of the ball for the current level.

### MapGenerator

The MapGenerator class generates the map of bricks for each level.

#### Fields

* int[][] map: The map of bricks.
* int brickWidth, brickHeight: The width and height of each brick.

#### Methods

* **MapGenerator(int row, int col)**: Constructor to initialize the map generator with the specified number of rows and columns.
* **void draw(Graphics2D g)**: Draws the map of bricks on the screen.
* **void setBrickValue(int value, int row, int col)**: Sets the value of a brick in the map.

### SoundManager

The SoundManager class handles the sound effects in the game.

#### Fields

* Clip clip: The audio clip for the sound effect.

#### Methods

* **void playSound(String filePath)**: Plays the sound effect from the specified file path.
* **void stopSound()**: Stops the currently playing sound effect.

### BrickBreakerGame

The BrickBreakerGame class is the main class to start the game.

#### Methods

* **public static void main(String[] args)**: The main method to start the game.

## How to Run

1. Ensure you have Java installed on your system.
2. Compile all the Java files in the project.
3. Run the BrickBreakerGame class to start the game.

## Future Improvements

* Adding more levels with different difficulty settings.
* Implementing power-ups for the paddle and ball.
* Adding a scoring system with leaderboards.
* Improving the graphical interface with better animations and effects.
* Making the game mobile-friendly by developing an Android version.