



Xin chào! Mình tên là Bảo Phúc.

Mình là sinh viên thực hiện dự án
mini cuối kì môn Lập Trình Java:
Game Phá Gạch - Brick Breaker



Giới thiệu về Dự án cuối kì môn Java: Game Phá Gạch – Brick Breaker

Nguyễn Hoàng Bảo Phúc
63635100

Ngày 08/01/2024 | 9:30
Nhà Đa Năng 205



Chủ đề giới thiệu

Những gì chúng ta sẽ biết về dự án hôm nay

1. Mô tả sơ lược Dự án

2. Các công nghệ/khiến thức được sử dụng

3. Các chức năng thông qua các Class của Game

4. Kế hoạch phát triển

Mô tả sơ lược dự án



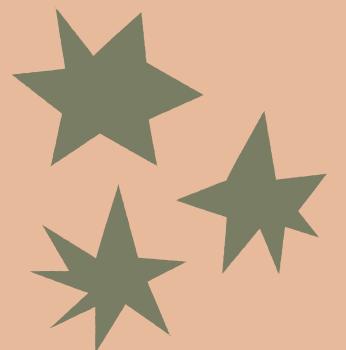
Các ý của phần này

- Lời giới thiệu
- Các hình ảnh cho dự án

1. Lời giới thiệu



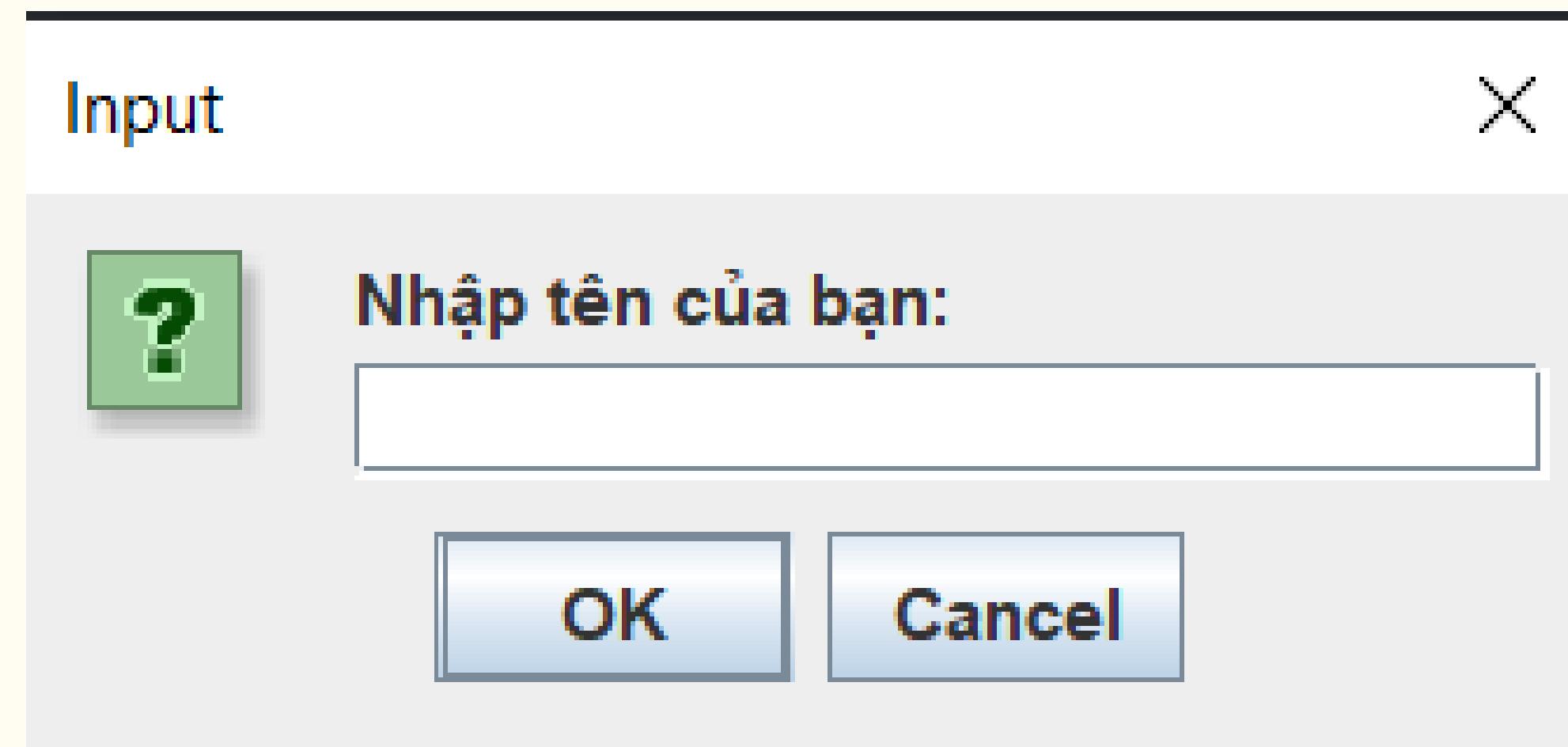
Java





Các hình ảnh của dự án

2.1. Ô lấy tên người chơi



- Xuất hiện vào lúc mới bắt đầu game, và sau khi người chơi thất bại
- Dùng để thu thập tên của người chơi, phục vụ cho bảng xếp hạng

2.2. Màn hình chào mừng.



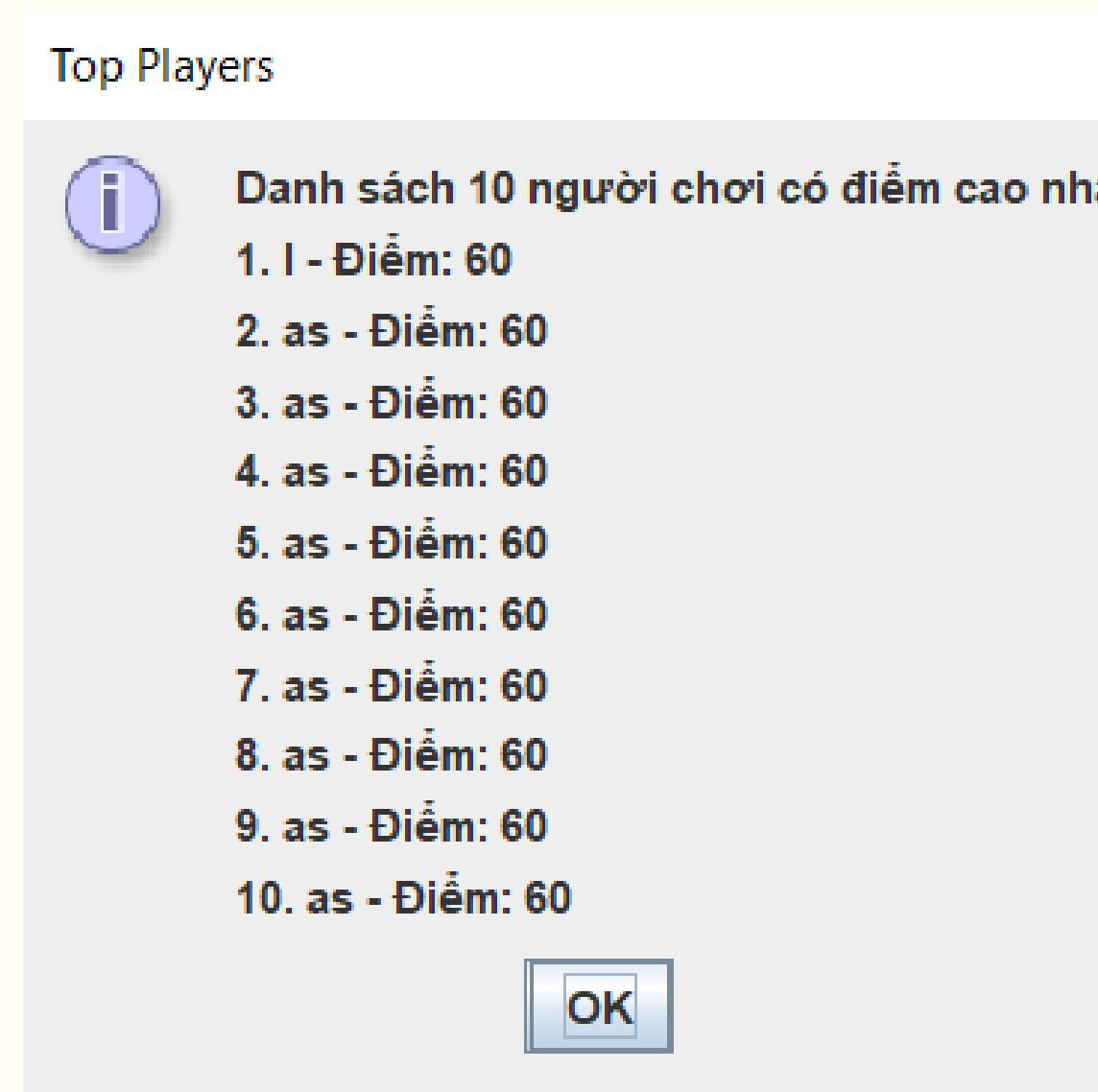
- Xuất hiện sau khi mình nhập tên người chơi
- Giao diện này để ta thấy, chuẩn bị tinh thần để nhấn Enter và vào game

2.3. Màn hình qua một level.



- Sau khi chiến thắng một màn, màn hình sẽ hiển thị lên
- Giao diện thông báo cho ta biết level kế tiếp sắp phải đối mặt

- Sau khi thua thì màn hình này sẽ hiển thị
- Cung cấp cho ta những cái tên, điểm số của 10 người chơi có thành tích xuất sắc nhất
- Một phần tạo động lực cho người chơi phấn đấu để được ghi danh lên bảng này



Các kiến thức/công nghệ được sử dụng



1. Java Swing.



- Trò chơi sử dụng thư viện đồ họa Swing của Java để xây dựng giao diện người dùng và vẽ đồ họa.

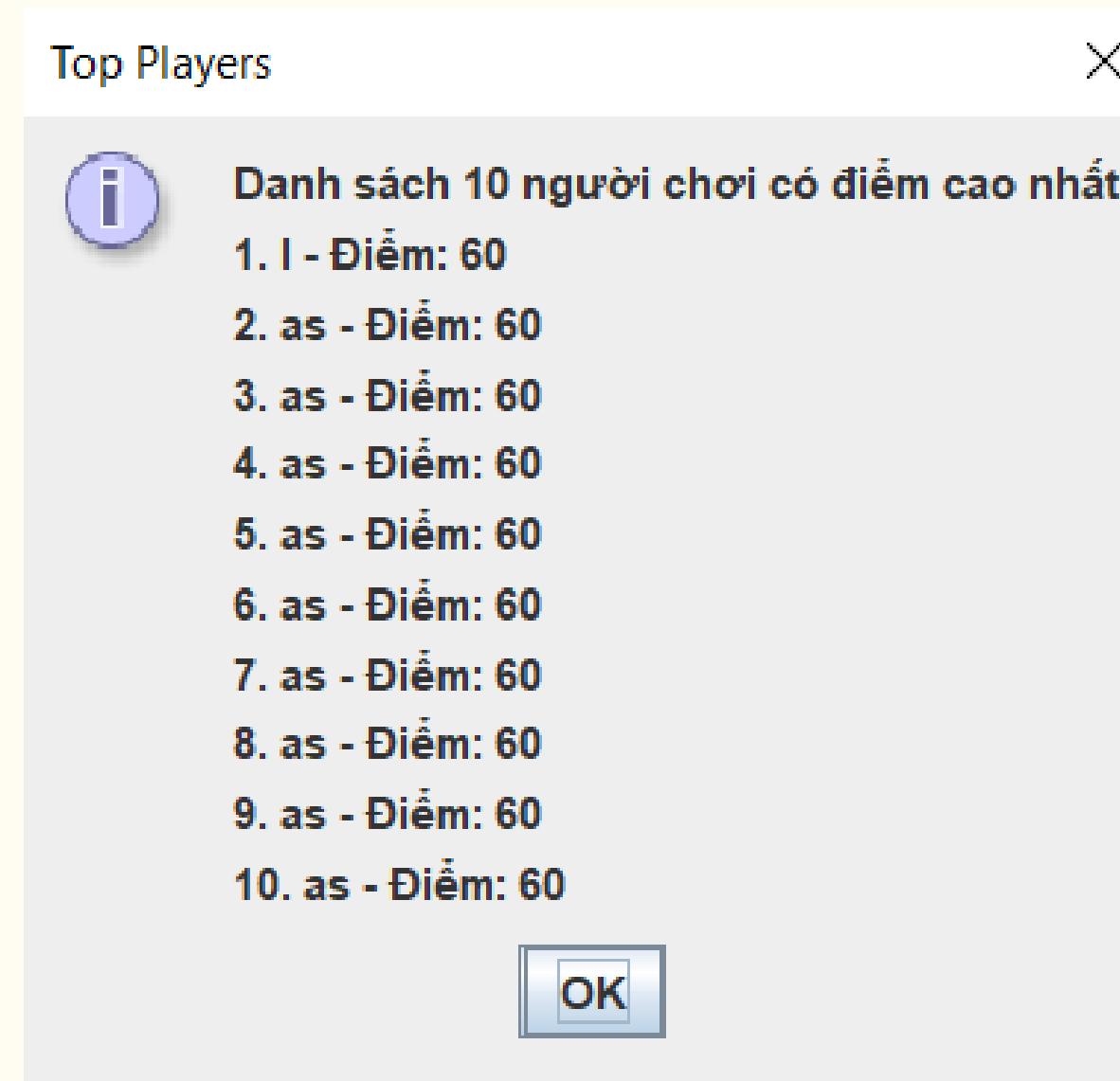
2. Event Handling (Xử lý sự kiện).



```
1 public class Gameplay extends JPanel implements KeyListener, ActionListener
```

- Sử dụng giao thức KeyListener để xử lý sự kiện phím.
- Sử dụng giao thức ActionListener để xử lý các sự kiện định kỳ, chẳng hạn như cập nhật vị trí của quả bóng và vẽ lại màn hình.

3.File I/O (Input/Output).



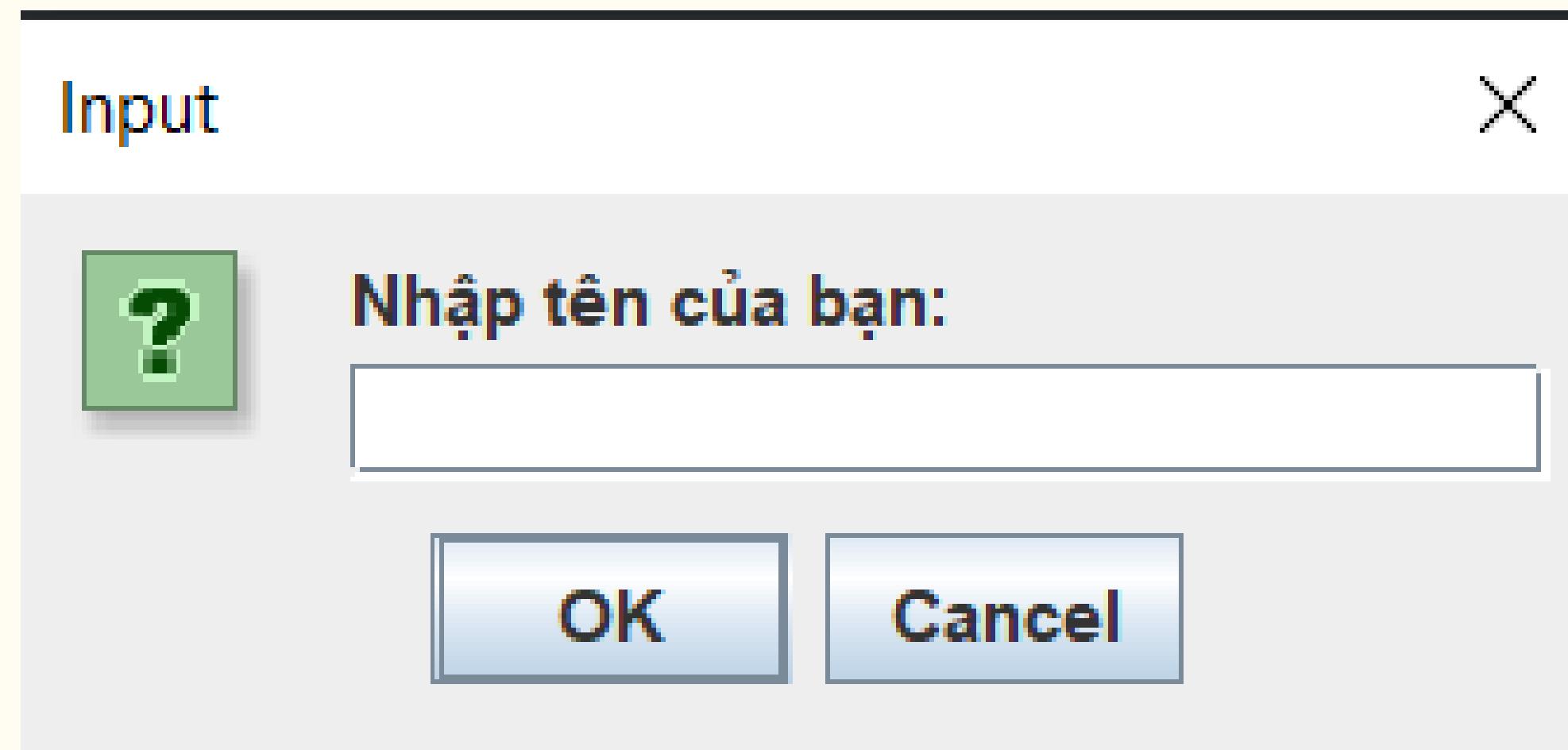
- Sử dụng BufferedWriter để ghi dữ liệu người chơi (tên và điểm số) vào một tệp tin.
- Sử dụng BufferedReader để đọc dữ liệu người chơi từ tệp tin.

4. Hiệu ứng đồ họa đơn giản.



- Sử dụng đối tượng **Graphics2D** để vẽ các thành phần đồ họa như viên gạch, thanh điều khiển, và quả bóng.
- Sử dụng hiệu ứng căn chỉnh màu sắc, vị trí để tạo ra giao diện đơn giản và hấp dẫn.

5. JOptionPane.



- Sử dụng JOptionPane để hiển thị hộp thoại đầu vào để nhập tên người chơi

6. Cấu trúc Dữ liệu và Lập trình Hướng đối tượng.

```
● ● ●  
1 public class Gameplay extends JPanel implements KeyListener, ActionListener  
2 public class Player  
3 public class MapGenerator
```

- Sử dụng mảng để lưu trữ thông tin về viên gạch và quản lý bản đồ gạch.
- Sử dụng lập trình hướng đối tượng với các class như MapGenerator, Gameplay, và Player.

Chức năng thông qua từng Class.





01 Player

02 MapGenerator

03 Gamelay

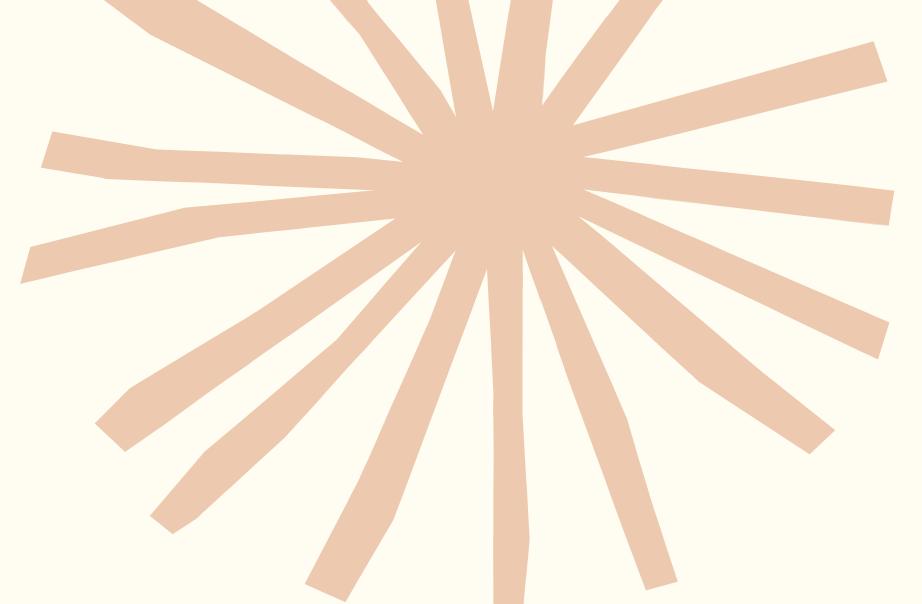
04 Main



1. Player.java

Chức năng của Class này nhằm tạo ra một đối tượng Player, sử dụng cho việc khởi tạo một mảng chứa các thông tin về tên, điểm số và các việc khác.

```
● ● ●  
1 public class Player{  
2     public String playerName;  
3     public int playerScore;  
4  
5     public Player(String playerName, int playerScore) {  
6         this.playerName = playerName;  
7         this.playerScore = playerScore;  
8     }  
9 }
```



2. MapGenerator.java



```
1 import java.awt.BasicStroke;
2 import java.awt.Color;
3 import java.awt.Graphics2D;
4
5 public class MapGenerator
6 {
7     public int[][] map;
8     public int brickWidth;
9     public int brickHeight;
10
11
12     public MapGenerator (int row, int col)
13     {
14         map = new int[row][col];
15         for(int i = 0; i<map.length; i++)
16         {
17             for(int j =0; j<map[0].length; j++)
18             {
19                 map[i][j] = 1;
20             }
21         }
22
23         brickWidth = 540/col;
24         brickHeight = 150/row;
25     }
26
27     public void draw(Graphics2D g)
28     {
29         for(int i = 0; i<map.length; i++)
30         {
31             for(int j =0; j<map[0].length; j++)
32             {
33                 if(map[i][j] > 0)
34                 {
35                     g.setColor(Color.white);
36                     g.fillRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
37
38                     // this is just to show separate brick, game can still run without it
39                     g.setStroke(new BasicStroke(3));
40                     g.setColor(Color.black);
41                     g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
42                 }
43             }
44         }
45     }
46
47     public void setBrickValue(int value, int row, int col)
48     {
49         map[row][col] = value;
50     }
51 }
```

Đây là toàn bộ nội dung của class này, bao gồm:

- các thuộc tính
- Constructor
- draw
- setBrickValue



```
1 public int[][] map;  
2 public int brickWidth;  
3 public int brickHeight;
```

Class này gồm 3 thuộc tính:

- **int[][] map:** dùng để tạo ra 1 mảng 2 chiều, chứa các viên gạch
- **int brickWidth:** độ rộng của 1 viên gạch
- **int BrickHeight:** chiều cao của 1 viên gạch



```
1 public MapGenerator (int row, int col)
2 {
3     map = new int[row][col];
4     for(int i = 0; i<map.length; i++)
5     {
6         for(int j =0; j<map[0].length; j++)
7         {
8             map[i][j] = 1;
9         }
10    }
11
12    brickWidth = 540/col;
13    brickHeight = 150/row;
14 }
```

- Phương thức khởi tạo có 2 tham số truyền vào là số hàng và số cột của map
- $\text{map}[i][j] = 1$ trong 2 vòng for là giá trị của viên gạch, biểu thị viên gạch tồn tại hay không
- độ cao và độ rộng của viên gạch sẽ tùy thuộc vào số hàng số cột của map theo công thức ở bên trái



```
1 public void draw(Graphics2D g)
2 {
3     for(int i = 0; i<map.length; i++)
4     {
5         for(int j =0; j<map[0].length; j++)
6         {
7             if(map[i][j] > 0)
8             {
9                 g.setColor(Color.white);
10                g.fillRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
11
12                // this is just to show separate brick, game can still run without it
13                g.setStroke(new BasicStroke(3));
14                g.setColor(Color.black);
15                g.drawRect(j * brickWidth + 80, i * brickHeight + 50, brickWidth, brickHeight);
16            }
17        }
18    }
19 }
```

- Đây là phương thức vẽ map, duyệt qua toàn bộ map, nếu viên gạch tồn tại thì cho nó màu trắng, ...



```
1 public void setBrickValue(int value, int row, int col)
2 {
3     map[row][col] = value;
4 }
```

- Phương thức này được tạo ra để cập nhật lại giá trị cho những viên gạch, thông qua việc bóng va chạm với gạch thì phương thức này sẽ được gọi để thay đổi giá trị của nó.

3. Gameplay.java

Class này bao gồm những phần sau:



```
1 private boolean play = false;  
2 private int score = 0;  
3 private int scoreNext;  
4 private boolean isPause = false;  
5 private boolean hello = true;
```



```
1 public Gameplay() {  
2     map = new MapGenerator(row, col);  
3     addKeyListener(this);  
4     setFocusable(true);  
5     setFocusTraversalKeysEnabled(false);  
6     timer = new Timer(delay, this);  
7     getPlayerName();  
8     timer.start();  
9 }
```



```
1 public void paint(Graphics g) {
```



Và còn những thành phần
khác nữa..

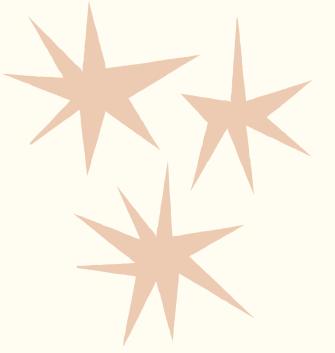


3.1. Các biến được sử dụng



```
1 private static final String PLAYER_DATA_FILE = "data_players.txt";
2 private boolean play = false;
3 private int score = 0;
4 private int scoreNext;
5 private boolean isPause = false;
6 private boolean hello = true;
7
8 public int col = 5, row = 3;
9 public int colNext = 5, rowNext = 3;
10
11 private String playerName;
12
13 private ArrayList<Player> playersList;
14
15 private int totalBricks = row * col;
16 public int level = 1;
17
18 final Timer timer;
19 final int delay = 8;
20
21 private int playerX = 310;
22
23 private int ballposX = 120;
24 private int ballposY = 350;
25 private int ballXdir = -1;
26 private int ballYdir = -2;
27
28 private MapGenerator map;
```

**Đây là toàn bộ các biến được
sử dụng trong class này**



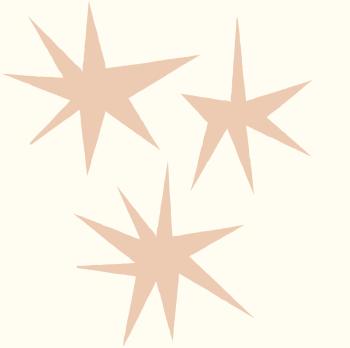
3.2. Constructor

- a. Tạo một map với số cột số dòng tương ứng với mỗi level khác nhau
- b. Tạo ra một đồng hồ bấm giờ phục vụ các chức năng khác trong class
- c. getPlayerName để hiện ra cửa sổ, người chơi nhập tên vào.



```
1 public Gameplay() {  
2     map = new MapGenerator(row, col);  
3     addKeyListener(this);  
4     setFocusable(true);  
5     setFocusTraversalKeysEnabled(false);  
6     timer = new Timer(delay, this);  
7     getPlayerName();  
8     timer.start();  
9 }
```





3.3. paint

a. Vẽ các thành phần chính

b. Vẽ các màn hình theo các trường hợp

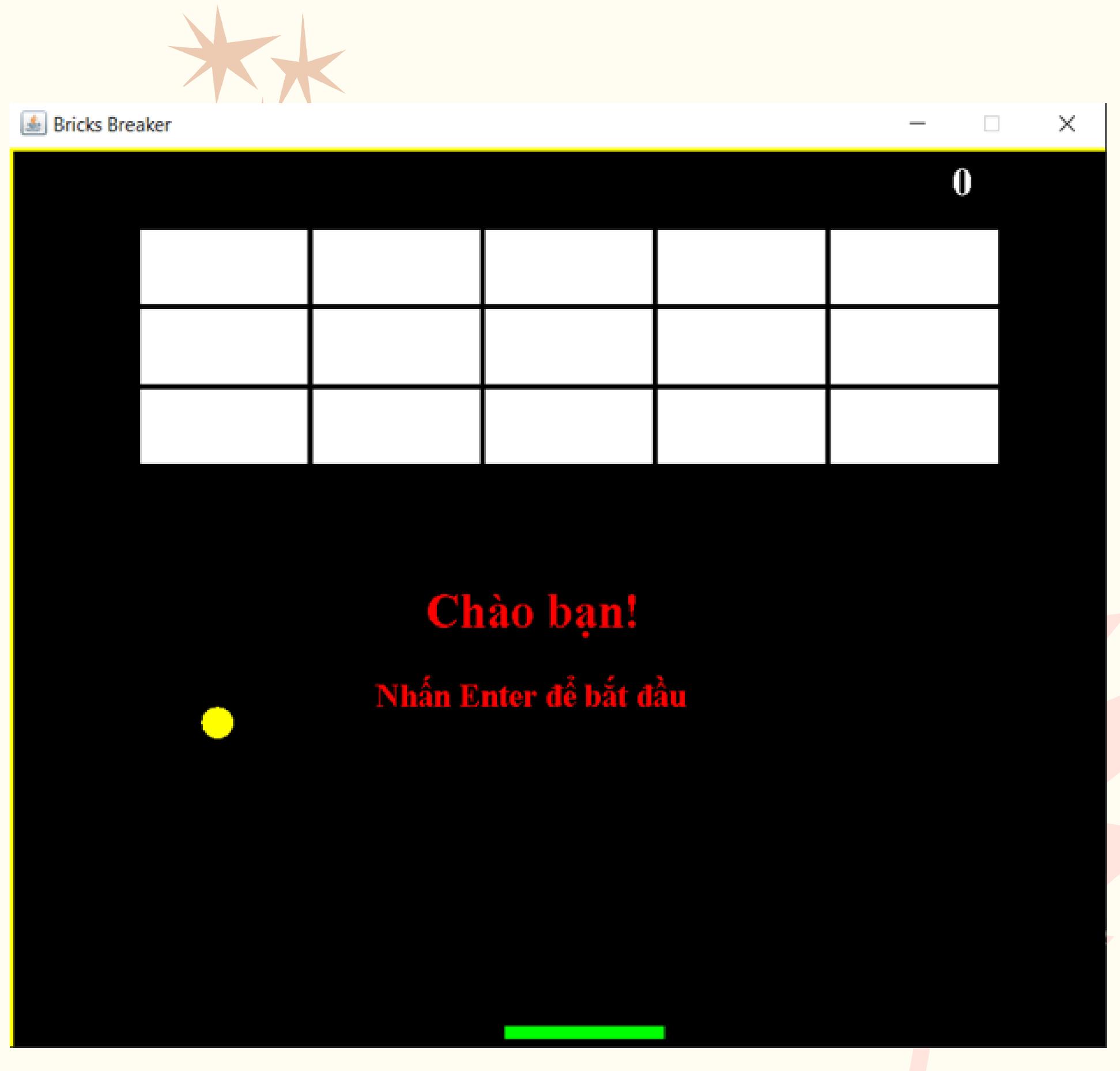


```
1 public void paint(Graphics g) {
```



3.3a. Vẽ các thành phần chính

```
● ● ●  
1 // background  
2     g.setColor(Color.black);  
3     g.fillRect(1, 1, 692, 592);  
4  
5 // drawing map  
6 map.draw((Graphics2D) g);  
7  
8 // borders  
9 g.setColor(Color.yellow);  
10 g.fillRect(0, 0, 3, 592);  
11 g.fillRect(0, 0, 692, 3);  
12 g.fillRect(691, 0, 3, 592);  
13  
14 // the scores  
15 g.setColor(Color.white);  
16 g.setFont(new Font("serif", Font.BOLD, 25));  
17 g.drawString(" " + score, 590, 30);  
18  
19 // the paddle  
20 g.setColor(Color.green);  
21 g.fillRect(playerX, 550, 100, 8);  
22  
23 // the ball  
24 g.setColor(Color.yellow);  
25 g.fillOval(ballposX, ballposY, 20, 20);
```

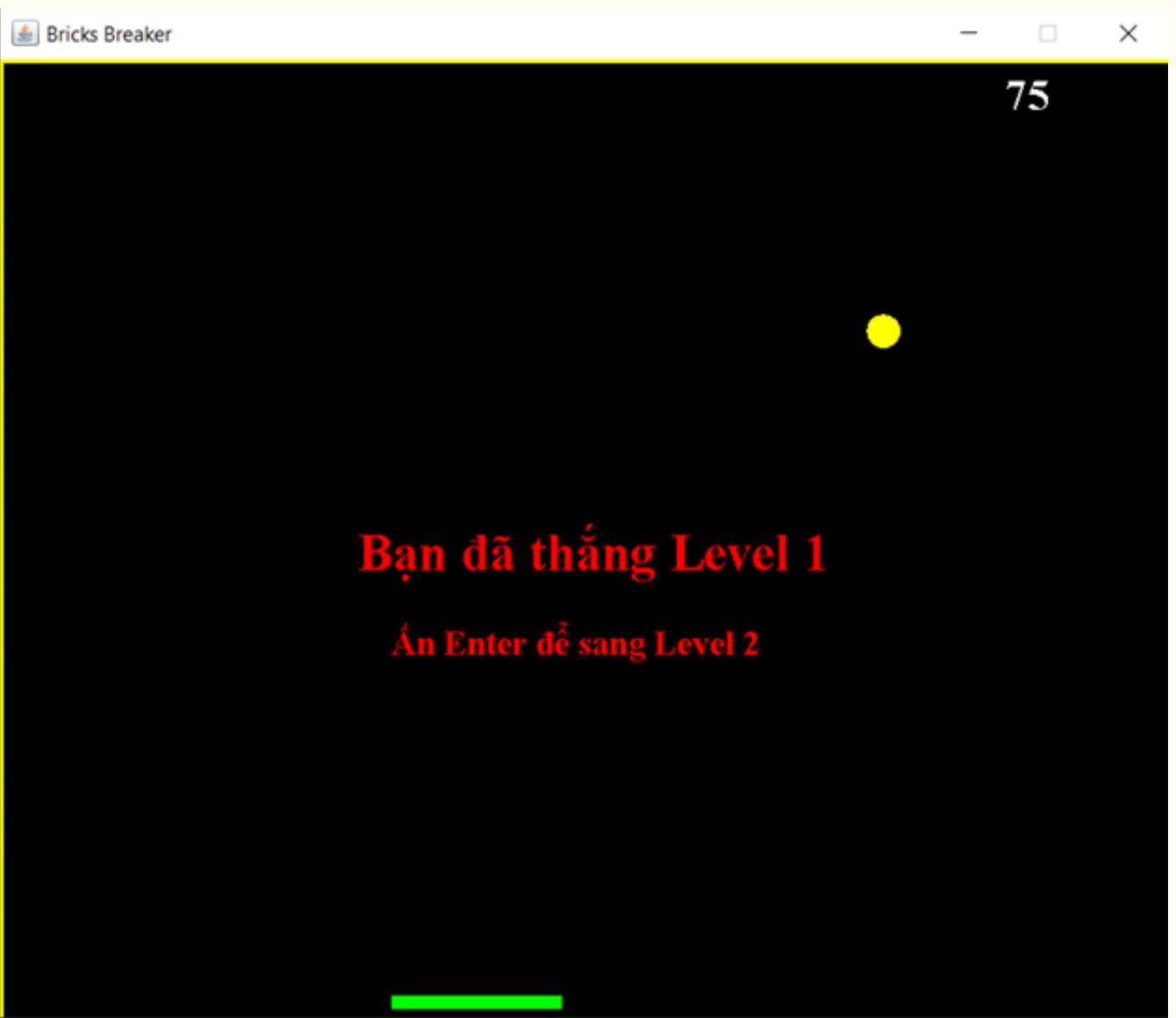


3.3b. Vẽ các màn hình theo các trường hợp

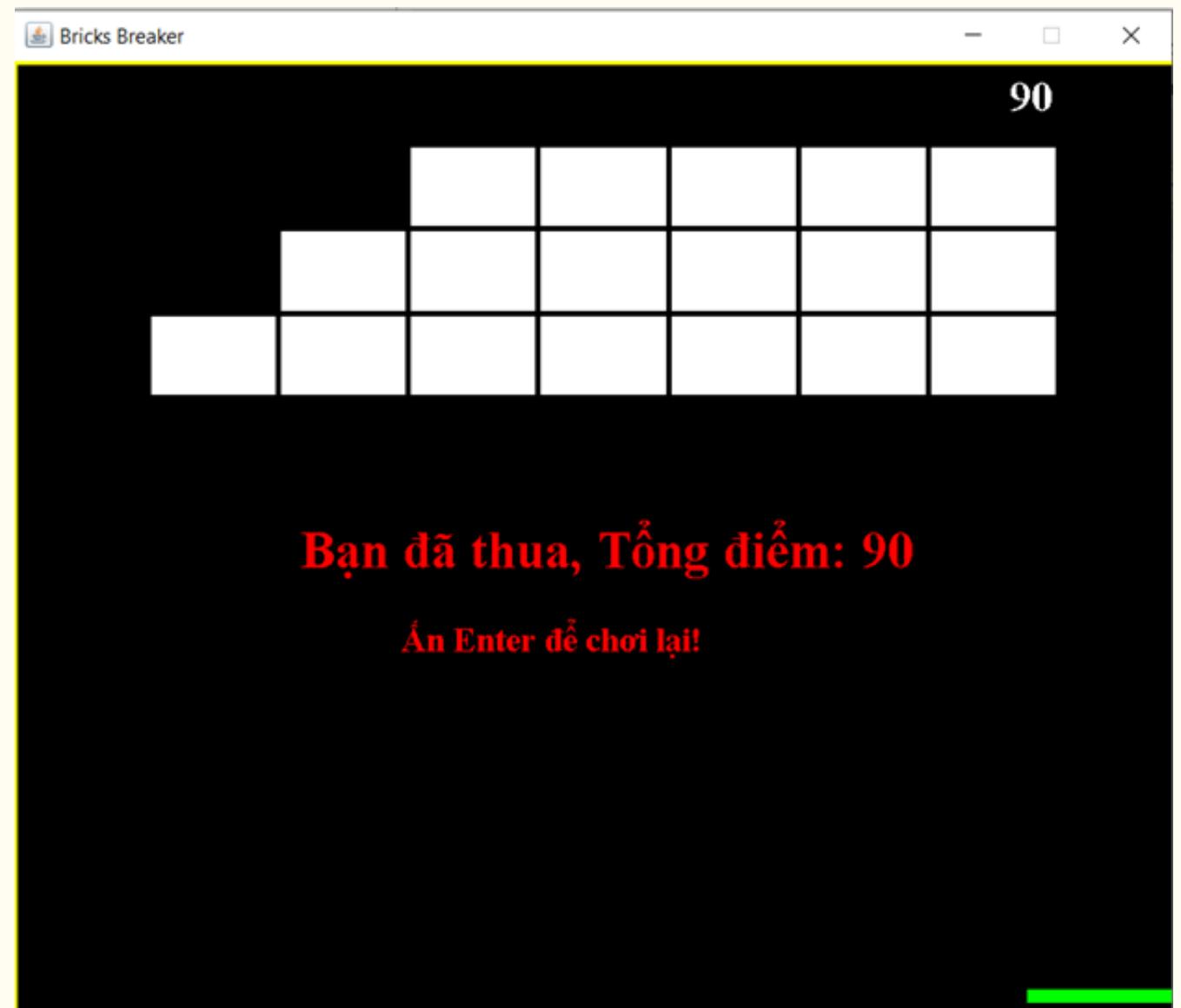
```
if(hello){  
    g.setColor(Color.RED);  
    g.setFont(new Font("serif",Font.BOLD,30));  
    g.drawString("Chào bạn!", 260, 300);  
  
    g.setColor(Color.RED);  
    g.setFont(new Font("serif", Font.BOLD,20));  
    g.drawString("Nhấn Enter để bắt đầu", 230,350);  
}
```



```
1 // khi bạn thắng game
2     if (totalBricks <= 0) {
3         play = false;
4         level++;
5         ballXdir = 0;
6         ballYdir = 0;
7         g.setColor(Color.RED);
8         g.setFont(new Font("serif", Font.BOLD, 30));
9         g.drawString("Bạn đã thắng Level " + (level - 1), 210, 300);
10
11        g.setColor(Color.RED);
12        g.setFont(new Font("serif", Font.BOLD, 20));
13        g.drawString("Ấn Enter để sang Level " + level, 230, 350);
14    }
```



```
1 // khi bạn thua game
2         if (ballposY > 570) {
3             play = false;
4             ballXdir = 0;
5             ballYdir = 0;
6             g.setColor(Color.RED);
7             g.setFont(new Font("serif", Font.BOLD, 30));
8             g.drawString("Bạn đã thua, Tổng điểm: " + score, 170, 300);
9
10            g.setColor(Color.RED);
11            g.setFont(new Font("serif", Font.BOLD, 20));
12            g.drawString("Ấn Enter để chơi lại!", 230, 350);
13
14        }
```





```
1 // khi bạn Pause game
2     if (isPause) {
3         g.setColor(Color.RED);
4         g.setFont(new Font("serif", Font.BOLD, 30));
5         g.drawString("Tạm dừng!", 230, 300);
6
7         g.setColor(Color.RED);
8         g.setFont(new Font("serif", Font.BOLD, 20));
9         g.drawString("Ấn P để tiếp tục", 230, 350);
10    }
```

Tạm dừng!

Ấn P để tiếp tục



3.4. keyPressed

a. Thiết lập nút di chuyển paddle

b. Thiết lập nút Enter

c. Thiết lập nút Pause (P)

```
1 public void keyPressed(KeyEvent e) {
```



a. Thiết lập nút di chuyển paddle



```
1 // set nút di chuyển paddle sang phải
2     if (e.getKeyCode() == KeyEvent.VK_RIGHT) {
3         if (playerX >= 600) {
4             playerX = 600;
5         } else {
6             moveRight();
7         }
8     }
9
10    // set nút di chuyển paddle sang trái
11    if (e.getKeyCode() == KeyEvent.VK_LEFT) {
12        if (playerX < 10) {
13            playerX = 10;
14        } else {
15            moveLeft();
16        }
17    }
```



```
1 public void moveRight() {
2     play = true;
3     playerX += 20;
4 }
5
6 public void moveLeft() {
7     play = true;
8     playerX -= 20;
9 }
```

b. Thiết lập nút Enter



```
1 else if (totalBricks == 0) {  
2     play = false;  
3     ballposX = 120;  
4     ballposY = 350;  
5     ballXdir = -1;  
6     ballYdir = -2;  
7     playerX = 310;  
8     if (level % 2 == 0) {  
9         colNext += 2;  
10    } else {  
11        rowNext++;  
12    }  
13    scoreNext += score;  
14    totalBricks = colNext * rowNext;  
15    map = new MapGenerator(rowNext, colNext);  
16    repaint();  
17}  
18}
```



```
1 // khi thua  
2 if (ballposY > 570) {  
3     getPlayerData();  
4     getPlayerName();  
5     score = 0;  
6     play = true;  
7     ballposX = 120;  
8     ballposY = 350;  
9     ballXdir = -1;  
10    ballYdir = -2;  
11    playerX = 310;  
12    totalBricks = row * col;  
13    map = new MapGenerator(row, col);  
14    repaint();  
15}  
16 }
```

khi thắng thì enter qua level khác

khi thua thì enter chơi lại

Enter để bắt đầu game



```
1 else if (hello) {  
2     hello = false;  
3     play = true;  
4     timer.start();  
5     map = new MapGenerator(row, col);  
6     repaint();  
7 }
```



c. Thiết lập nút Pause - P



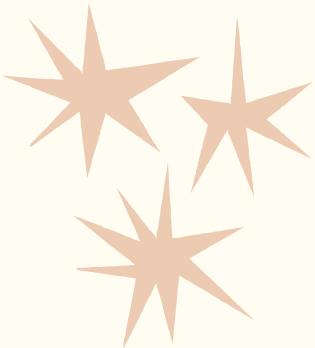
```
1 public void pauseGame() {  
2     if (isPause) {  
3         timer.stop();  
4         repaint();  
5     } else {  
6         timer.start();  
7     }  
8 }
```



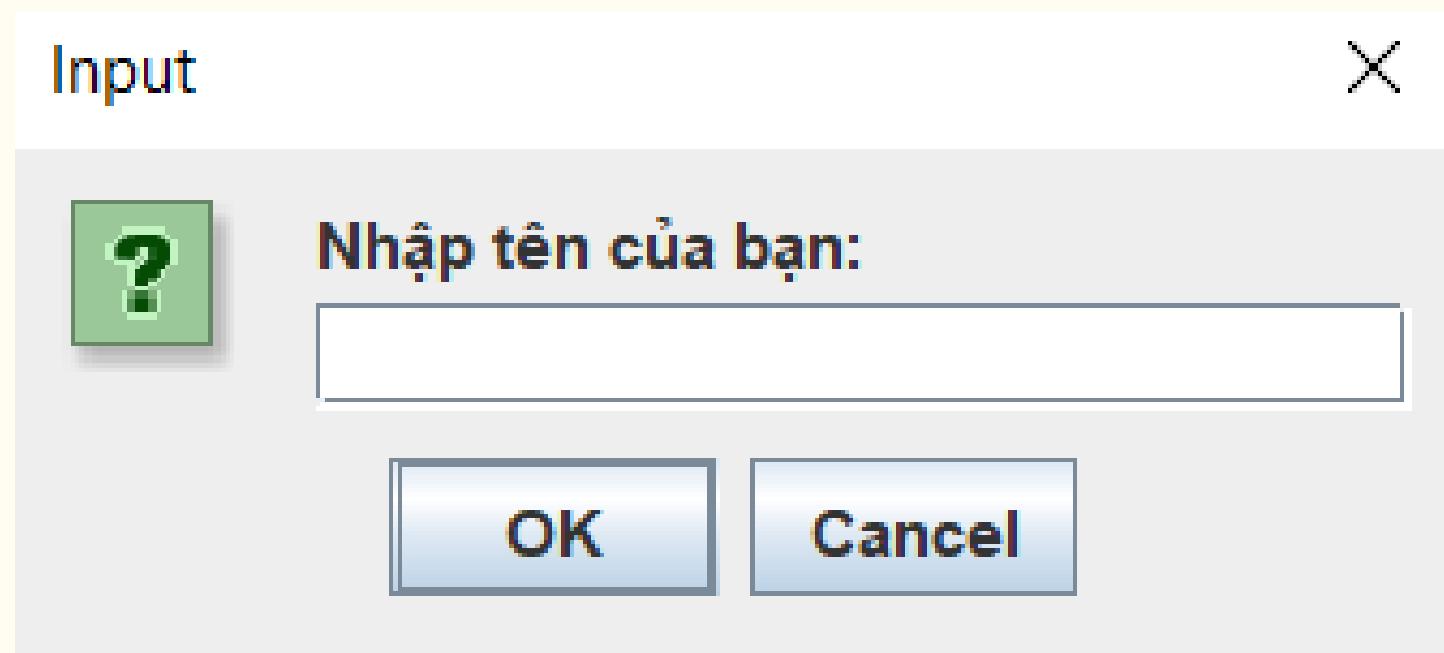
```
1 if (e.getKeyCode() == KeyEvent.VK_P) {  
2     isPause = !isPause;  
3     pauseGame();  
4 }
```

3.5. JOptionPane

Dùng để tạo ô nhập tên người chơi



```
● ● ●  
1 public void getPlayerName() {  
2     playerName = JOptionPane.showInputDialog("Nhập tên của bạn: ");  
3     if (playerName == null || playerName.trim().isEmpty()) {  
4         playerName = "Người-Chơi";  
5     }  
6 }
```



3.6. Các phương thức với File I/O



a. writePlayerData:

```
● ● ●

1 public void writePlayerData() {
2     try (BufferedWriter writer = new BufferedWriter(new FileWriter(PLAYER_DATA_FILE, true))) {
3         writer.write(playerName + " " + score);
4         writer.newLine();
5     } catch (IOException e) {
6         System.out.println("Không thể ghi dữ liệu người chơi vào file: " + e.getMessage());
7     }
8 }
```

Phương thức này ghi dữ liệu vào PLAYER_DATA_FILE, bao gồm:

- **tên người chơi, điểm số, được ngăn cách nhau bằng 1 khoảng trắng**
- **xuống 1 dòng**

b. readPlayerData:

```
● ● ●  
1 public void readPlayerData() {  
2     playersList = new ArrayList<>();  
3  
4     try (BufferedReader reader = new BufferedReader(new FileReader(PLAYER_DATA_FILE))) {  
5         String line;  
6         while ((line = reader.readLine()) != null) {  
7             String[] parts = line.split(" ");  
8             String playerName = parts[0];  
9             int playerScore = Integer.parseInt(parts[1]);  
10            playersList.add(new Player(playerName, playerScore));  
11        }  
12    } catch (IOException e) {  
13        System.out.println("Không thể đọc dữ liệu người chơi từ file: " + e.getMessage());  
14    }  
15 }
```

Phương thức này đọc dữ liệu từ PLAYER_DATA_FILE, bao gồm:

- **đọc dữ liệu từng dòng khác null**
- **tạo mảng chứa playerName và score, tách dòng bằng dấu cách**
để được 2 phần tử trong mảng
- **add đối tượng Player với 2 thuộc tính vào playersList**

c. getPlayerData:

```
● ● ●  
1 public void getPlayerData() {  
2     playersList = new ArrayList<>();  
3     Player player = new Player(playerName, score);  
4     playersList.add(player);  
5     playersList.sort((p1, p2) -> Integer.compare(p2.playerScore, p1.playerScore));  
6     writePlayerData();  
7     showTopPlayers();  
8 }
```

Phương thức này lấy dữ liệu của Player gồm tên và điểm số sau khi thất bại, cụ thể:

- **tạo một ArrayList để lưu trữ thông tin**
- **add đối tượng vừa tạo vào ArrayList trên**
- **sắp xếp danh sách giảm dần theo điểm số, luôn đảm bảo rằng người có điểm cao nhất luôn xuất hiện đầu tiên trong hộp thoại**
- **gọi phương thức writePlayerData**
- **tiếp tục gọi showTopPlayers để hiển thị danh sách**

d. showTopPlayers:

```
● ● ●

1 public void showTopPlayers() {
2     readPlayerData();
3     playersList.sort((p1, p2) -> Integer.compare(p2.playerScore, p1.playerScore));
4     StringBuilder message = new StringBuilder("Danh sách 10 người chơi có điểm cao nhất:\n");
5     int count = Math.min(playersList.size(), 10);
6     for (int i = 0; i < count; i++) {
7         message.append(i + 1).append(". ").append(playersList.get(i).playerName)
8             .append(" - Điểm: ").append(playersList.get(i).playerScore).append("\n");
9     }
10    JOptionPane.showMessageDialog(null, message.toString(), "Top Players", JOptionPane.INFORMATION_MESSAGE);
11 }
```

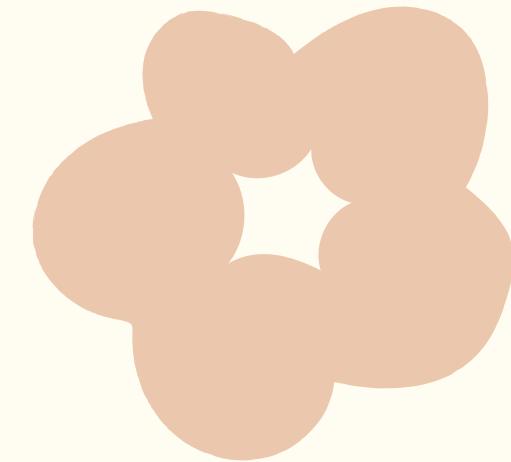
Phương thức này có nhiệm vụ hiển thị danh sách 10 người chơi điểm cao nhất thông qua 1 hộp thoại, cụ thể:

- **đọc dữ liệu của file lưu trữ**
- **sắp xếp danh sách người chơi giảm dần theo điểm số**
- **tạo một đối tượng StringBuilder để tạo thông điệp hiển thị**
- **xác định số lượng người chơi, nếu < 10 thì hiển thị danh sách ít hơn 10 người**
- **hiển thị danh sách người chơi qua hộp thoại**

3.7. actionPerformed



```
1 public void actionPerformed(ActionEvent e) {
```



Phương thức này được sử dụng để kiểm tra các điều kiện hoặc trường hợp trong quá trình trò chơi đang diễn ra, cho nên những phần sau đây sẽ nằm trong if(play)



```
1 public void actionPerformed(ActionEvent e){  
2     timer.start();  
3     if(play){  
4         ....  
5     }  
6 }
```

a. Kiểm tra các trường hợp va chạm giữa bóng và paddle

a1. Nếu bóng va chạm với paddle



```
1 if (new Rectangle(ballposX, ballposY, 20, 20)
2     .intersects(new Rectangle(playerX, 550, 30, 8))) {
3     ballYdir = -ballYdir;
4     ballXdir = -2;
5 }
```

- Sử dụng lớp Rectangle để xác định xem hình dạng của quả bóng có giao cắt với hình dạng của paddle không. Nếu có, thì bóng đang chạm vào paddle.
- Thay đổi hướng di chuyển của quả bóng theo chiều dọc.
- Đổi hướng di chuyển của quả bóng theo chiều ngang.

a2. Nếu bóng va chạm với bên phải hoặc bên trái của paddle



```
1 else if (new Rectangle(ballposX, ballposY, 20, 20)
2     .intersects(new Rectangle(playerX + 70, 550, 30, 8))) {
3     ballYdir = -ballYdir;
4     ballXdir = ballXdir + 1;
5 }
```

- **Kiểm tra xem bóng có chạm vào bên phải hoặc bên trái của paddle không**
- **Thay đổi hướng di chuyển của quả bóng theo chiều dọc.**
- **Đổi hướng di chuyển của quả bóng theo chiều ngang.**

a3. Nếu bóng va chạm ngay giữa của paddle



```
1 else if (new Rectangle(ballposX, ballposY, 20, 20)
2             .intersects(new Rectangle(playerX + 30, 550, 40, 8))) {
3     ballYdir = -ballYdir;
4 }
```

- Kiểm tra xem bóng có chạm vào giữa của paddle không
- Thay đổi hướng di chuyển của quả bóng theo chiều dọc.

b. Kiểm tra va chạm với các viên gạch

```
● ● ●  
1 A:  
2 for (int i = 0; i < map.map.length; i++) {  
3     for (int j = 0; j < map.map[0].length; j++) {  
4         if (map.map[i][j] > 0) {  
5             int brickX = j * map.brickWidth + 80;  
6             int brickY = i * map.brickHeight + 50;  
7             int brickWidth = map.brickWidth;  
8             int brickHeight = map.brickHeight;  
9             Rectangle rect = new Rectangle(brickX, brickY, brickWidth, brickHeight);  
10            Rectangle ballRect = new Rectangle(ballposX, ballposY, 20, 20);  
11  
12            // nếu bóng chạm gạch  
13            if (ballRect.intersects(rect)) {  
14                map.setBrickValue(0, i, j);  
15                score += 5;  
16                totalBricks--;  
17                if (ballposX + 19 <= rect.x || ballposX + 1 >= rect.x + rect.width) {  
18                    ballXdir = -ballXdir;  
19                }  
20                else {  
21                    ballYdir = -ballYdir;  
22                }  
23            break A;  
24        }  
25    }  
26 }  
27 }  
28 }
```

sử dụng 2 vòng lặp duyệt qua tất cả các viên gạch:

- kiểm tra các viên gạch có trọng số = 1
- nếu có va chạm, cho viên gạch đó = 0
- cộng điểm và trừ đi số viên gạch
- xác định hướng va chạm của bóng để thay đổi hướng di chuyển

Tiếp theo sẽ cập nhật tọa độ và kiểm tra bóng có chạm biên hay không



```
1 ballposX += ballXdir;  
2 ballposY += ballYdir;  
3  
4 if (ballposX < 0) {  
5     ballXdir = -ballXdir;  
6 }  
7 if (ballposY < 0) {  
8     ballYdir = -ballYdir;  
9 }  
10 if (ballposX > 670) {  
11     ballXdir = -ballXdir;  
12 }  
13  
14 repaint();
```

- Nếu bóng chạm vào biên trái hoặc phải, đổi hướng di chuyển theo chiều ngang
- Nếu bóng chạm vào biên trên, đổi hướng di chuyển theo chiều dọc.
- Gọi phương thức `repaint()` để vẽ lại giao diện, cập nhật vị trí mới của quả bóng và các viên gạch trên màn hình.

4. Main.java

```
● ● ●  
1 import javax.swing.JFrame;  
2  
3 public class Main {  
4     public static void main(String[] args){  
5         JFrame obj = new JFrame();  
6         Gameplay gamePlay = new Gameplay();  
7         obj.setBounds(10, 10, 700, 600);  
8         obj.setTitle("Bricks Breaker");  
9         obj.setResizable(false);  
10        obj.setVisible(true);  
11        obj.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
12        obj.add(gamePlay);  
13        obj.setVisible(true);  
14    }  
15 }
```

- **tạo 1 đối tượng JFrame mới, đóng vai trò là cửa sổ hiển thị game**
- **tạo một đối tượng lớp Gameplay là lớp chứa logic và hình ảnh đồ họa trò chơi**
- **không cho phép thay đổi kích thước của cửa sổ**
- **add đối tượng gamePlay vào JFrame, để nó hiển thị trong cửa sổ**

4. Kế hoạch phát triển

Các hạng mục cần phát triển:

1. Thêm các option về hình dạng bóng, màu sắc cho người chơi thoái mái lựa chọn
2. Thay đổi về giao diện, thêm âm thanh cho trò chơi sống động hơn
3. Tích hợp thêm các sự hỗ trợ ẩn chứa trong viên gạch như nhân số bóng, cộng thêm số bóng,...
4. Tạo ra thêm nhiều màu sắc cho những viên gạch qua mỗi level của trò chơi

Cảm ơn bạn đã lắng nghe!