

NIM : 2222105206  
Nama : Siti Rohimatuz Zahro  
No. Absen : 39  
Kelas : 2TI03  
Matkul : UAS Pemrograman Berorientasi Objek (PBO)  
Dosen Pengampu : Novan Zulkarnain, S. T., M. Kom.  
Hari, Tanggal : Kamis, 20 Juni 2024

---

## 1. UJIAN AKHIR SEMESTER (UAS) MEMBUAT GAME DARI JAVA

# Menjelajahi Dunia Game Brick Breaker

Brick Breaker, sebuah game klasik yang telah menemani generasi gamer sejak lama, kembali hadir untuk dipelajari secara mendalam. Dalam ujian akhir semester ini, kita akan menyelami dunia Brick Breaker, menguak mekanisme permainan, dan menguraikan kode-kodenya dengan detail.

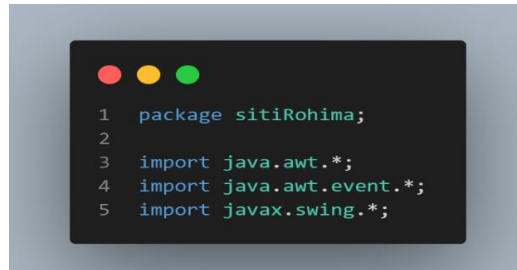
Perjalanan ini akan membawa pembaca menelusuri berbagai aspek penting Brick Breaker, mulai dari konsep dasar game, aturan permainan, hingga implementasi kode-kodenya dalam bahasa pemrograman Java. Kita akan mempelajari bagaimana game ini dirancang, bagaimana ia merespons input pengguna, dan bagaimana ia menghasilkan visualisasi yang menarik.

Baik pembaca seorang gamer yang ingin memahami lebih dalam cara kerja Brick Breaker, maupun seorang programmer yang ingin mempelajari teknik pemrograman game, pendahuluan ini akan menjadi gerbang menuju pemahaman yang komprehensif tentang game klasik ini.

Manfaat:

- Meningkatkan pemahaman tentang game klasik Brick Breaker
- Memperoleh pengetahuan tentang pemrograman game Java
- Mengembangkan kemampuan analisis dan pemecahan masalah
- Memperluas wawasan tentang desain dan implementasi game

**Mari kita bersiap untuk menyelami dunia Brick Breaker dan menguak rahasia di balik keseruannya! Berikut penjelasannya :**



```
1 package sitiRohima;
2
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
```

Kode ini mengatur tahap untuk membuat aplikasi antarmuka pengguna grafis (GUI) di Java. Dengan menggunakan paket `java.awt`, `java.awt.event`, dan `javax.swing`, pengembang memiliki akses ke alat yang diperlukan untuk membangun elemen interaktif, menangani input pengguna, dan membangun komponen visual aplikasi.

### **Detailnya:**

#### **1. Package (`package sitiRohima;`):**

Baris ini bersifat opsional tetapi direkomendasikan untuk mengatur kode. Ini menyatakan bahwa kode berikut adalah milik sebuah paket bernama `sitiRohima`. Fungsinya untuk membantu menghindari konflik penamaan ketika programmer memiliki beberapa kelas dengan nama yang sama dan meningkatkan pengorganisasian kode.

#### **2. Mengimpor (`import java.awt.*;`, `import java.awt.event.*;`, `import javax.swing.*;`):**

Impor yang digunakan ini memberitahu kompiler untuk menyertakan fungsionalitas dari kelas-kelas yang sudah ditulis sebelumnya di dalam pustaka Java.

`import java.awt.*;`: Mengimpor semua kelas dari paket `java.awt`, yang menyediakan fungsi dasar untuk membuat antarmuka pengguna grafis (GUI) di Java, seperti menggambar bentuk, menangani warna, dan mengelola tata letak. Tanpa harus mengirim gambar ke codingan.

`import java.awt.event.*;`: Mengimpor semua kelas dari paket `java.awt.event`, yang menyediakan mekanisme untuk menangani interaksi pengguna dengan elemen GUI seperti klik mouse, penekanan tombol, dan perubahan ukuran jendela.

`import javax.swing.*;`: Mengimpor semua kelas dari paket `javax.swing`, salah satu komponen yang digunakan pada kelas ini yaitu `JFrame` (untuk membuat jendela) dan `JPanel` (untuk membuat panel di dalam jendela).

```

1 public class BrickBreaker extends JPanel implements KeyListener, ActionListener {
2
3     // Game constants
4     private static final int WIDTH = 800;
5     private static final int HEIGHT = 600;
6     private static final int PADDLE_WIDTH = 100;
7     private static final int PADDLE_HEIGHT = 20;
8     private static final int BALL_SIZE = 20;
9     private static final int BRICK_WIDTH = 80;
10    private static final int BRICK_HEIGHT = 30;
11
12    // Game variables
13    private int paddleX = WIDTH / 2;
14    private int paddleY = HEIGHT - PADDLE_HEIGHT - 20;
15    private int ballX = WIDTH / 2;
16    private int ballY = HEIGHT / 2;
17    private int ballDX = 2;
18    private int ballDY = -2;
19    private boolean gameOver = false;
20    private boolean restart = false;
21    private int score = 0;
22
23    // Brick array
24    private boolean[][] bricks = new boolean[10][5];
25
26    // Colors
27    private Color paddleColor = Color.BLUE;
28    private Color ballColor = Color.RED;
29    private Color brickColor = Color.GREEN;

```

Kode ini meletakkan dasar untuk game Brick Breaker dengan mendefinisikan konstanta, variabel, tata letak brick, dan elemen visual game. Komponen-komponen ini kemudian digunakan di bagian lain dari kode untuk menangani mekanisme permainan, interaksi pemain, dan pembaruan visual.

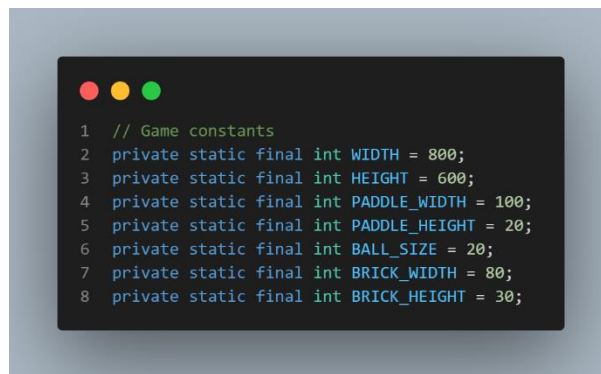
### Detailnya:

#### 1. Definisi Kelas:

*public class BrickBreaker extends JPanel* mengimplementasikan *KeyListener*, *ActionListener* mendefinisikan class *BrickBreaker*:

*extends JPanel*: Mewarisi properti dan metode dari kelas *JPanel*, yang merupakan blok bangunan fundamental untuk membuat panel GUI. *KeyListener* dan *ActionListener* untuk mengindikasikan bahwa kelas tersebut akan menangani peristiwa keyboard (untuk gerakan paddle) dan peristiwa pengatur waktu (untuk perulangan permainan).

## 2. Konstanta Game:

A screenshot of a code editor window with a dark background and light-colored text. The code defines several static final integer constants for game parameters. The window has three colored window control buttons (red, yellow, green) in the top-left corner.

```
1 // Game constants
2 private static final int WIDTH = 800;
3 private static final int HEIGHT = 600;
4 private static final int PADDLE_WIDTH = 100;
5 private static final int PADDLE_HEIGHT = 20;
6 private static final int BALL_SIZE = 20;
7 private static final int BRICK_WIDTH = 80;
8 private static final int BRICK_HEIGHT = 30;
```

Konstanta ini mendefinisikan nilai tetap untuk berbagai aspek permainan seperti:

- a. Lebar dan tinggi jendela game
- b. Lebar dan tinggi paddle
- c. Diameter bola
- d. Lebar dan tinggi 1 brick

## 3. Variabel Game:

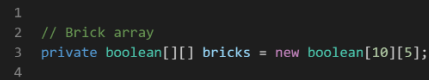
A screenshot of a code editor window with a dark background and light-colored text. The code defines several private integer and boolean variables for game state and player position. The window has three colored window control buttons (red, yellow, green) in the top-left corner.

```
1 // Game variables
2 private int paddleX = WIDTH / 2;
3 private int paddleY = HEIGHT - PADDLE_HEIGHT - 20;
4 private int ballX = WIDTH / 2;
5 private int ballY = HEIGHT / 2;
6 private int ballDX = 2;
7 private int ballDY = -2;
8 private boolean gameOver = false;
9 private boolean restart = false;
10 private int score = 0;
```

Variabel-variabel ini menyimpan status dinamis permainan:

- a. *paddleX* dan *paddleY*: Koordinat X dan Y awal dari paddle pemain (berada di tengah secara horizontal dan di dekat bagian bawah).
- b. *ballX* dan *ballY*: Koordinat X dan Y awal dari bola (berada di tengah horizontal dan vertikal).
- c. *ballDX* dan *ballDY*: Gerakan (arah dan kecepatan) horizontal dan vertikal awal bola (2 piksel per frame untuk keduanya di sini).
- d. *gameOver*: Boolean yang menunjukkan apakah permainan telah berakhir (bernilai false).
- e. *restart*: Boolean yang menunjukkan apakah pemain telah meminta untuk memulai ulang (bernilai false).
- f. *score*: Variabel integer untuk melacak skor pemain (skor awal 0).

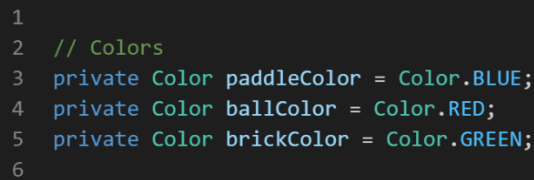
#### 4. brick array:



```
1
2 // Brick array
3 private boolean[][] bricks = new boolean[10][5];
4
```

*private boolean[][] brick = new boolean[10][5];*:: Membuat array boolean 2D brick dengan 10 baris (mewakili baris brick) dan 5 kolom (mewakili kolom brick). Setiap elemen dalam array (*brick [i][j]*) bernilai *true* jika brick yang bersangkutan ada dan *false* jika brick tersebut hancur/pecah. Hal ini memungkinkan permainan untuk melacak keadaan setiap bata.

#### 5. Color:



```
1
2 // Colors
3 private Color paddleColor = Color.BLUE;
4 private Color ballColor = Color.RED;
5 private Color brickColor = Color.GREEN;
6
```

*private Color paddleColor = Color.BLUE;*:: Mendefinisikan warna paddle pemain (biru).

*private Color ballColor = Color.RED;*:: Mendefinisikan warna bola (merah).

*private Color brickColor = Color.GREEN;*:: Mendefinisikan warna brick (hijau).



Potongan kode ini menyiapkan jendela permainan, mengatur penanganan input pengguna, menginisialisasi brick di posisi awal, dan membuat loop permainan inti menggunakan timer. Metode *actionPerformed*, yang dipanggil berulang kali oleh *timer*, akan menangani logika permainan seperti memperbarui posisi bola, memeriksa tabrakan, memperbarui skor, dan menangani skenario game over dan restart.

#### Detailnya:

Potongan kode dari konstruktor kelas *BrickBreaker* ini (*public BrickBreaker() { ... }*) melakukan beberapa tugas penting untuk menginisialisasi lingkungan permainan:

##### 1. Setting Up for User Input:

- a. *addKeyListener(this);*: untuk memungkinkan objek menerima dan menangani peristiwa keyboard (khususnya, penekanan tombol) di dalam kode. ini sangat penting untuk mengontrol gerakan paddle pemain.
- b. *setFocusable(true);*: Membuat panel fokus.

##### 2. Mendefinisikan Jendela Permainan:

*setPreferredSize(new Dimension(WIDTH, HEIGHT));*: Mengatur ukuran yang diinginkan dari panel permainan ke nilai yang ditentukan dalam konstanta *WIDTH* dan *HEIGHT*. Ini menentukan dimensi jendela permainan dalam piksel.

##### 3. Mengatur Warna Latar Belakang:

*setBackground(Color.BLACK);*: Mengatur warna latar belakang panel permainan menjadi hitam. Ini memberikan kanvas kosong untuk menampilkan elemen permainan (paddle, bola, brick, skor).

##### 4. Menginisialisasi brick:

- a. *for (int i = 0; i < 10; i++) { ... }*: Perulangan ini mengulang sebanyak 10 kali, mewakili 10 brick bata di dalam permainan.
- b. *for (int j = 0; j < 5; j++) { ... }*: Di dalam out loop, loop interested ini mengulang sebanyak 5 kali, merepresentasikan 5 kolom brick di setiap baris.

- c. *brick[i][j] = true*:: Baris ini menetapkan nilai pada setiap elemen (*bricks[i][j]*) dalam array brick 2D menjadi true. Karena array brick adalah array boolean, nilai true mengindikasikan bahwa brick ada di lokasi tersebut di dalam kotak permainan.

#### 5. Membuat Pengatur Timer Loop Game:

*Timer timer = new Timer(10, this)*:: Membuat objek Timer baru. Argumen pertama (*10*) menentukan penundaan (dalam milidetik) di antara detak timer. Di sini, ini diatur ke 10 milidetik, yang berarti timer akan memicu peristiwa setiap 10 milidetik. Argumen kedua (*this*) mengacu pada objek BrickBreaker saat ini, yang mengimplementasikan antarmuka *ActionListener* (seperti yang terlihat pada definisi kelas).

*timer.start()*:: Memulai pengatur waktu. Ini memulai perulangan permainan, karena setiap 10 milidetik, metode *actionPerformed* dari objek BrickBreaker akan dipanggil (karena ini merujuk ke objek saat ini).

```

1 // pengaturan komponen dalam game brick breaker
2 @Override
3 public void paintComponent(Graphics g) {
4     super.paintComponent(g);
5
6     // membuat paddle
7     g.setColor(paddleColor);
8     g.fillRect(paddleX, paddleY, PADDLE_WIDTH, PADDLE_HEIGHT);
9
10    // membuat ball
11    g.setColor(ballColor);
12    g.fillOval(ballX, ballY, BALL_SIZE, BALL_SIZE);
13
14    // membuat bricks
15    for (int i = 0; i < 10; i++) {
16        for (int j = 0; j < 5; j++) {
17            if (bricks[i][j]) {
18                g.setColor(brickColor);
19                g.fillRect(i * BRICK_WIDTH, j * BRICK_HEIGHT, BRICK_WIDTH, BRICK_HEIGHT);
20            }
21        }
22    }
23
24    // membuat score
25    g.setColor(Color.WHITE);
26    g.drawString("Score: " + score, 10, 20);
27
28    // membuat game over message
29    if (gameOver) {
30        g.setColor(Color.RED);
31        g.drawString("Game Over! Press R to restart.", WIDTH / 2 - 100, HEIGHT / 2);
32    }
33 }

```

metode ini menggambarkan semua elemen visual permainan berdasarkan status variabel permainan saat ini (paddleX, paddleY, ballX, ballY, susunan brick, skor, dan gameOver). Metode ini akan dipanggil setiap kali keadaan permainan berubah atau ganti warna untuk memastikan visualnya mencerminkan permainan yang sedang berlangsung

#### 1. Memanggil Metode Superclass:

*super.paintComponent(g);* adalah baris yang memanggil metode *paintComponent* dari superclass (JPanel). Hal ini memastikan bahwa setiap perilaku pengecatan default dari kelas induk dieksekusi terlebih dahulu.

#### 2. Menggambar Paddle:

*g.setColor(paddleColor);* Mengatur warna gambar ke *paddleColor* yang ditentukan (mungkin biru berdasarkan kode sebelumnya).

*g.fillRect(paddleX, paddleY, PADDLE\_WIDTH, PADDLE\_HEIGHT);* Menggambar persegi panjang, koordinat X dan Y paddle (paddleX dan paddleY), serta lebar dan tingginya (*PADDLE\_WIDTH* dan *PADDLE\_HEIGHT*). Hal ini secara visual merepresentasikan paddle pemain di layar.



### 3. Menggambar Bola:

*g.setColor(ballColor);*: Mengatur warna gambar ke warna bola yang ditentukan (mungkin merah berdasarkan kode sebelumnya).

*g.fillOval(ballX, ballY, BALL\_SIZE, BALL\_SIZE);*: Menggambar sebuah oval terisi menggunakan warna gambar saat ini, koordinat X dan Y bola (ballX dan ballY), dan diameternya (BALL\_SIZE). Ini akan membuat bola melingkar di layar.

### 4. Menggambar Brick:


- *for (int i = 0; i < 10; i++) { ... }*: Perulangan ini mengulang sebanyak 10 kali, mewakili 10 baris brick.
- *for (int j = 0; j < 5; j++) { ... }*: Di dalam out loop, loop interested ini mengulang sebanyak 5 kali, merepresentasikan 5 kolom brick pada setiap baris.
- *if (batu bata[i][j]) { ... }*: Pernyataan ini memeriksa apakah elemen yang sesuai (brick [i] [j]) dalam array brick 2D adalah true. Jika true (berarti brick ada di lokasi tersebut), kode berikut akan dieksekusi:
- *g.setColor(brickColor);*: Mengatur warna gambar ke warna brick yang ditentukan (mungkin hijau berdasarkan kode sebelumnya).
- *g.fillRect(i \* BRICK\_WIDTH, j \* BRICK\_HEIGHT, BRICK\_WIDTH, BRICK\_HEIGHT);*: Menggambar persegi panjang yang terisi menggunakan warna gambar saat ini, koordinat X dan Y yang dihitung berdasarkan penghitung perulangan (i dan j) dikalikan dengan lebar dan tinggi batu bata (BRICK\_WIDTH dan BRICK\_HEIGHT), dan lebar dan tinggi masing-masing batu bata. Hal ini akan menghasilkan masing-masing bata pada layar.

### 5. Menggambar Skor:

*g.setColor(Color.WHITE);*: Mengatur warna gambar menjadi putih untuk menampilkan skor.  
*g.drawString("Skor: " + skor, 10, 20);*: Menggambar sebuah string teks "Skor: " diikuti dengan nilai skor saat ini, yang diposisikan pada (10, 20) di layar. Ini akan menampilkan skor pemain di sudut kiri atas.

### 6. Menggambar message Game Over (jika ada):

- *if (gameOver) { ... }*: Pernyataan bersyarat ini memeriksa apakah flag gameOver diatur ke true. Jika true (yang berarti permainan selesai), kode berikut akan dijalankan:
- *g.setColor(Color.RED);*: Mengatur warna gambar menjadi merah untuk menyoroti pesan game over.
- *g.drawString("Game Over! Tekan R untuk memulai ulang.", WIDTH / 2 - 100, HEIGHT / 2);*: Menggambar sebuah string teks "Game Over! Tekan R untuk memulai kembali." yang dipusatkan secara horizontal (menggunakan WIDTH / 2 - 100) dan vertikal (menggunakan HEIGHT / 2) di layar. Ini memberi tahu pemain tentang status permainan dan opsi untuk memulai ulang.



```

1  @Override
2  public void keyPressed(KeyEvent e) {
3      int key = e.getKeyCode();
4
5      if (key == KeyEvent.VK_LEFT) {
6          paddleX -= 10;
7      } else if (key == KeyEvent.VK_RIGHT) {
8          paddleX += 10;
9      } else if (key == KeyEvent.VK_R) {
10         restart = true;
11     }
12 }
13
14 @Override
15 public void keyReleased(KeyEvent e) {
16 }
17
18 @Override
19 public void keyTyped(KeyEvent e) {
20 }
21

```

Potongan kode ini menyediakan kontrol pemain dasar untuk game Brick Breaker. Dengan menekan tombol panah kiri dan kanan, pengguna dapat menggerakkan dayung secara horizontal untuk membelokkan bola. Selain itu, menekan tombol 'R' akan memicu permainan dimulai ulang ketika flag gameOver diatur. Kode ini mendefinisikan tiga metode di dalam kelas BrickBreaker yang menangani berbagai aspek interaksi keyboard:

#### Detailnya:

##### 1. *@Override public void keyPressed(KeyEvent e):*

Metode ini dipanggil setiap kali user menekan sebuah tombol ketika jendela permainan sedang terfokus. *int key = e.getKeyCode();* adalah baris yang mengambil kode dari tombol yang ditekan dan menyimpannya di variabel key menggunakan metode *getKeyCode()* dari objek KeyEvent (e). Pernyataan bersyarat (*if, else if*) memeriksa nilai variabel key untuk menentukan tombol mana yang ditekan:

- a. *if (key == KeyEvent.VK\_LEFT) { ... }:* Jika kode tombol yang ditekan sesuai dengan `KeyEvent.VK_LEFT` (konstanta yang merepresentasikan tombol panah kiri), koordinat X dayung (`paddleX`) akan berkurang 10 piksel, yang secara efektif menggerakkan dayung ke arah kiri.
- b. *else if (key == KeyEvent.VK\_RIGHT) { ... }:* Jika kode tombol yang ditekan sesuai dengan `KeyEvent.VK_RIGHT` (konstanta yang merepresentasikan tombol panah kanan), koordinat X dayung (`paddleX`) akan

bertambah 10 piksel, yang secara efektif  
menggerakkan dayung ke arah kanan.

- c. *else if (key == KeyEvent.VK\_R) { ... }*: Jika kode tombol yang ditekan sesuai dengan `KeyEvent.VK_R` (konstanta yang mewakili tombol 'R'), maka flag restart akan disetel ke `true`. Ini kemungkinan besar digunakan untuk memberi sinyal pada perulangan permainan (diimplementasikan di tempat lain) untuk memulai ulang permainan.

2. *@Override public void keyReleased(KeyEvent e) (Kosong):*

Metode ini, meskipun didefinisikan, saat ini kosong (`{ }`). Ini mungkin dimaksudkan untuk digunakan sewaktu-waktu untuk menangani tindakan spesifik yang perlu diambil ketika pengguna melepaskan kunci.

3. *@Override public void keyTyped(KeyEvent e) (Kosong):*

Metode ini, meskipun didefinisikan, saat ini kosong (`{ }`). Ini mungkin dimaksudkan untuk digunakan sewaktu-waktu untuk menangani input karakter (huruf atau angka yang diketik).

```

1  @Override
2  public void actionPerformed(ActionEvent e) {
3      // Update ball position
4      ballX += ballDX;
5      ballY += ballDY;
6
7      // Collision with walls
8      if (ballX < 0 || ballX > WIDTH - BALL_SIZE) {
9          ballDX = -ballDX;
10     }
11     if (ballY < 0) {
12         ballDY = -ballDY;
13     }
14
15     // Collision with paddle
16     if (ballY > paddleY - BALL_SIZE && ballX > paddleX && ballX < paddleX + PADDLE_WIDTH) {
17         ballDY = -ballDY;
18     }
19
20     // Collision with bricks
21     for (int i = 0; i < 10; i++) {
22         for (int j = 0; j < 5; j++) {
23             if (bricks[i][j] && ballX > i * BRICK_WIDTH && ballX < i * BRICK_WIDTH + BRICK_WIDTH
24                 && ballY > j * BRICK_HEIGHT && ballY < j * BRICK_HEIGHT + BRICK_HEIGHT) {
25                 bricks[i][j] = false;
26                 score++;
27                 ballDY = -ballDY;
28             }
29         }
30     }
31
32     // Game over
33     if (ballY > HEIGHT) {
34         gameOver = true;
35     }
36
37     // Restart game
38     if (restart) {
39         gameOver = false;
40         restart = false;
41         score = 0;
42         ballX = WIDTH / 2;
43         ballY = HEIGHT / 2;
44         ballDX = 2;
45         ballDY = -2;
46         paddleX = WIDTH / 2;
47         paddleY = HEIGHT - PADDLE_HEIGHT - 20;
48
49         // Reset bricks
50         for (int i = 0; i < 10; i++) {
51             for (int j = 0; j < 5; j++) {
52                 bricks[i][j] = true;
53             }
54         }
55     }
56
57     // Repaint the panel
58     repaint();
59 }

```

Kode ini mendefinisikan metode *actionPerformed* di dalam kelas *BrickBreaker*. Metode ini adalah inti dari perulangan permainan, karena metode ini dipanggil setiap 10 milidetik oleh timer yang Anda lihat sebelumnya.

### Detailnya:

#### 1. Memperbarui Posisi Bola :

*ballX += ballDX;* untuk memperbarui koordinat X bola dengan menambahkan nilai pergerakan horizontalnya (*ballDX*). *ballY += ballDY;* untuk memperbarui koordinat Y bola dengan menambahkan nilai pergerakan vertikalnya (*ballDY*).

#### 2. Tabrakan dengan Dinding :

*if (ballX < 0 || ballX > WIDTH - BALL\_SIZE) { ... }* untuk memeriksa apakah koordinat X bola akan keluar dari layar (kurang dari 0 atau lebih besar dari LEBAR dikurangi ukuran bola).

*ballDX = -ballDX;* Membalikkan arah gerakan horizontal bola dengan mengalikan *ballDX* dengan -1. Ini akan memantulkan bola kembali ketika mengenai dinding kiri atau kanan.

*if (ballY < 0) { ... }* untuk memeriksa apakah koordinat Y bola akan berada di atas bagian atas layar (kurang dari 0).

*ballDY = -ballDY;* untuk membalikkan arah gerakan vertikal bola dengan mengalikan *ballDY* dengan -1. Ini akan memantulkan bola kembali saat mengenai dinding atas.

### 3. Tabrakan Paddle :

*if (ballY > paddleY - BALL\_SIZE && ballX > paddleX && ballX < paddleX + PADDLE\_WIDTH)*  
*{ ... }* : Pernyataan bersyarat ini memeriksa tabrakan antara bola dan paddle.

Jika kedua kondisi tersebut benar, tabrakan telah terjadi:

*ballDY = -ballDY;* Membalikkan arah gerakan vertikal bola, secara efektif memantulkannya dari paddle.

### 4. Tabrakan Brick :

Perulangan bersarang mengulang seluruh susunan brick (10 baris dan 5 kolom):

*for (int i = 0; i < 10; i++) { ... }* (perulangan luar untuk baris)

*for (int j = 0; j < 5; j++) { ... }* (perulangan dalam untuk kolom)

Di dalam perulangan, pernyataan kondisional lain memeriksa tabrakan antara bola dan brick tertentu.

Jika semua kondisi benar, tabrakan dengan brick telah terjadi:

*brick[i][j] = false;* untuk mengatur elemen brick yang sesuai dalam array brick menjadi false, secara efektif menandainya sebagai hancur/pecah dan menghapusnya dari permainan.

*skor++;* Menambah skor pemain sebesar 1.

*ballDY = -ballDY;* untuk membalikkan arah gerakan vertikal bola, memantulkannya dari bata.

### 5. Game Over :

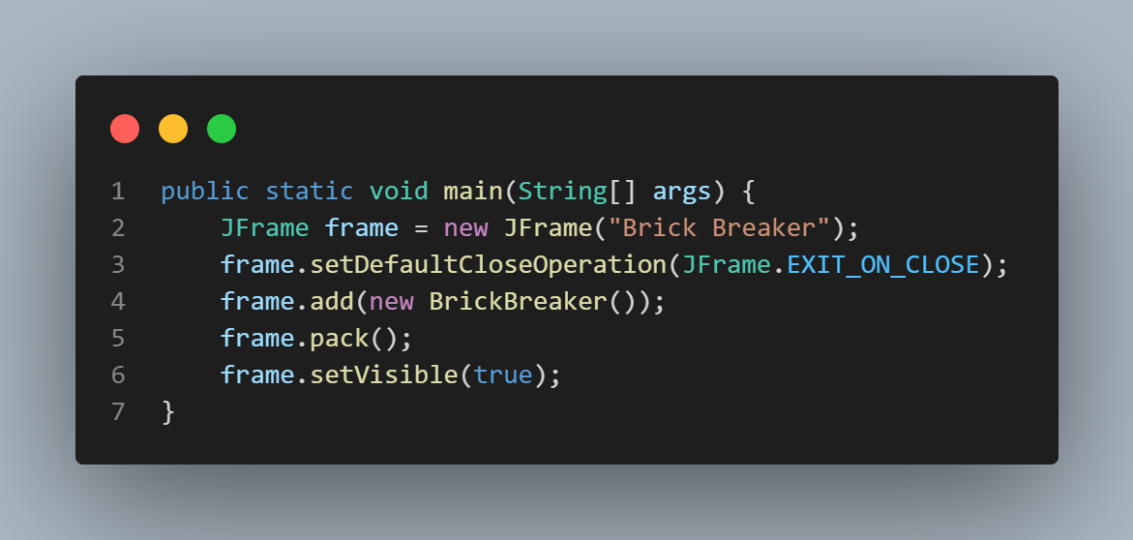
*if (ballY > HEIGHT) { ... }* untuk memeriksa apakah koordinat Y bola telah berada di bawah bagian bawah layar (lebih besar dari HEIGHT). Jika ya : *gameOver = true* untuk mengatur flag *gameOver* menjadi true, yang mengindikasikan bahwa permainan telah berakhir.

### 6. Memulai Ulang Permainan (Sudah Dijelaskan):

*if (restart) { ... }* untuk memeriksa apakah restart diatur ke true (menunjukkan bahwa pemain telah menekan tombol 'R' untuk memulai ulang). Jika ya, ini akan mengatur ulang status permainan.

### 7. Repaint Panel:

*repaint();* untuk memanggil metode *repaint()* dari komponen *JPanel*. Hal ini memicu sistem untuk menggambar ulang elemen permainan di layar, yang mencerminkan posisi dan status objek permainan yang telah diperbarui.



```

1  public static void main(String[] args) {
2      JFrame frame = new JFrame("Brick Breaker");
3      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
4      frame.add(new BrickBreaker());
5      frame.pack();
6      frame.setVisible(true);
7  }

```

Kode ini mendefinisikan metode utama, yang merupakan titik masuk program Java yang dibuat.

#### Detailnya:

1. Membuat Bingkai Permainan:

*JFrame frame = new JFrame("Brick Breaker");* adalah baris membuat objek JFrame baru bernama frame. JFrame adalah kelas inti dalam pustaka Swing Java yang merepresentasikan sebuah jendela standar pada layar. Konstruktor menetapkan judul jendela menjadi "Brick Breaker".

2. Mengatur Perilaku Penutupan:

*frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);* adalah baris yang mengonfigurasi bagaimana program harus berperilaku ketika pengguna menutup jendela. Di sini, *setDefaultCloseOperation* digunakan dengan argumen *JFrame.EXIT\_ON\_CLOSE*. Ini memberitahu window manager untuk menghentikan seluruh aplikasi Java ketika pengguna mengklik tombol tutup pada jendela.

3. Menambahkan Panel Permainan:

*frame.add(new BrickBreaker());* adalah baris yang menambahkan sebuah instance baru dari kelas BrickBreaker ke dalam frame. Metode tambah JFrame mengambil objek Komponen sebagai argumen, dan BrickBreaker kemungkinan merupakan subkelas dari Komponen.

4. Memasang Frame:

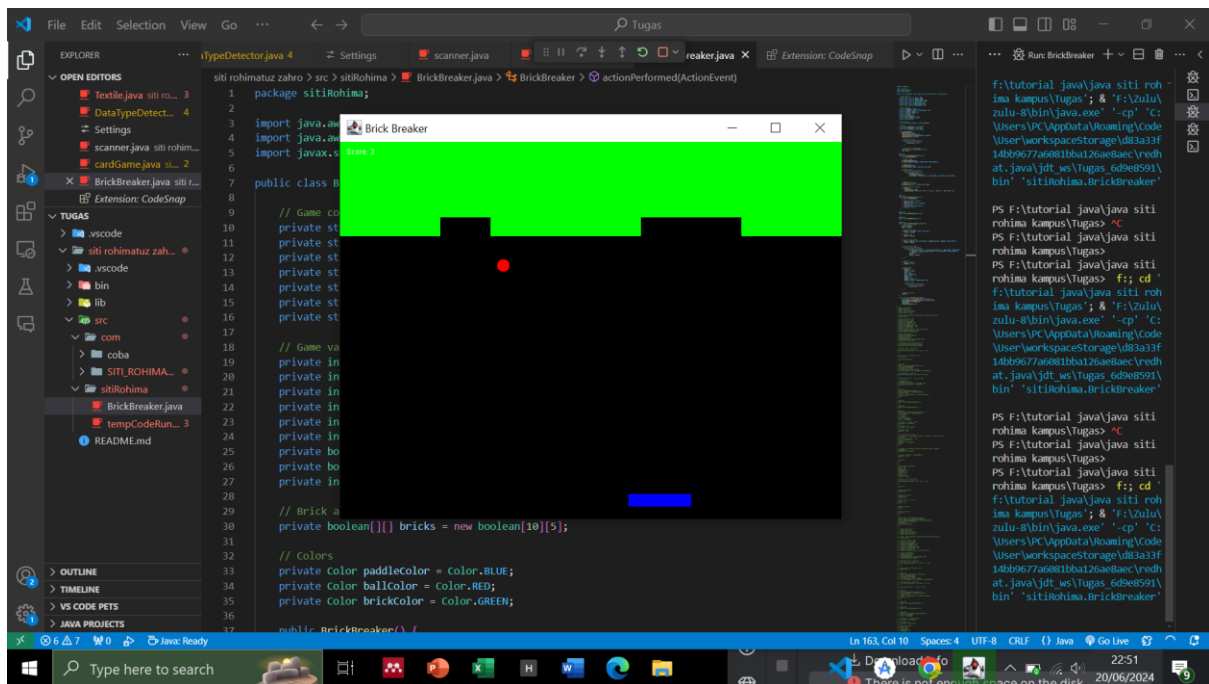
*frame.pack();* adalah baris yang memanggil metode pack pada objek frame. Metode pack secara otomatis mengubah ukuran frame agar sesuai dengan ukuran yang diinginkan dari komponen-komponennya. Hal ini memastikan bahwa semua elemen game ditampilkan dengan benar di dalam jendela.

5. Membuat Frame Terlihat:

*frame.setVisible(true);* adalah baris yang mengatur visibilitas frame menjadi true. Hal ini membuat jendela muncul di layar pengguna, sehingga mereka dapat berinteraksi dengan game.

Singkatnya, metode utama (Main methode) bertanggung jawab untuk mengatur jendela game, menambahkan komponen game, dan membuat semuanya terlihat oleh pengguna. Ini bertindak sebagai titik awal untuk aplikasi game Brick Breaker.

Tampilan window dari game BRICK BREAKER saat di run



Tampilan window dari game BRICK BREAKER saat game over

