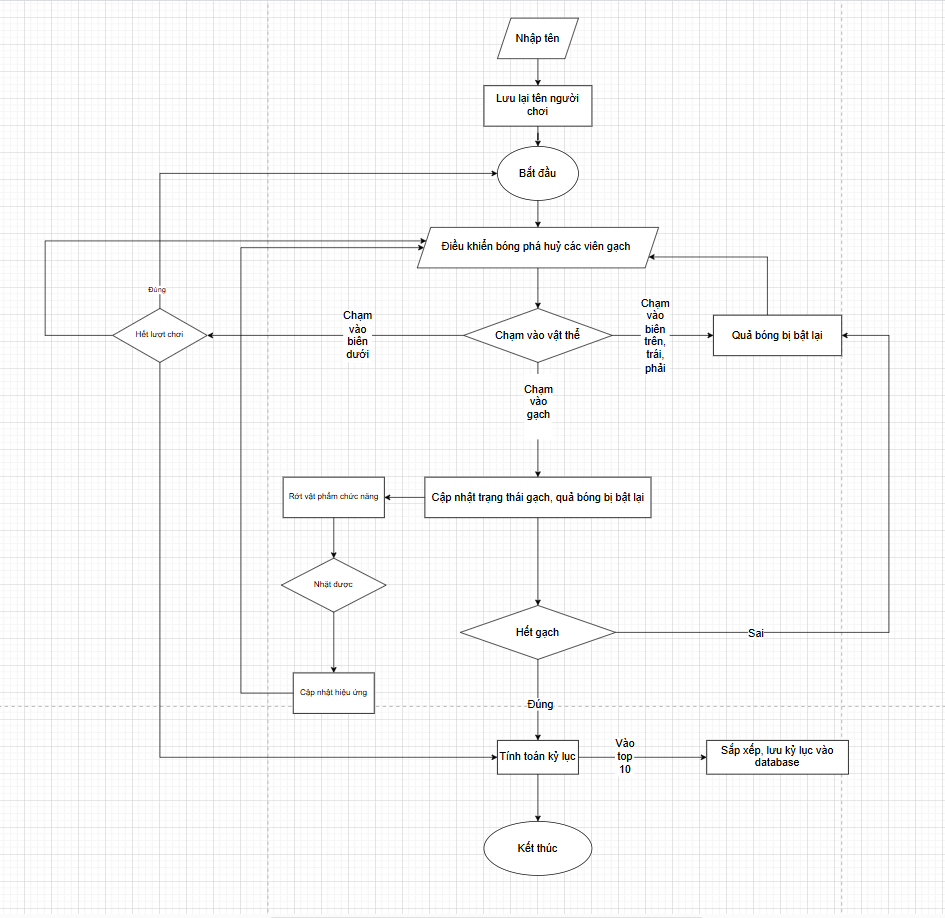


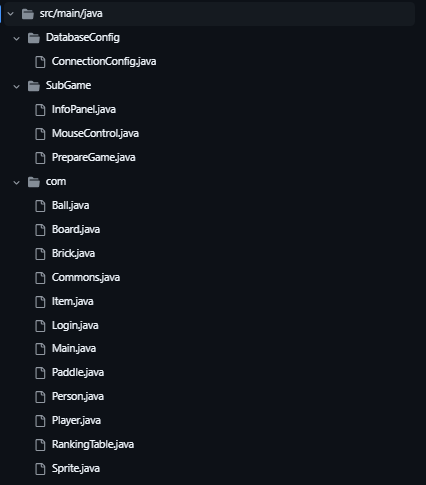
1. **Giới thiệu đề tài:**
2. **Tên đề tài:** Trò chơi Breakout Ball
3. **Mô tả trò chơi:**

* Người chơi (Player) sẽ được yêu cầu nhập tên, sau đó nhấn nút “Get started” để vào trò chơi. Mỗi người chơi có 3 mạng, sau khi kết thúc sẽ hiển thị ra bảng danh sách 10 người chơi có số điểm cao nhất từ cơ sở dữ liệu của trò chơi.
* Trò chơi gồm 1 quả bóng (Ball), 1 thanh đỡ bóng (Paddle) và các khối gạch (Brick). Người chơi sẽ điều khiển thanh đỡ bóng giúp bóng phá vỡ các viên gạch. Bóng sẽ bật lại khi chạm vào gạch, cạnh trên và 2 cạnh bên màn hình. Với mỗi viên gạch bị phá vỡ, điểm của người chơi sẽ tăng lên và có tỉ lệ rơi ra vật phẩm (Item) với hiệu ứng có lợi hoặc bất lợi, khi vật phẩm chạm vào thanh đỡ bóng, hiệu ứng sẽ được áp dụng trong vòng 5 giây. Ngoài phá gạch, người chơi phải đảm bảo không để bóng rơi xuống cạnh dưới màn hình, nếu rơi người chơi sẽ mất 1 mạng. Trò chơi sẽ kết thúc khi người chơi hết số mạng (Thất bại) hoặc phá vỡ hết gạch (Chiến thắng).

1. **Lưu đồ giải thuật:**

* **Màn hình đăng nhập, kết nối database, theo dõi tiến trình, di chuyển của bóng và thanh đỡ, in ra bảng xếp hạng (Minh)**
* **Xử lý va chạm của bóng với gạch, thanh đỡ và các cạnh màn hình (Tuấn Anh)**
* **Tạo gạch, vật phẩm và xử lý các hiệu ứng của vật phẩm (Tú)**

1. **Cây thư mục:**

****

1. **Các chức năng:**

**Package com:**

1. **startGame(), stopGame() (Minh):**

* startGame():
* Tạo 1 sảnh chờ và thêm sự kiện sau khi click chuột vào bất kỳ nơi nào sẽ tạo 1 thread, gán vào biến clock và bắt đầu trò chơi.
* stopGame():
* Kết thúc game, đưa giá trị clock = null

1. **addInfoPanel()(Minh):**

* Đưa 2 panel trên và dưới theo dõi thông tin người dùng và tiến trình trò chơi.

1. **addPrepareGame(Minh):**

* Tạo 1 giao diện chờ để tăng tính trải nghiệm người chơi.

1. **@Override run (Minh):**

* Là hàm được override lại từ lớp runnable dùng để can thiệp vào quá trình game đang chạy và kiểm soát các thông tin trong game.
* Hàm gồm có 3 phần chính:
* Update(): Gọi hàm update để cập nhật lại vị trí
* Repaint(): Vẽ lại các component
* Maintain FPS: Theo dõi FPS

1. **createBrick(Brick[] brick) (Tú):**

* Dùng 2 vòng lặp for để tạo từng viên gạch từ trên xuống dưới, từ trái qua phải. Biến amount\_brick tăng dần khi mỗi viên gạch được tạo để tính tổng số gạch, phục vụ cho công việc vẽ gạch.
* Tọa độ viên gạch đầu tiên sẽ là (0,0) nên toàn bộ khối gạch sẽ nằm lệch về sát góc trên bên trái màn hình. dùng 2 biến margin\_top và margin\_side để căn lề cho toàn bộ khối gạch.
* margin\_side: bằng chiều dài của màn hình trừ đi tích của chiều dài 1 viên gạch với tổng số gạch hàng ngang, lấy kết quả chia 2 để đều 2 bên.
* margin\_top: bằng ⅛ chiều dài màn hình.

1. **dropItem(int x, int y) (Tú):**

* Tại 1 thời điểm sẽ có nhiều nhất là 2 vật phẩm rơi ra khi phá gạch. Nếu Item1 đang là null sẽ có tỉ lệ rơi ra Item1, tương tự với Item2. Hàm sẽ sinh ngẫu nhiên 1 số nguyên dương từ 1 đến 18, nếu số đó chia hết cho 3 tương đương có 6 loại vật phẩm. 2 đối số x, y là tọa độ của vật phẩm khi được tạo ra: Hàm khởi tạo của Item là Item(int x, int y, int num) (với num là số nguyên được sinh ngẫu nhiên trước đó).

1. **createItem() (Tú):**

* Nếu Item1 khác null thì cho rơi xuống (phương thức move() của đối tượng Item). Nếu chạm vào thanh đỡ bóng (Paddle) thì gọi hàm touchItem() để áp dụng hiệu ứng của vật phẩm đồng thời đặt lại Item1 là null, còn khi Item1 rơi quá chiều dài của màn hình thì xem như bị bỏ qua và đặt lại là null. Tương tự với Item2.

1. **touchItem(Item item) (Tú):**

* Có 1 cấu trúc rẽ nhánh “switch” với đối số là thuộc tính “num” của đối tượng Item:
* num = 3: Thanh đỡ (Paddle) sẽ dài ra, sau đó gọi hàm setDefaultPaddle() để trở lại độ dài mặc định.
* num = 6: Thanh đỡ (Paddle) sẽ ngắn lại, sau đó gọi hàm setDefaultPaddle() để trở lại độ dài mặc định.
* num = 9: Quả bóng (Ball) to hơn, đồng thời di chuyển chậm lại, sau đó gọi hàm setDefaultBallSize() để trả lại các giá trị thuộc tính mặc định.
* num = 12: Quả bóng (Ball) nhỏ hơn, đồng thời di chuyển nhanh hơn, sau đó gọi hàm setDefaultBallSize() để trả lại các giá trị thuộc tính mặc định.
* num = 15: Người chơi được cộng thêm 1 mạng.
* num = 18: Biến item\_status đang giá trị 0 sẽ được gán bằng 1, khi va chạm với gạch bóng sẽ tiếp tục đi thẳng thay vì bị bật lại. Sau đó, gọi hàm setDefaultBall() để đặt lại giá trị của biến item\_status bằng 0.

1. **setDefaultPaddle(), setDefaultBallSize(), setDefaultBallSize() (Tú):**

* Tạo 1 nhiệm vụ TimerTask để làm các nhiệm vụ nêu trên. Nếu các luồng Timer như defaultPaddle, defaultBallSize, defaultBall đang là null, sẽ khởi tạo nó (new) và lập lịch (phương thức schedule() của lớp Timer) thực hiện nhiệm vụ trong vòng 5 giây. Nếu các luồng đang chạy (khác null), dùng phương thức cancel() để hủy lập lịch của luồng, sau đó làm tương tự như khi luồng bằng null.

1. **savePerformance() (Minh):**

* Là hàm dùng để kết nối với database, tạo câu lệnh query gọi Store Procedure lưu, cập nhật lại thông tin người chơi.

1. **addRankingTable() (Minh):**

* Tạo bảng xếp hạng người chơi và hiện thông tin 10 người gần nhất.

1. **checkCollision() (Tuấn Anh):**

* Đây là hàm kiểm tra va chạm cho bóng và xử lý các sự kiện xảy ra khi có va chạm
* Hàm gồm có:

+ Kiểm tra va chạm giữa bóng và paddle. Nếu có va chạm bóng sẽ đổi hướng theo trục Y và được đặt lại vị trí ngay trên paddle (nhằm đảm bảo bóng không bị dính vào paddle hay va chạm nhiều lần với paddle)

+ Kiểm tra va chạm giữa bóng và gạch. Nếu có va chạm gạch sẽ bị phá hủy và gọi hàm “dropItem()” để sinh ra vật phẩm. Đồng thời người chơi sẽ nhận điểm và cấp nhật biến “flagEmptyBrick”. Để kiểm tra va chạm với gạch, ta cần:

* intersects(int prevX, int prevY, int newX, int newY, Rectangle brick): Hàm này tạo một hình chữ nhật mô phỏng đường đi của bóng từ vị trí trước đến vị trí mới và kiểm tra xem nó có giao với gạch hay không. Giúp phát hiện va chạm giữa bóng và gạch trong khoảng thời gian di chuyển giữa hai khung hình, ngay cả khi bóng di chuyển nhanh và có thể vượt qua gạch trong một lần cập nhật vị trí.
* handlePreciseBrickCollision(Rectangle brick): Đây là hàm xử lý va chạm chính xác giữa bóng. Nếu va chạm ở cạnh trên hoặc dưới, bóng sẽ đổi hướng theo trục Y. Nếu va chạm ở cạnh trái hoặc phải, bóng sẽ đổi hướng theo trục X.
* Kiểm tra khi toàn bộ gạch đã bị phá hết hết (“flagEmptyBrick” bằng 0), trò chơi sẽ kết thúc, lưu thông tin người chơi và hiển thị bảng xếp hạng.
* Kiểm tra va chạm giữa bóng và tường, đảm bảo bóng sẽ đổi hướng khi chạm biên.
* Nếu bóng rơi xuống dưới màn hình (khi người chơi "mất mạng"), trừ một mạng của người chơi. Nếu hết mạng, trò chơi sẽ kết thúc; nếu còn mạng, gọi hàm resetBallAndPaddle() để chuẩn bị lại bóng và paddle cho lần chơi tiếp theo.
* resetBallAndPaddle(): hàm này đặt lại vị trí của bóng khi người chơi mất mạng; bóng được đặt ở giữa và ngay trên paddle để chuẩn bị cho lượt chơi tiếp theo.

**Package DatabaseConfig:**

1. **Class ConnectionConfig (Minh):**

Dùng để chứa mã cấu hình với database SQL Server

Gồm các bước sau:

**B1. Tải thư viện jdbc:**

**<!-- https://mvnrepository.com/artifact/com.microsoft.sqlserver/mssql-jdbc -->**

**<dependency>**

**<groupId>com.microsoft.sqlserver</groupId>**

**<artifactId>mssql-jdbc</artifactId>**

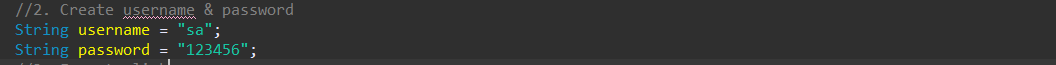
**<version>12.8.1.jre11</version>**

**</dependency>**

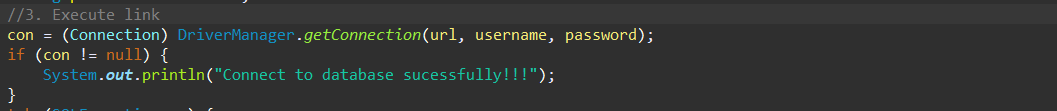
**B2. Tạo kết nối:**

Tạo url kết nối với database:

* API: jdbc.
* Loại database: sql server.
* Máy chủ: máy cá nhân
* Port: 1433
* Database name: BreakOutBall

**B3. Tạo tài khoản mật khẩu.**

**B4. Thực thi câu lệnh.**

****

\*\*Link tải database:

[BreakOutBallGame/BreakOutBall.sqlatmain·MinhKhongCau/BreakOutBallGame](https://github.com/MinhKhongCau/BreakOutBallGame/blob/main/BreakOutBall.sql)

Cấu hình SQL Server:

Bật đăng nhập bằng SQL Server Authentication

* SQL Server Property -> Security -> SQL Server Authentication

Bật TCP/ IP

* SQL Server Configuration Manager: Server network configuration -> TCP/IP -> enable

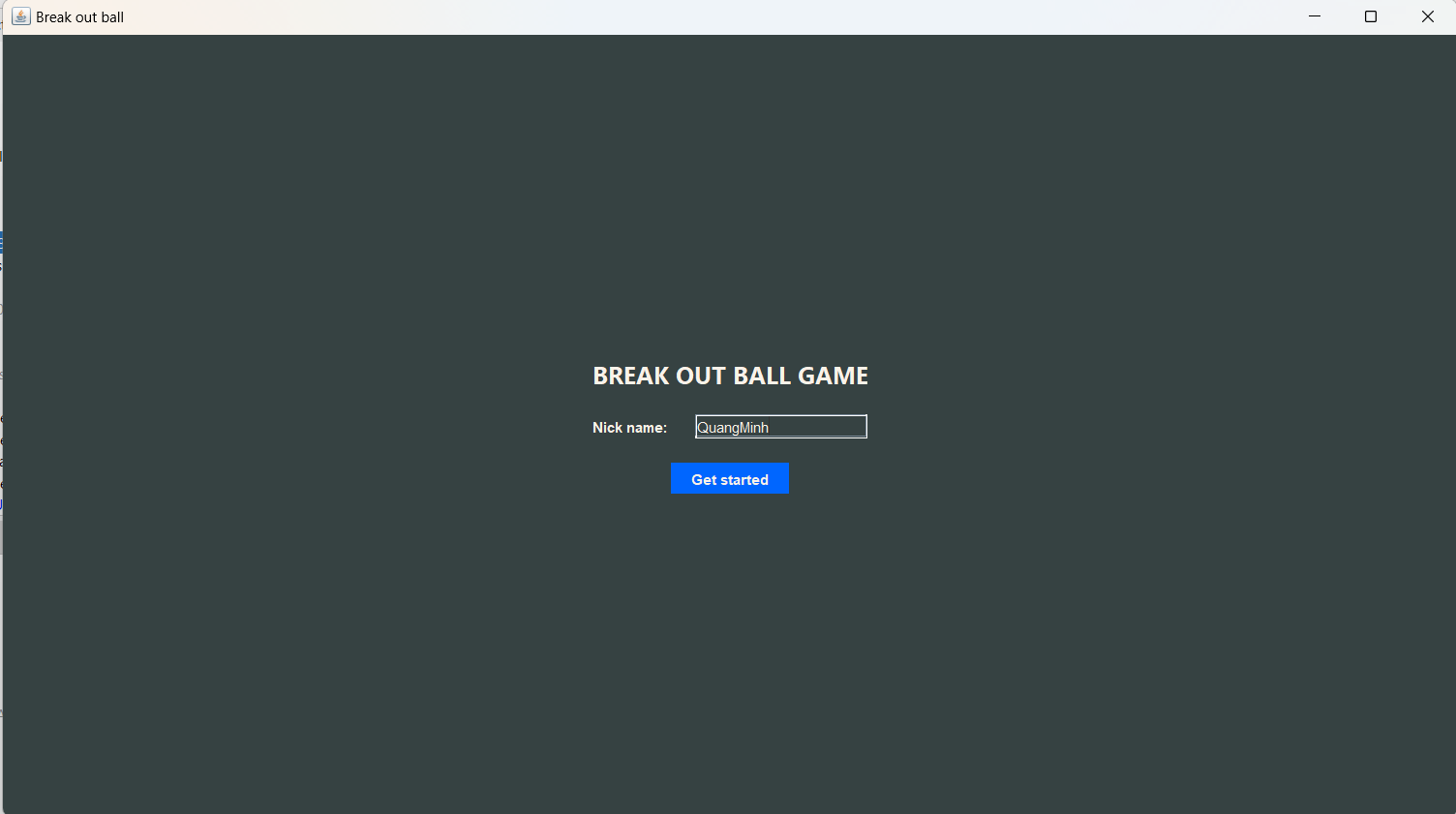
Bật login bằng tài khoản sa

* Login -> sa -> Status -> Login -> Enable.

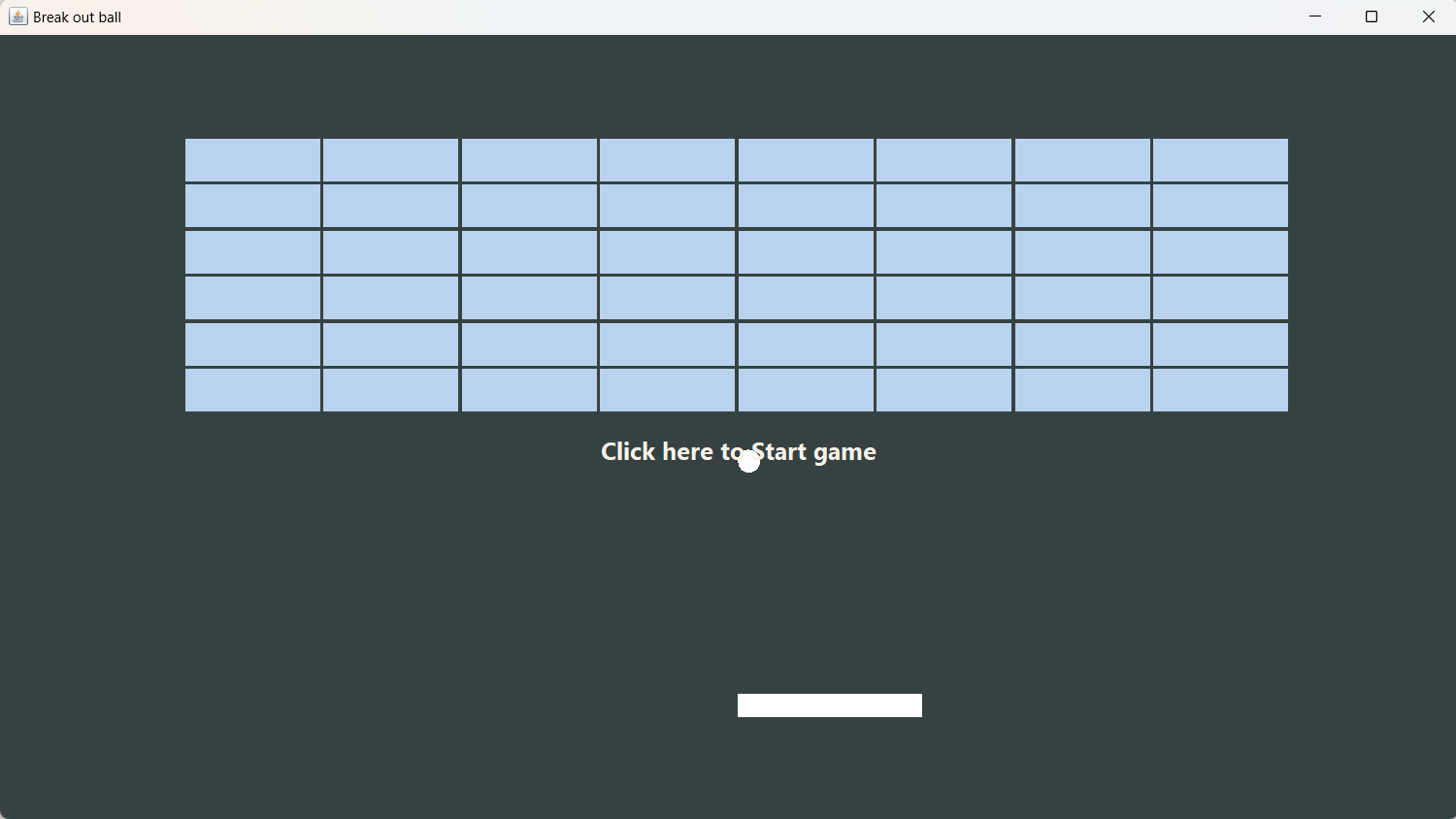
1. **Package SubGame (Minh):**

* Chứa các class dùng để mô tả thêm phần UI cho game bên cạnh các thành phần chính.
* Class PrepareGame: chứa màn hình chờ trước khi chơi
* Class InfoPanel: chứa 2 phần trên dưới theo dõi thông tin người chơi và tiến trình game.

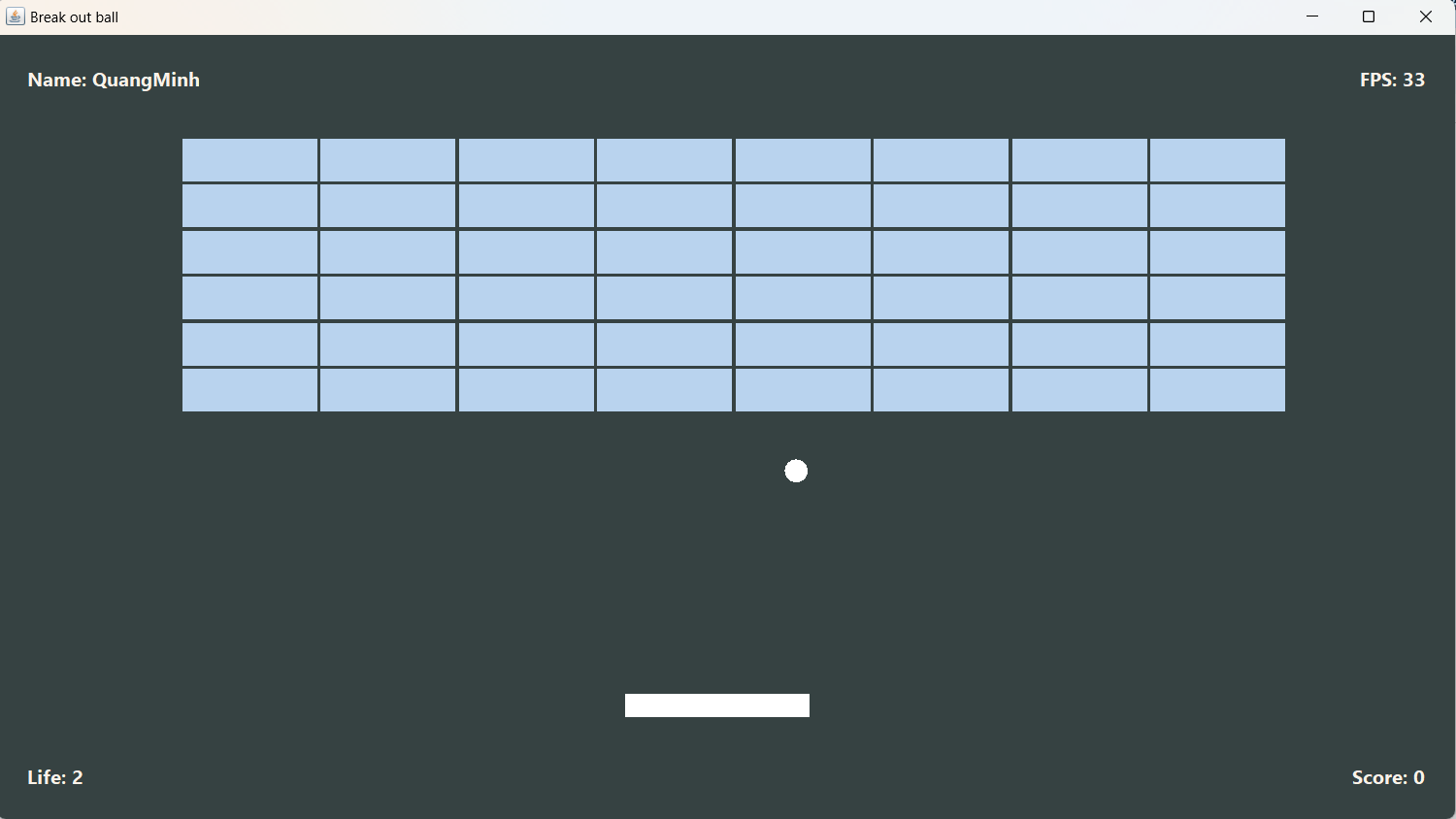
1. **Một vài hình ảnh của trò chơi:**
2. **Giao diện đăng nhập:**

****

1. **Giao diện chưa vào game:**

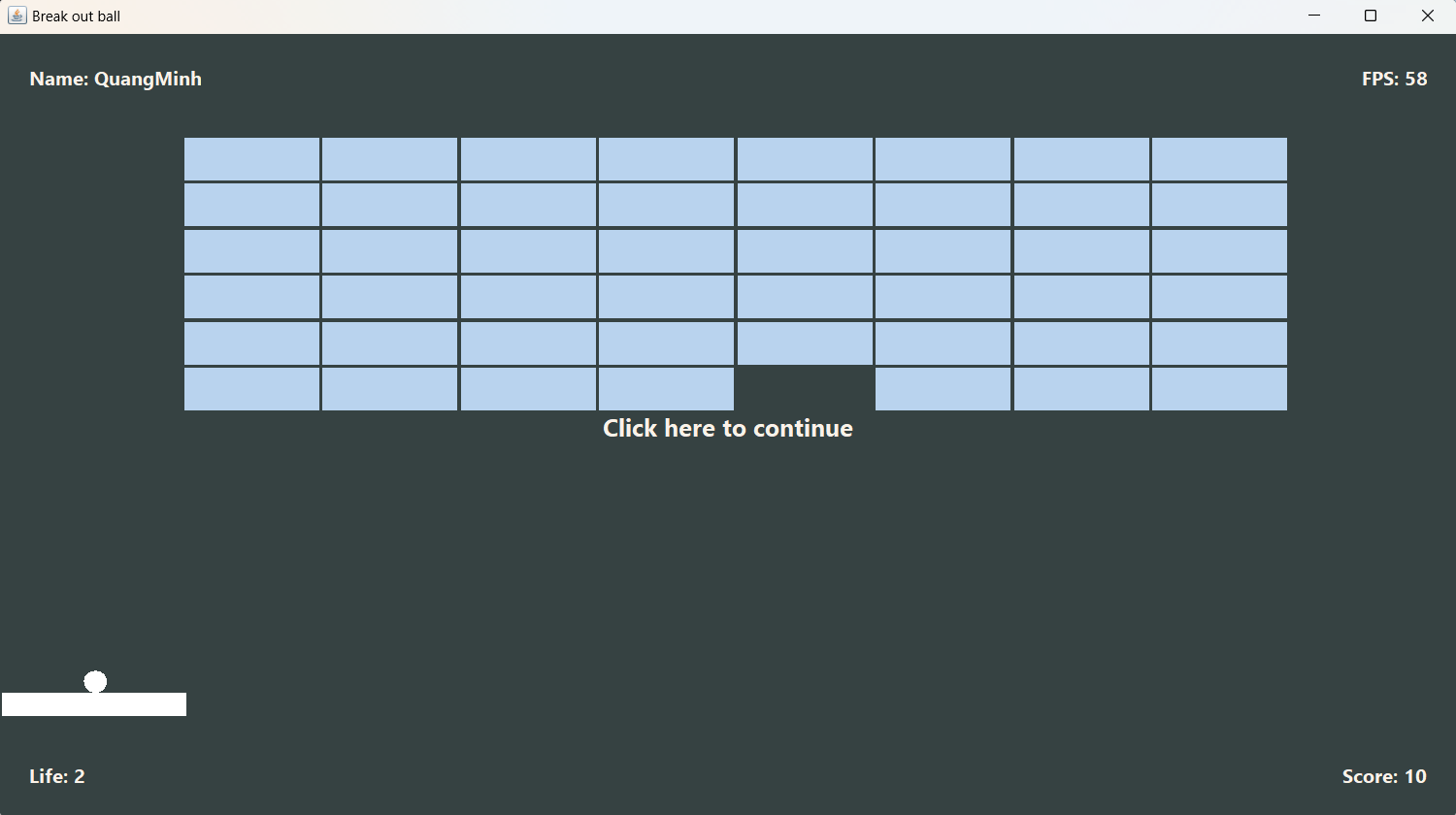
****

1. **Giao diện khi chạy game:**

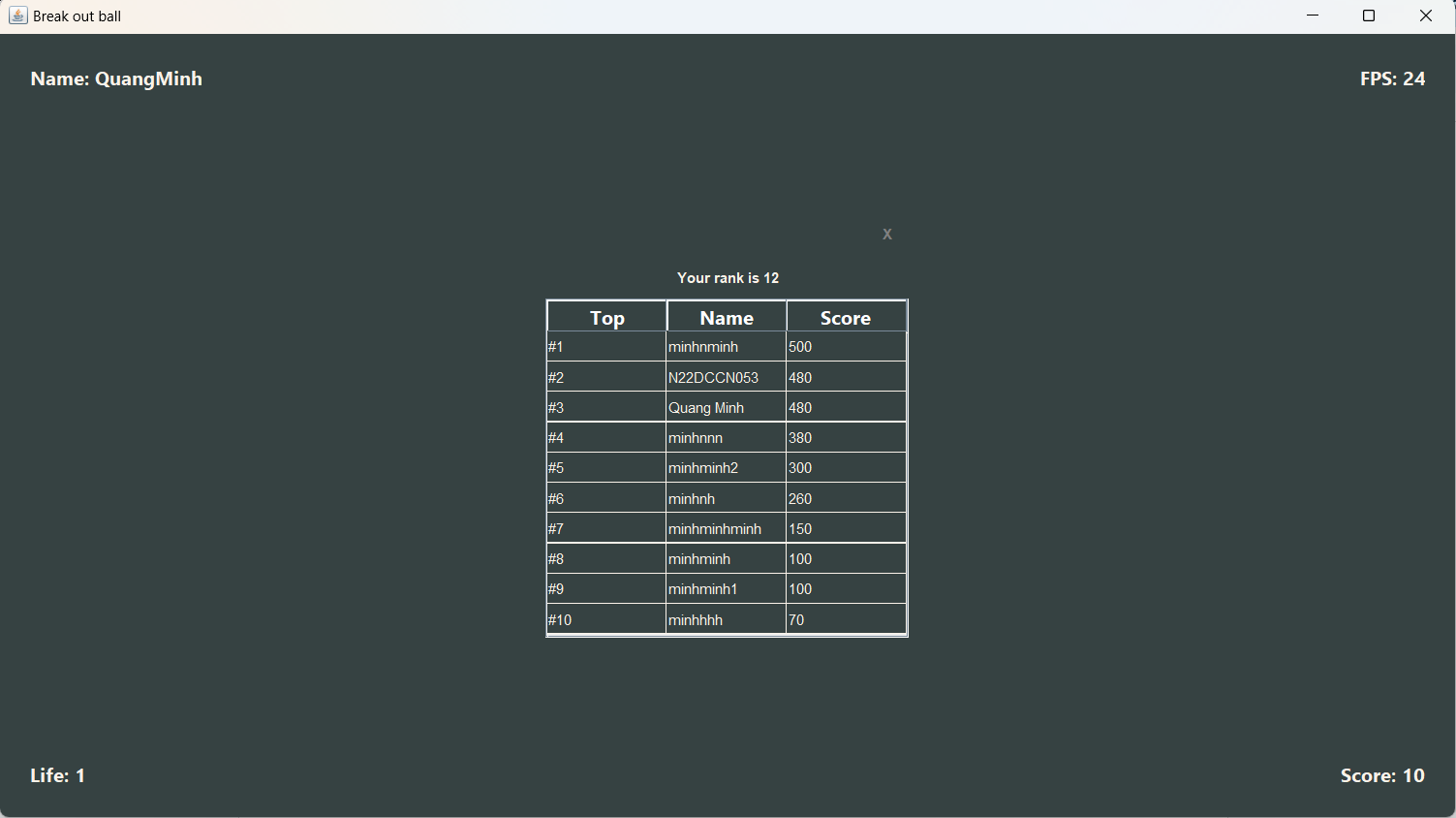
****

* **Gồm các thành phần:**
* **Tên người chơi, theo dõi FPS.**
* **Giao diện chính**
* **Số mạng, điểm.**

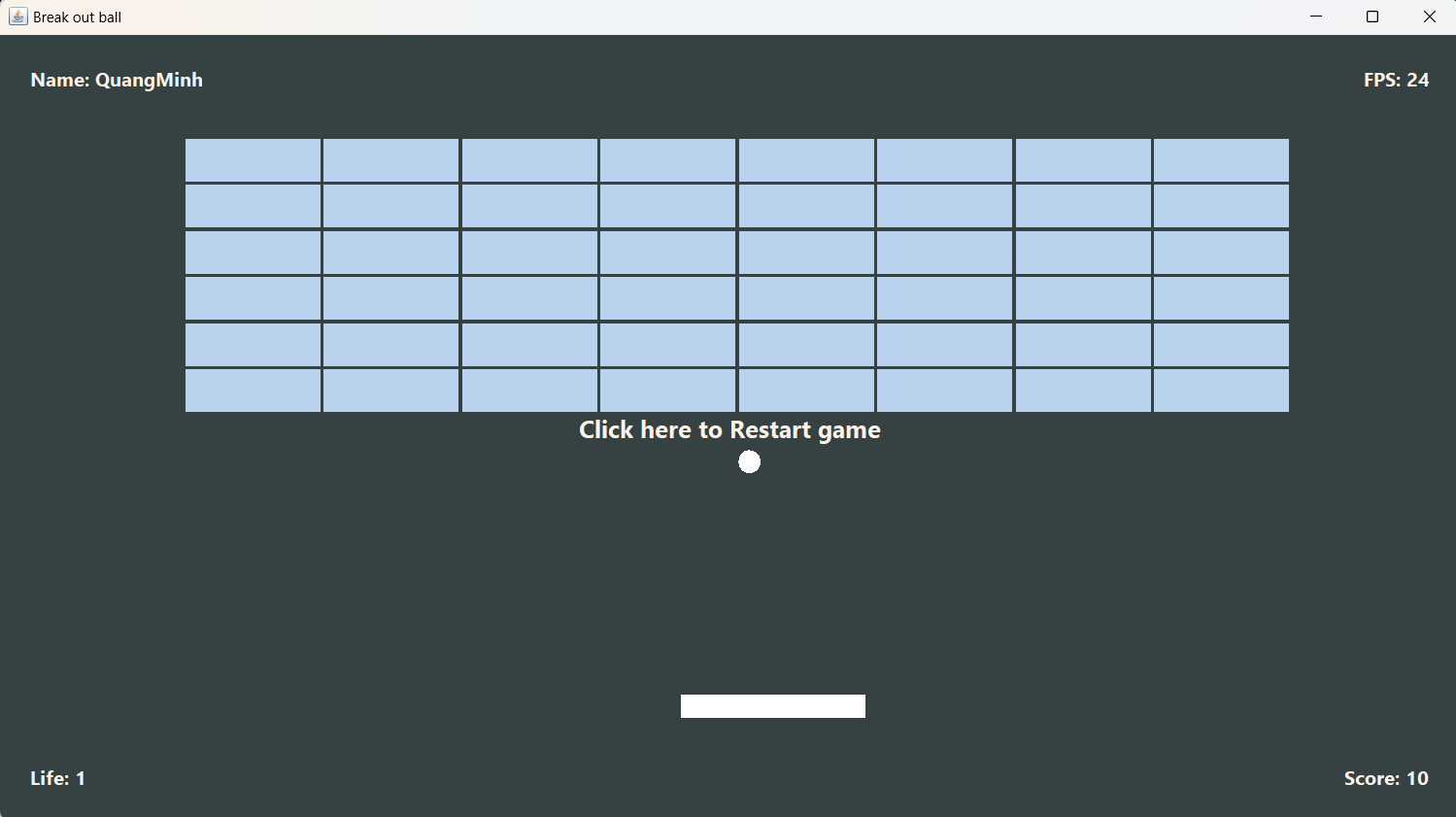
1. **Giao diện khi bóng rớt.**

****

1. **Bảng xếp hạng.**

****

1. **Chơi lại.**

****

1. **Tổng kết:**

* **Thông qua việc làm game đã giúp chúng em có thể hiểu hơn về ngôn ngữ java, sự mạnh mẽ của nó trong việc phát triển game hay phần mềm.**
* **Áp dụng được cách tính delta time để theo dõi tiến trình game.**

**[Hiểu về Delta Time khi làm Game](https://viblo.asia/p/hieu-ve-delta-time-khi-lam-game-Qbq5Q6WJKD8)**

* **Link Github game:**

[**MinhKhongCau/BreakOutBallGame**](https://github.com/MinhKhongCau/BreakOutBallGame)