TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo học phần: Lập trình hướng đối tượng

Mô phỏng game Tetris

Nhóm sinh viên thực hiện: 05

Giảng viên hướng dẫn: TS. Nguyễn Minh Hải

Thành phố Hồ Chí Minh, tháng 11 năm 2025

TRƯỜNG ĐẠI HỌC SÀI GÒN

KHOA CÔNG NGHỆ THÔNG TIN

Báo cáo học phần: Lập trình hướng đối tượng

Mô phỏng game Tetris

Thành viên: Nguyễn Mai Nam - 3124411173

Lê Phú Hiếu - 3124411090

Hoàng Phú Thịnh - 3124411291

Nguyễn Ngọc Tú - 3124411332

Thành phố Hồ Chí Minh, tháng 11 năm 2025

LỜI CAM ĐOAN

Chúng em xin cam đoan rằng tiểu luận môn Lập trình hướng đối tượng (OOP) với chủ đề "Thiết kế game Tetris" là sản phẩm nghiên cứu và thực hiện của toàn bộ thành viên trong nhóm. Toàn bộ nội dung, ý tưởng, thiết kế lớp (classes), mô hình quan hệ đối tượng, thuật toán xử lý va chạm, logic rơi khối, và các tài liệu minh họa trong tiểu luận đều do chúng em thảo luận, triển khai và kiểm nghiệm kỹ lưỡng.

Chúng em đã tham khảo, trích dẫn đầy đủ các nguồn tài liệu, bài báo, và tài nguyên lập trình đáng tin cậy để đảm bảo tính chính xác và khoa học cho các phần: phân tích yêu cầu, thiết kế UML (class diagrams, sequence diagrams), cài đặt các lớp chính (Block, Board, GameController, Renderer...), quản lý trạng thái trò chơi, và xử lý sự kiện người dùng. Mọi nguồn tham khảo đều được ghi rõ trong phần tài liệu tham khảo.

Chúng em nhận thức rõ trách nhiệm về nội dung tiểu luận; nếu có sai sót hoặc thiếu sót trong quá trình thực hiện, nhóm sẽ chủ động rà soát và chỉnh sửa. Hy vọng tiểu luận này sẽ góp phần làm rõ các khái niệm OOP trong bối cảnh thiết kế trò chơi, đồng thời cung cấp một hướng tiếp cận thực tế và có thể mở rộng cho việc phát triển game Tetris.

MỤC LỤC

DANH MỤC CÁC KÝ HIỆU VÀ CHỮ VIẾT TẮT

STT	Ký hiệu và chữ viết tắt	Tiếng Anh	Tiếng Việt
1	OOP	Object-Oriented Programming	Lập trình hướng đối tượng
2	GUI	Graphical User Interface	Giao diện đồ họa
3	JSON	JavaScript Object Notation	Định dạng dữ liệu JSON
4	JFrame	Java Frame	Cửa sổ chính trong Java Swing
5	UML	Unified Modeling Language	Ngôn ngữ mô hình hóa thống nhất
6	AWT	Abstract Window Toolkit	Bộ công cụ giao diện trừu tượng
7	UI	User Interface	Giao diện người dùng
8	JLabel	Java Label	Nhãn hiển thị trong Java Swing
9	2D	Two-Dimensional	Không gian 2 chiều
10	UX	User Experience	Trải nghiệm người dùng

DANH MỤC CÁC BẢNG

DANH MỤC CÁC HÌNH

LỜI MỞ ĐẦU

Trò chơi Tetris là một trong những trò chơi điện tử kinh điển, ra đời từ những năm 1980, với lối chơi đơn giản nhưng mang tính logic và thử thách cao. Trong trò chơi này, các khối gạch có hình dạng khác nhau (gọi là *Tetromino*) sẽ rơi xuống từ phía trên màn hình, và người chơi cần sắp xếp sao cho chúng tạo thành các hàng ngang hoàn chỉnh. Khi một hàng được lấp đầy, nó sẽ biến mất và người chơi được cộng điểm. Trò chơi kết thúc khi các khối gạch chồng lên đến đỉnh màn hình.

Mục tiêu dự án

Đề tài "Mô phỏng trò chơi xếp gạch (Tetris)" được thực hiện nhằm mục tiêu vận dụng kiến thức về lập trình hướng đối tượng (OOP) và xử lý giao diện đồ họa (GUI) để xây dựng một ứng dụng mô phỏng hoạt động của trò chơi Tetris. Thông qua đề tài này, sinh viên có cơ hội củng cố các kỹ năng về quản lý đối tượng, lập trình sự kiện, và xử lý đồ họa trong môi trường Java Swing/AWT.

Sản phẩm cuối cùng là một trò chơi Tetris hoàn chỉnh, có thể chạy độc lập, bao gồm các tính năng cơ bản như: tạo khối ngẫu nhiên, di chuyển, xoay khối, kiểm tra va chạm, tính điểm và kết thúc trò chơi. Đề tài không chỉ giúp rèn luyện kỹ năng lập trình mà còn góp phần nâng cao tư duy logic và khả năng thiết kế phần mềm theo hướng mô-đun.

Lý do chọn đề tài

Nhóm chọn đề tài mô phỏng game Tetris vì đây là trò chơi đơn giản nhưng chứa nhiều cơ chế quan trọng như xử lý va chạm, xoay khối, xóa hàng và quản lý lưới. Điều này giúp nhóm rèn luyện tư duy thuật toán, lập trình hướng đối tượng và xây dựng giao diện với Java Swing. Đồng thời, Tetris dễ thực hiện, dễ kiểm thử và phù hợp cho một bài tập lớn cần sản phẩm hoàn chỉnh.

Phạm vi dự án

Để bài tập lớn hoàn thành đúng tiến độ và đạt chất lượng, phạm vi được giới hạn như sau:

Mô phỏng trò chơi Tetris gồm toàn bộ cơ chế chơi (sinh Tetromino, rơi, xoay, va chạm), quản lý lưới chơi (grid), phát hiện và xóa hàng đầy, tính điểm, tăng độ khó theo cấp độ và kiểm tra kết thúc trò chơi.

Giao diện được thiết kế bằng Java Swing, gồm các thành phần chính như màn hình menu, màn hình chơi, khu vực hiển thị điểm số, level và khối kế tiếp. Các thao tác của người chơi (phím điều khiển) được xử lý trực tiếp trong giao diện.

Ứng dụng được phát triển dưới dạng ứng dụng Desktop độc lập, đảm bảo chạy ổn định trên các môi trường Windows hoặc các hệ điều hành hỗ trợ chạy ứng dụng Java.

Công cụ và ngôn ngữ lập trình

Dự án sử dụng các công cụ và ngôn ngữ phổ biến để đảm bảo tính khả thi và hiệu quả:

+ Ngôn ngữ lập trình: Java

+ IDE: Visual Studio Code, IntelliJ IDEA, Eclipse

+ Thư viện / công cụ hỗ trợ: Swing (giao diện), (tùy chọn) thư viện âm thanh nhẹ để chơi hiệu ứng; có thể dùng file để lưu highscore.

Cấu trúc báo cáo

- Báo cáo được trình bày chi tiết theo các phần sau:

+ **Phần 1.** Phân tích và mô hình hóa đối tượng

+ **Phần 2.** Thiết kế hệ thống

+ **Phần 3.** Phân tích và cài đặt hệ thống chi tiết

+ **Phần 4.** Giao diện

Phân công nhiệm vụ

Nhóm trưởng: Hoàng Phú Thịnh

STT	Họ và tên	MSSV	Khối lượng công việc (%)	Nhiệm vụ
1	Hoàng Phú Thịnh	3124411291	25	Kiểm thử + thiết kế
2	Lê Phú Hiếu	3124411090	25	Lập trình + thiết kế
3	Nguyễn Mai Nam	3124411173	25	Lập trình + thiết kế
4	Nguyễn Ngọc Tú	3124411332	25	Báo cáo + kiểm thử + thiết kế

PHẦN 1. PHÂN TÍCH & MÔ HÌNH HÓA ĐỐI TƯỢNG

1.1. Yêu cầu kĩ thuật tối thiểu

Bảng 1.1. Tiêu chí đánh giá

Tiêu chí	Điểm	Mô tả
Có ít nhất 15 lớp (class)	1đ	Mỗi lớp phải thể hiện vai trò cụ thể, đặt tên chuẩn
Có hàm thiết lập hàm khởi tạo (constructor) cho các lớp	1đ	Thể hiện khởi tạo dữ liệu và quan hệ lớp
Kế thừa hợp lý (có class cơ sở + class con)	1đ	Dùng từ khóa extends hoặc inherit đúng ngữ nghĩa
Đa hình (phương thức ghi đè/overriding, overload)	1đ	Thực thi đúng nguyên lý OOP
Có lớp quản lý danh sách đối tượng (array/list)	2đ	Có chức năng nhập, xuất, thêm, xóa, sửa, tìm kiếm, thống kê
Đọc và ghi dữ liệu lên file (text hoặc JSON)	1đ	Ghi lại và khôi phục dữ liệu
Có thuộc tính và phương thức static	1đ	Dùng hợp lý, ví dụ đếm số lượng đối tượng
Có lớp trừu tượng và hàm trừu tượng (abstract)	1đ	Mô hình hóa hành vi chung
Có interface và class thực thi interface	1đ	Dùng để chuẩn hóa hành vi giữa các lớp
Tính sáng tạo, mở rộng thêm chức năng	1đ	Ví dụ: giao diện GUI, báo cáo, thống kê biểu đồ

1.2. Phân tích

1.2.1. Mục tiêu trò chơi

- Xếp các khối gạch sao cho tạo thành hàng ngang liền mạch.

- Mỗi hàng hoàn thành sẽ bị xóa và người chơi được cộng điểm.

- Duy trì trò chơi càng lâu càng tốt mà không để các khối chạm đỉnh.

- Khi điểm số tăng, tốc độ rơi của khối gạch cũng tăng, giúp trò chơi trở nên khó hơn.

1.2.2. Cấu trúc và thành phần chính

1.2.2.1. Logic xử lý trò chơi

- Sinh ngẫu nhiên các khối gạch (Tetromino).

- Xử lý va chạm giữa khối gạch và lưới chơi (Grid).

- Phát hiện hàng đầy, xóa hàng và cập nhật điểm số.

- Tăng tốc độ rơi theo cấp độ (level).

- Kiểm tra điều kiện kết thúc trò chơi.

1.2.2.2. Giao diện hiển thị (GUI)

- Hiển thị khung lưới và các khối gạch hiện tại.

- Hiển thị điểm số, cấp độ và khối kế tiếp.

- Cập nhật đồ họa theo thời gian thực khi người chơi thao tác. Dựa trên thư viện Java Swing/AWT để vẽ đối tượng.

1.2.2.3. Xử lý điều khiển (Control)

- Nhận sự kiện bàn phím từ người chơi (xoay, di chuyển trái/phải, thả nhanh).

- Cập nhật vị trí hoặc trạng thái của khối gạch tương ứng.

- Đảm bảo thao tác diễn ra trơn tru, không xung đột với logic rơi tự động của game.

1.2.2.4. Chương trình chính (Main)

- Khởi tạo cửa sổ trò chơi (JFrame).

- Tạo đối tượng GamePanel và bắt đầu vòng lặp trò chơi.

- Quản lý các tiến trình chính như cập nhật logic, vẽ lại giao diện và xử lý sự kiện.

1.2.3. Mô hình hướng đối tượng

- Trò chơi được thiết kế theo mô hình lập trình hướng đối tượng (OOP) với các lớp chính:

+ **Game**: Quản lý tiến trình, điểm số, tốc độ và khối hiện tại.

+ **Grid**: Lưu trữ ma trận các ô, kiểm tra hàng đầy và va chạm.

+ **Tetromino** (lớp trừu tượng): Đại diện cho khối gạch chung.

+ **IBlock, OBlock, TBlock, LBlock, JBlock, SBlock, ZBlock: Kế thừa từ Tetromino**, thể hiện từng hình dạng.

+ **GamePanel**: Quản lý giao diện và nhận sự kiện từ người chơi.

+ **SoundManager** (tùy chọn): Quản lý hiệu ứng âm thanh.

+ **Main**: Khởi tạo và chạy chương trình.

1.3. Tóm tắt phần 1

Trò chơi Tetris được xây dựng với mục tiêu sắp xếp các khối Tetromino rơi xuống để tạo thành những hàng ngang liền mạch. Khi một hàng được lấp đầy, hệ thống sẽ tự động xóa hàng đó và cộng điểm cho người chơi, đồng thời tiếp tục tạo ra không gian mới cho các khối tiếp theo. Người chơi cần duy trì trò chơi càng lâu càng tốt bằng cách điều khiển các khối sao cho không để chúng chất chồng lên đến đỉnh màn hình. Khi điểm số tăng, tốc độ rơi của các khối cũng được tăng lên nhằm tạo thêm thử thách và giúp gameplay trở nên hấp dẫn hơn. Hệ thống của game bao gồm nhiều thành phần hoạt động kết hợp với nhau. Phần logic chịu trách nhiệm sinh ngẫu nhiên các Tetromino, xử lý va chạm với lưới và các khối đã rơi, kiểm tra các hàng đầy để xóa và tính điểm, đồng thời xác định khi nào trò chơi kết thúc.

PHẦN 2. THIẾT KẾ HỆ THỐNG

2.1. Mục tiêu thiết kế

- Trình bày ngắn gọn mục tiêu của hệ thống:

+ Mô phỏng lại trò chơi Tetris cổ điển.

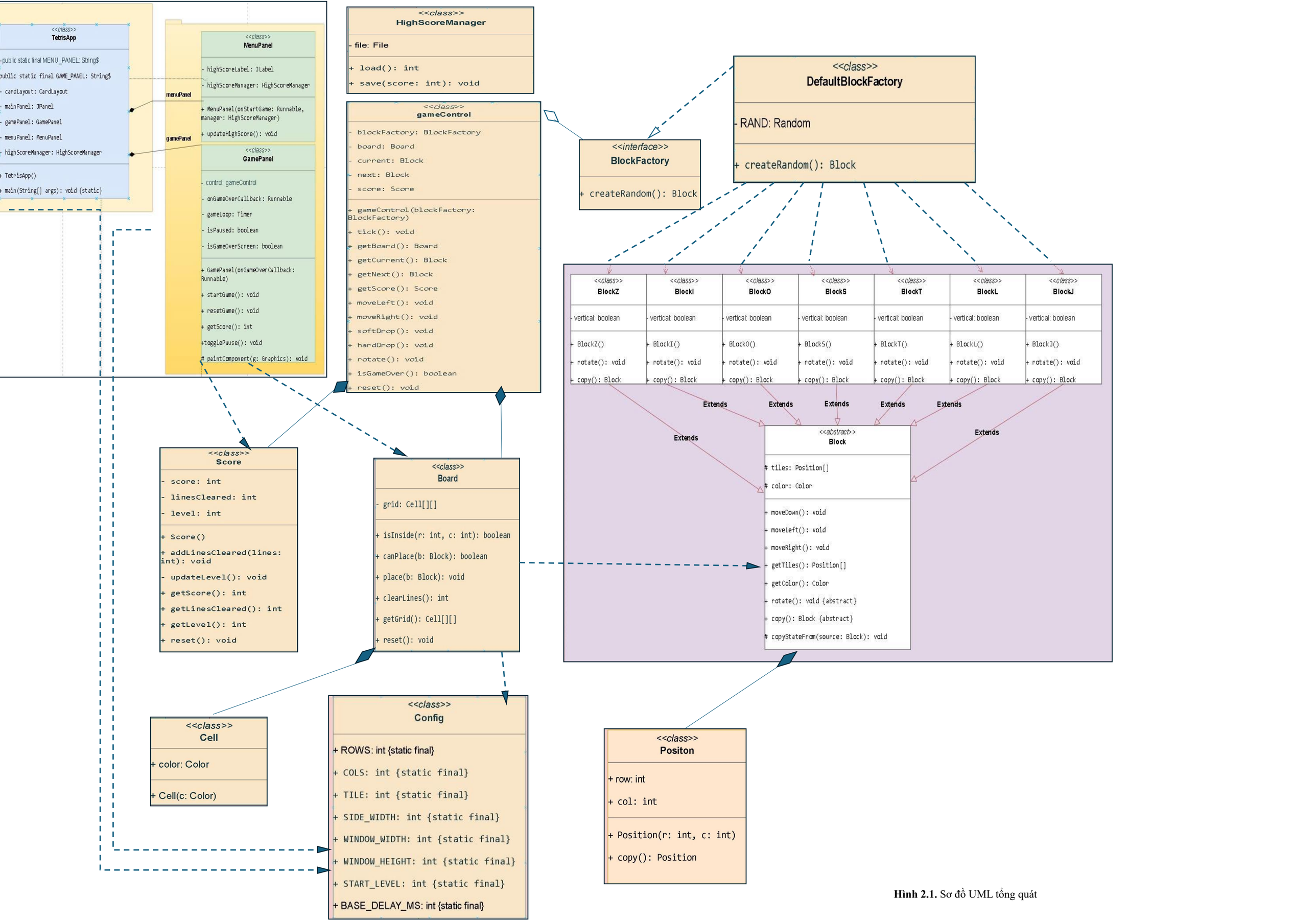
+ Cung cấp giao diện trực quan, dễ điều khiển.

+ Xử lý logic rơi khối, va chạm, xóa hàng, điểm số và tốc độ tăng dần.

2.2. Kiến trúc tổng thể

Hệ thống trò chơi Tetris được xây dựng với kiến trúc phân tách rõ ràng giữa logic cốt lõi và giao diện người dùng nhằm đảm bảo tính ổn định, dễ bảo trì và khả năng mở rộng. Phần logic Core chịu trách nhiệm xử lý toàn bộ cơ chế của trò chơi, bao gồm việc sinh các khối Tetromino, điều khiển rơi, kiểm tra va chạm, xóa hàng, tính điểm, quản lý tốc độ và kiểm tra điều kiện kết thúc. Mỗi khối được biểu diễn bằng lớp Block cùng các lớp con, lưu trữ hình dạng, màu sắc và vị trí hiện tại, đồng thời hỗ trợ các phương thức xoay và di chuyển theo thao tác người chơi. Board quản lý toàn bộ lưới chơi 20x10, kiểm tra vị trí hợp lệ, đặt khối khi chạm đáy hoặc va chạm với các khối khác, tự động xóa hàng đầy và trả về số hàng để tính điểm. BlockFactory và DefaultBlockFactory chịu trách nhiệm sinh các khối mới theo chuẩn Tetris hoặc ngẫu nhiên, giúp GameControl không cần quan tâm đến chi tiết khởi tạo từng loại khối.

Phần giao diện UI được xây dựng bằng Java Swing/AWT, gồm MenuPanel hiển thị các lựa chọn bắt đầu game, xem điểm cao và thoát ứng dụng, cùng GamePanel hiển thị bảng chơi, khối hiện tại, khối tiếp theo và điểm số theo thời gian thực. TetrisApp đóng vai trò cửa sổ chính và trung tâm điều phối, quản lý MenuPanel, GamePanel và HighScoreManager, đồng thời điều khiển luồng chuyển đổi giữa trạng thái menu, đang chơi và kết thúc game. Kiến trúc này đảm bảo toàn bộ luồng dữ liệu và sự kiện được xử lý mượt mà, logic và giao diện hoạt động tách biệt nhưng liên kết chặt chẽ.



Hình 2.1. Sơ đồ UML tổng quát

2.2.1. Core

Phần Core là thành phần trung tâm và quan trọng nhất của hệ thống, đảm nhiệm toàn bộ logic hoạt động của trò chơi Tetris. Đây là nơi xử lý tất cả các cơ chế cốt lõi như sinh các khối Tetromino, quản lý vị trí và trạng thái của khối, kiểm tra va chạm với lưới chơi, xử lý xóa các hàng đầy, tính toán điểm số, tăng tốc độ rơi theo cấp độ và kiểm tra điều kiện kết thúc trò chơi. Core được thiết kế hoàn toàn độc lập với giao diện hiển thị, nghĩa là toàn bộ các lớp và phương thức trong Core chỉ tập trung vào logic và trạng thái của trò chơi mà không quan tâm đến việc vẽ hoặc thao tác người dùng. Điều này giúp cho hệ thống dễ bảo trì, mở rộng, và có thể thay đổi giao diện mà không ảnh hưởng đến cách thức vận hành của game.

2.2.1.1. Lớp Block (Khối gạch)

- Chức năng:

    + Đại diện cho một khối Tetris (I, O, T, L, J, S, Z).

    + Mỗi khối được biểu diễn bằng ma trận 4x4 chứa các ô có hoặc không có giá trị.

- Nhiệm vụ chính:

    + Xác định hình dạng và màu sắc của khối.

    + Cho phép xoay khối theo chiều kim đồng hồ.

    + Lưu trữ vị trí hiện tại của khối trên bảng chơi.

- Thuộc tính tiêu biểu:

    + int x, y: tọa độ hiện tại của khối.

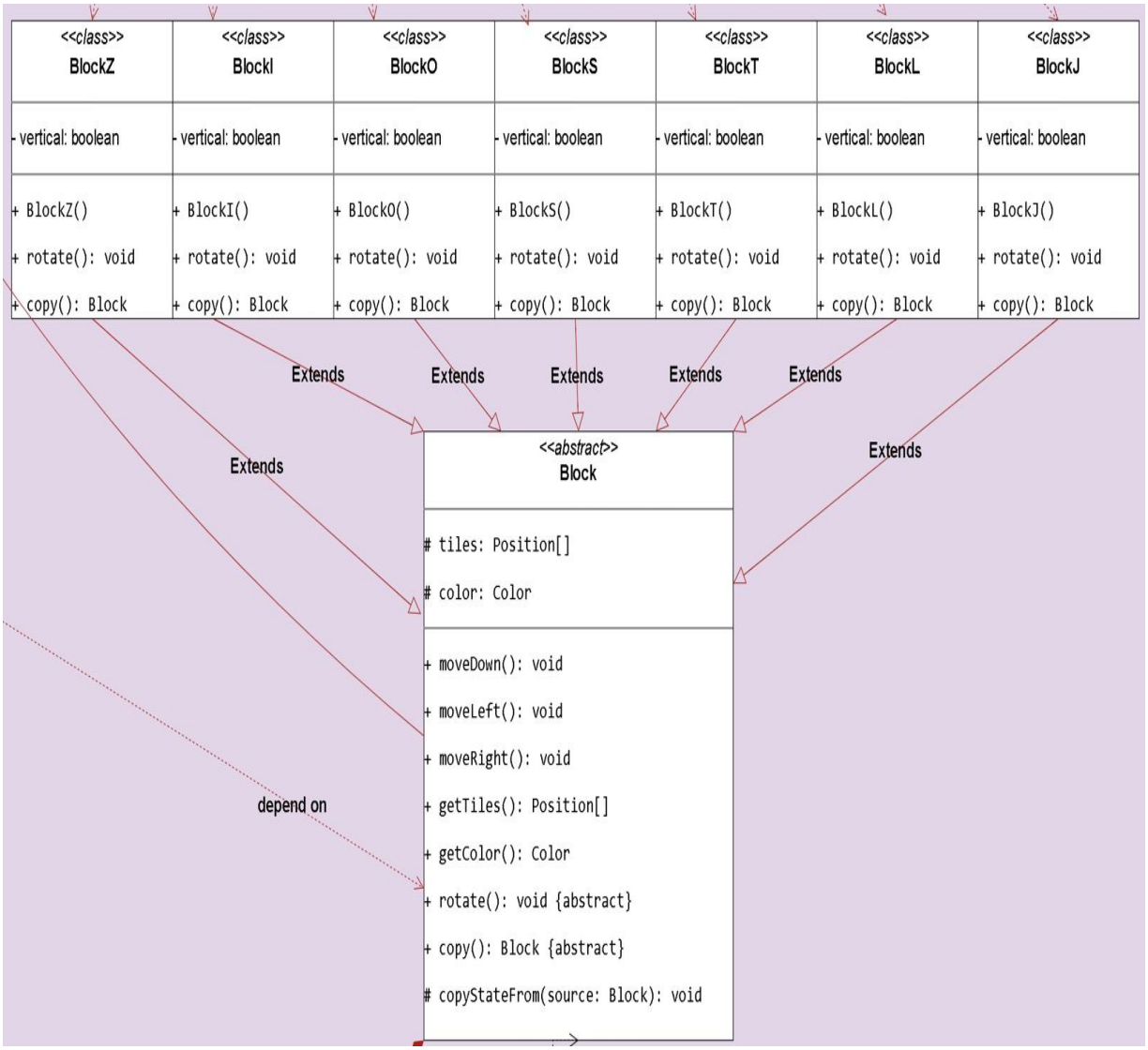
    + Color color: màu sắc của khối.

- Phương thức chính:

    + rotate(): xoay khối 90 độ.

    + moveLeft(), moveRight(), moveDown(): di chuyển khối.

Hình 2.1 dưới đây minh họa sơ đồ UML của lớp đại diện cho khối Tetris, mô tả các thuộc tính, phương thức và mối quan hệ trong hệ thống



Hình 2.2. Bảng vẽ UML lớp Block

2.2.1.2. Lớp Board (Bảng chơi)

- Chức năng:

+ Quản lý toàn bộ không gian chơi của game Tetris.

+ Đây là nơi kiểm tra va chạm, lưu trạng thái của các khối đã rơi và xử lý khi hàng được lấp đầy.

- Nhiệm vụ chính:

+ Lưu trạng thái của từng ô trong bảng.

+ Kiểm tra khối có thể di chuyển hoặc xoay hợp lệ không.

+ Xóa hàng đầy và cập nhật điểm.

- Thuộc tính tiêu biểu:

+ int grid[HEIGHT][WIDTH]: ma trận lưu ô đã chiếm chỗ.

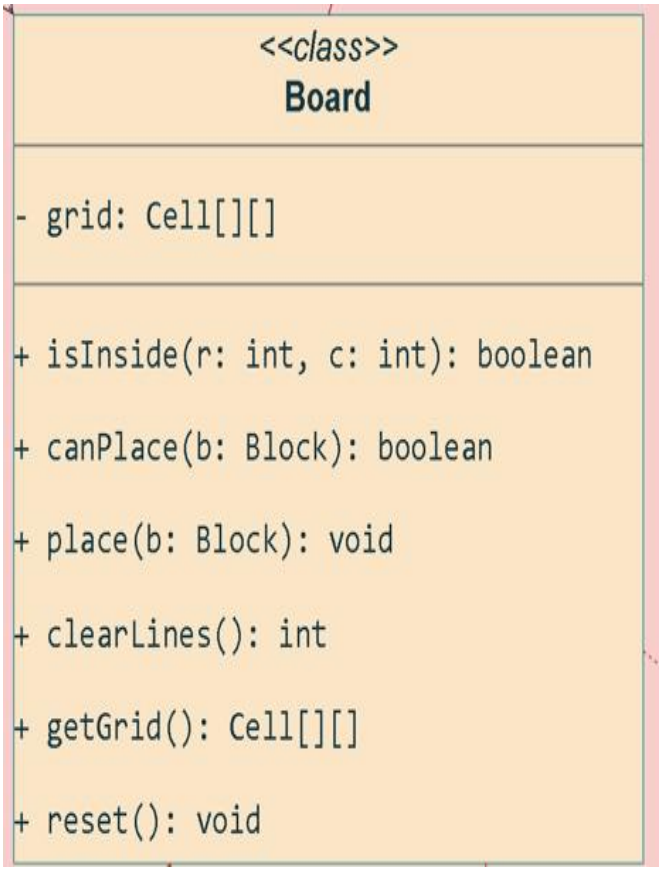
- Phương thức chính:

+ canPlace(Block b): kiểm tra khối có thể di chuyển được không.

+ getGrid(Block b): dùng để vẽ hoặc kiểm tra.

+ clearLines(): xóa hàng đầy và tăng điểm.

Sơ đồ UML ở hình 2.2 dưới đây minh họa lớp *Board* – thành phần chịu trách nhiệm quản lý toàn bộ bảng chơi, bao gồm trạng thái các ô, xử lý va chạm, và cập nhật điểm số trong game Tetris:



Hình 2.3. Bảng vẽ UML lớp Board

2.2.1.3. Lớp BlockFactory (Bộ sinh khối)

- Chức năng:

+ Sinh ngẫu nhiên các khối Tetris mới.

- Nhiệm vụ chính:

+ Chọn ngẫu nhiên một trong bảy hình khối cơ bản.

+ Khởi tạo vị trí ban đầu của khối trên bảng.

- Phương thức chính:

+ generateBlock(): trả về một đối tượng Block mới với hình dạng ngẫu nhiên.

2.2.1.4. Lớp GameController (Bộ điều khiển trò chơi)

- Chức năng:

+ Là trung tâm điều phối giữa các thành phần Core.

+ Quản lý toàn bộ trạng thái trò chơi: bắt đầu, tạm dừng, kết thúc, điểm và tốc độ rơi.

- Nhiệm vụ chính:

+ Gọi các hàm cập nhật vị trí khối theo thời gian.

+ Xử lý sự kiện từ người chơi (di chuyển, xoay, thả nhanh).

+ Kiểm tra điều kiện thua cuộc (Game Over).

- Phương thức chính:

+ startGame(): khởi tạo trò chơi.

+ update(): cập nhật khối và bảng mỗi khung hình.

+ handleInput(KeyEvent e): nhận tín hiệu điều khiển.

+ gameOver(): dừng trò chơi khi bảng bị đầy.

2.2.2. UI (Java Swing/AWT)

- Chức năng:

Giao diện người dùng (UI) là phần hiển thị và tương tác chính của người chơi với game Tetris. Thành phần này chịu trách nhiệm vẽ bảng chơi, khối Tetris, điểm số và xử lý các sự kiện từ bàn phím để điều khiển trò chơi. Toàn bộ được xây dựng dựa trên thư viện Java Swing/AWT nhằm đảm bảo khả năng hiển thị linh hoạt và dễ mở rộng.

- Nhiệm vụ chính:

+ Hiển thị bảng chơi (grid) và các khối Tetris đang rơi.

+ Cập nhật giao diện khi khối di chuyển, xoay hoặc khi hàng được xóa.

+ Hiển thị thông tin điểm số, cấp độ, và khối tiếp theo (Next Block).

+ Lắng nghe và xử lý các thao tác bàn phím (trái, phải, xuống, xoay).

+ Kết nối với lớp Board để cập nhật trạng thái và vẽ lại nội dung.

2.2.2.1.TetrisApp

- Chức năng:

+ Là cửa sổ (JFrame) chính của trò chơi.

+ Quản lý và chuyển đổi giữa MenuPanel và GamePanel.

+ Khởi tạo toàn bộ giao diện người dùng khi ứng dụng bắt đầu chạy.

- Thuộc tính:

+ menuPanel: MenuPanel – panel hiển thị menu chính.

+ gamePanel: GamePanel – panel hiển thị màn chơi.

+ highScoreManager: HighScoreManager – quản lý điểm cao.

- Phương thức:

+ TetrisApp() – khởi tạo ứng dụng, thiết lập cửa sổ chính.

+ startGame(): void – chuyển từ menu sang màn chơi chính.

+ showMenu(): void – quay lại menu sau khi kết thúc trò chơi.

- Vai trò:

+ TetrisApp đóng vai trò điều phối trung tâm, chịu trách nhiệm tạo và hiển thị các panel, đồng thời xử lý luồng chuyển đổi giữa trạng thái menu và trò chơi.

2.2.2.2. MenuPanel

- Chức năng:

+ Hiển thị giao diện menu chính và điểm cao nhất.

+ Cho phép người chơi bắt đầu một trò chơi mới.

+ Tương tác với HighScoreManager để lấy và cập nhật điểm cao.

- Thuộc tính:

+ highScoreLabel: JLabel – hiển thị điểm cao nhất.

+ highScoreManager: HighScoreManager – quản lý dữ liệu điểm cao.

- Phương thức:

+ MenuPanel(onStartGame: Runnable, manager: HighScoreManager) – khởi tạo panel, nhận callback khi bắt đầu game.

+ updateHighScore(): void – cập nhật điểm cao sau khi trò chơi kết thúc.

- Vai trò:

+ Khi người chơi chọn “Start Game”, MenuPanel gọi callback onStartGame để chuyển sang GamePanel bắt đầu chơi.

2.2.2.3. GamePanel

- Chức năng:

+ Hiển thị khu vực chơi chính của Tetris.

+ Xử lý logic game: bắt đầu, reset, kết thúc, và cập nhật điểm.

+ Thông báo cho hệ thống khi trò chơi kết thúc để quay lại menu.

- Thuộc tính:

+ control: GameControl – đối tượng điều khiển logic game.

+ onGameOverCallback: Runnable – callback gọi khi trò chơi kết thúc.

+ gameLoop: Timer – bộ đếm thời gian cập nhật game liên tục.

- Phương thức:

+ GamePanel(onGameOverCallback: Runnable) – khởi tạo panel và liên kết callback.

+ startGame(): void – bắt đầu trò chơi mới.

+ resetGame(): void – đặt lại trạng thái trò chơi.

+ getScore(): int – trả về điểm hiện tại của người chơi.

+ paintComponent(g: Graphics): void – vẽ lại giao diện, bao gồm bảng chơi và các khối.

- Vai trò:

+ Trong suốt quá trình chơi, GamePanel chịu trách nhiệm hiển thị khối Tetris, xử lý va chạm, tính điểm và gọi callback khi game kết thúc để quay lại MenuPanel.

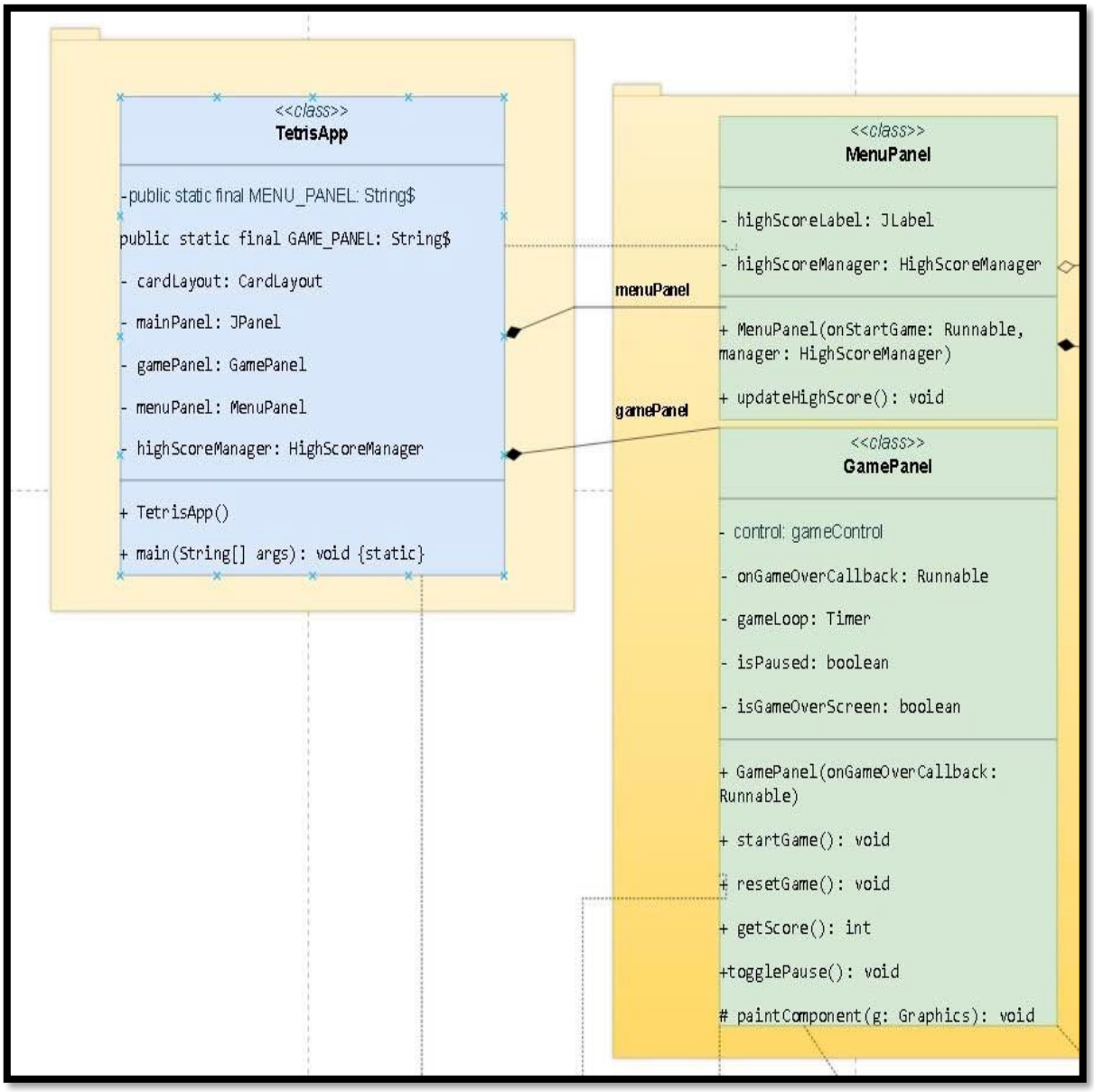
2.2.3. Mối quan hệ giữa các thành phần

TetrisApp là trung tâm điều phối giao diện chính của trò chơi Tetris, đóng vai trò quản lý cả MenuPanel và GamePanel, đồng thời điều khiển việc chuyển đổi giữa các màn hình và quản lý trạng thái trò chơi. Nó đảm bảo rằng người chơi luôn tương tác với giao diện phù hợp tùy theo trạng thái hiện tại: menu chính, đang chơi, hay game over.

Khi người chơi nhấn “Start Game” trên MenuPanel, panel này sẽ gửi một tín hiệu (callback) tới TetrisApp. TetrisApp nhận tín hiệu này, thực hiện ẩn MenuPanel và hiển thị GamePanel, từ đó khởi tạo một phiên chơi mới. Trong quá trình chơi, TetrisApp chịu trách nhiệm lắng nghe các sự kiện từ GamePanel, chẳng hạn như khi người chơi xếp hết các khối hoặc không còn nước đi hợp lệ.

Khi GamePanel báo "Game Over", TetrisApp sẽ ngay lập tức thực hiện các bước cần thiết: tạm dừng trò chơi, hiển thị lại MenuPanel để người chơi có thể chọn chơi lại hoặc thoát, đồng thời gọi HighScoreManager để cập nhật và lưu trữ điểm số cao mới nếu cần.

Nhờ vai trò trung tâm của TetrisApp, toàn bộ luồng điều phối giữa menu, game và điểm cao được thực hiện một cách trơn tru, giúp trải nghiệm người chơi liền mạch và trực quan. Hình 2.3 dưới đây minh họa sơ đồ UML lớp của trò chơi Tetris, thể hiện vai trò trung tâm của TetrisApp trong việc điều phối MenuPanel, GamePanel và HighScoreManager, cũng như các mối quan hệ và luồng tương tác giữa chúng.



Hình 2.3. Biểu đồ mối quan hệ giữa các lớp giao diện

2.3. Tóm tắt phần 2

Hệ thống được thiết kế với mục tiêu mô phỏng trò chơi Tetris cổ điển, cung cấp giao diện trực quan và xử lý đầy đủ các cơ chế như rơi khối, va chạm, xóa hàng, tính điểm và tăng tốc độ theo level. Kiến trúc tổng thể gồm hai phần chính: Core và UI. Core chịu trách nhiệm toàn bộ logic game, bao gồm các lớp như Block để mô tả hình dạng và thao tác xoay/di chuyển của khối; Board để quản lý lưới, kiểm tra va chạm và xóa hàng; BlockFactory để sinh ngẫu nhiên các khối mới; và GameController để điều phối cập nhật trạng thái trò chơi, xử lý input và kiểm tra Game Over. Tất cả lớp này hoạt động độc lập với giao diện để đảm bảo tính tái sử dụng và độ ổn định của logic.

Phần giao diện (UI) được xây dựng bằng Java Swing/AWT, đảm nhiệm việc hiển thị grid, khối đang rơi, điểm số, level, khối kế tiếp và nhận thao tác điều khiển từ người chơi. Ba thành phần chính gồm: TetrisApp (cửa sổ chính, quản lý MenuPanel và GamePanel), MenuPanel (menu chính, hiển thị điểm cao và nút bắt đầu game) và GamePanel (vùng chơi, cập nhật game, vẽ giao diện và thông báo khi game kết thúc). TetrisApp đóng vai trò trung tâm điều phối, chịu trách nhiệm chuyển đổi giữa menu và game, đồng thời kết nối với HighScoreManager để lưu điểm cao. Mối liên kết giữa các thành phần giúp game vận hành mượt mà, dễ mở rộng và đảm bảo trải nghiệm người dùng liền mạch.

PHẦN 3.CÀI ĐẶT VÀ MÔ TẢ CÁC CHỨC NĂNG CHÍNH

3.1. Core

3.1.1. BlockFactory

- Chức năng chi tiết:

+ Tạo các khối Tetris theo loại (I, J, L, O, S, T, Z).

+ Cho phép tạo khối ngẫu nhiên để sử dụng trong game.

+ Là nơi tập trung logic để khởi tạo khối với hình dạng chuẩn, tránh trùng lặp code trong GameControl.

- Mã giả:

```
class BlockFactory:
    method createBlock(type):
        // Tạo khối theo loại
        switch type:
            case I: return new IBlock()
            case J: return new JBlock()
            case L: return new LBlock()
            case O: return new OBlock()
            case S: return new SBlock()
            case T: return new TBlock()
            case Z: return new ZBlock()

    method createRandomBlock():
        // Chọn ngẫu nhiên một loại khối
        type = random choice from [I,J,L,O,S,T,Z]
        return createBlock(type)
```

3.1.2. Board.java

- Chức năng chi tiết:

+ Lưu trữ trạng thái bảng 2D (ROWS x COLUMNS) gồm các ô trống hoặc đầy.

+ Kiểm tra khối có thể di chuyển, xoay hay rơi xuống không.

+ Khi khối không thể rơi, đặt khối vào bảng.

+ Tự động xóa các hàng đầy và trả về số hàng xóa để tính điểm.

+ Quản lý va chạm giữa các khối và rìa bảng

- Mã giả:

```
class Board:
    attribute grid[ROWS][COLUMNS]  // true nếu ô đã đầy, false nếu trống

    method isValidPosition(block, position):
        // Kiểm tra từng ô của block tại vị trí position
        for each cell in block:
            if cell.row < 0 or cell.row >= ROWS:
                return false
            if cell.col < 0 or cell.col >= COLUMNS:
                return false
            if grid[cell.row][cell.col] is filled:
                return false
        return true

    method placeBlock(block, position):
        // Đặt block vào bảng
        for each cell in block:
            grid[cell.row][cell.col] = filled

    method clearFullRows():
        linesCleared = 0
        for row in 0..ROWS-1:
            if all cells in row are filled:
                remove row
                shift all rows above down by 1
                linesCleared += 1
        return linesCleared
```

3.1.3. Cell

- Chức năng chi tiết:

+ Đại diện một ô trên bảng, biết trống hay đầy.

+ Cung cấp phương thức fill() và clear() để cập nhật trạng thái khi khối rơi hoặc xóa hàng

- Mã giả:

```
class Cell:
    attribute isEmpty = true
    method fill():
        isEmpty = false
    method clear():
        isEmpty = true
```

3.1.4. Config

- Chức năng chi tiết:

+ Lưu tất cả hằng số cấu hình game, dễ chỉnh sửa mà không làm ảnh hưởng code chính.

+ Bao gồm kích thước bảng, tốc độ rơi khối, thang điểm.

- Mã giả:

```
class Config:
    constant ROWS = 20
    constant COLUMNS = 10
    constant INITIAL_SPEED = 500  // ms giữa mỗi bước rơi
    constant SCORE_TABLE = {1:100, 2:300, 3:500, 4:800}
```

3.1.5. DefaultBlockFactory

- Chức năng chi tiết:

+ Kế thừa BlockFactory.

+ Thực hiện tạo khối ngẫu nhiên theo chuẩn Tetris.

+ Giúp GameControl chỉ cần gọi phương thức tạo khối, không cần quan tâm chi tiết từng loại.

- Mã giả:

```
class DefaultBlockFactory extends BlockFactory:
    method createRandomBlock():
        type = random choice from [I,J,L,O,S,T,Z]
        return createBlock(type)
```

3.1.6. gameControl

- Chức năng chi tiết:

+ Điều khiển logic chính của game:

+ Quản lý bảng và khối hiện tại (currentBlock) và khối tiếp theo (nextBlock).

+ Xử lý di chuyển trái/phải, rơi nhanh, xoay khối.

+ Khi khối chạm đáy hoặc khối khác → đặt vào bảng, xóa hàng đầy và cập nhật điểm.

+ Kiểm tra game over khi khối mới không thể đặt vào bảng.

- Mã giả:

```
class GameControl:
    attribute board
    attribute currentBlock
    attribute nextBlock
    attribute score
    attribute blockFactory
    method startGame():
        board = new Board()
        currentBlock = blockFactory.createRandomBlock()
        nextBlock = blockFactory.createRandomBlock()
        loop gameTick() until gameOver

    method gameTick():
        if board.isValidPosition(currentBlock, currentBlock.position.down()):
            move currentBlock down
        else:
            board.placeBlock(currentBlock)
            lines = board.clearFullRows()
            score.update(lines)
            currentBlock = nextBlock
            nextBlock = blockFactory.createRandomBlock()
            if not board.isValidPosition(currentBlock, currentBlock.position):
                gameOver = true
```

```
    method moveLeft():
        if board.isValidPosition(currentBlock, currentBlock.position.left()):
            currentBlock.position.moveLeft()

    method moveRight():
        if board.isValidPosition(currentBlock, currentBlock.position.right()):
            currentBlock.position.moveRight()

    method rotate():
        currentBlock.rotate()
        if not board.isValidPosition(currentBlock, currentBlock.position):
            currentBlock.rotateBack()
```

3.1.7. HighScoreManager

- Chức năng chi tiết:

+ Lưu và quản lý danh sách điểm cao.

+ Ghi/đọc điểm từ file.

+ Thêm điểm mới, sắp xếp giảm dần, giữ top N điểm.

- Mã giả:

```
class HighScoreManager:
    attribute highScoresList

    method loadScores():
        read from file into highScoresList

    method saveScores():
        write highScoresList to file

    method addScore(name, score):
        insert score with name
        sort list descending
        keep top N scores
```

3.1.8. Position

- Chức năng chi tiết:

+ Đại diện vị trí khối trên bảng.

+ Cung cấp phương thức di chuyển trái, phải, xuống.

- Mã giả:

```
class Position:
    attribute row
    attribute column

    method moveDown():
        row += 1

    method moveLeft():
        column -= 1

    method moveRight():
        column += 1
```

3.1.9. Score

- Chức năng chi tiết:

+ Quản lý điểm của người chơi hiện tại.

+ Cập nhật dựa trên số hàng xóa được theo quy tắc Tetris.

- Mã giả:

```
class Score:
    attribute currentScore = 0

    method update(linesCleared):
        if linesCleared in SCORE_TABLE:
            currentScore += SCORE_TABLE[linesCleared]
```

3.2. UI

3.2.1. GamePanel

- Chức năng chi tiết:

+ Hiển thị bảng, khối hiện tại, khối tiếp theo, điểm.

+ Vẽ ô trống hoặc đầy dựa trên trạng thái Board.

- Mã giả:

```
class GamePanel:
    method paint(board, currentBlock, nextBlock, score):
        draw grid from board
        draw currentBlock at current position
        draw nextBlock in preview area
        draw current score
```

3.2.2. MenuPanel

- Chức năng chi tiết:

+ Hiển thị menu chính: Start Game, High Scores, Exit.

+ Xử lý input người dùng để điều hướng menu.

- Mã giả:

```
class MenuPanel:
    method displayMenu():
        show options: ["Start Game", "High Scores", "Exit"]
        handle user selection
```

3.3. Main

- Chức năng chi tiết:

+ Điểm khởi chạy của ứng dụng.

+ Khởi tạo menu và bắt đầu game khi chọn.

- Mã giả:

```
class TetrisApp:
    method main():
        menu = new MenuPanel()
        menu.displayMenu()
```

Bảng tóm tắt lớp (Bảng 3.1) tổng hợp chức năng chính, thuộc tính và các phương thức quan trọng của từng lớp, giúp hình dung rõ cấu trúc toàn hệ thống. Toàn bộ phần cài đặt thể hiện sự phân tách rạch ròi giữa logic lõi và giao diện, đảm bảo dự án dễ phát triển, bảo trì và mở rộng.

Bảng 3.1. Bảng tóm tắt lớp

Lớp	Chức năng chính	Thuộc tính	Phương thức
BlockFactory	Tạo các khối Tetris theo loại hoặc ngẫu nhiên		createBlock(type), createRandomBlock()
DefaultBlockFactory	Kế thừa BlockFactory, tạo khối ngẫu nhiên chuẩn		createRandomBlock()
Board	Quản lý bảng, kiểm tra va chạm, đặt khối, xóa hàng đầy	Grid [ROWS][COLUMNS]	isValidPosition (block, pos), placeBlock (block, pos), clearFullRows()
Cell	Đại diện một ô trên bảng (trống/đầy)	isEmpty	fill(), clear()
Config	Lưu cấu hình game	ROWS, COLUMNS, INITIAL_SPEED, SCORE_TABLE	

GameControl	Điều khiển logic game: di chuyển, xoay, rơi khối, xóa hàng, game over	board, currentBlock, nextBlock, score, blockFactory	startGame(), gameTick(), moveLeft(), moveRight(), rotate()
HighScoreManager	Quản lý điểm cao, đọc/ghi file, sắp xếp	highScoresList	loadScores(), saveScores(), addScore(name, score)
Position	Đại diện vị trí khối trên bảng	row, column	moveDown(), moveLeft(), moveRight()
Score	Quản lý điểm người chơi hiện tại	currentScore	
GamePanel	Giao diện hiển thị bảng, khối hiện tại và điểm		update(linesCleared)
MenuPanel	Giao diện menu chính, xử lý lựa chọn người dùng		paint(board, currentBlock, nextBlock, score)
TetrisApp	Entry point ứng dụng, khởi tạo menu và game		main()

3.4. Tóm tắt phần 3

Phần cài đặt của dự án mô phỏng game Tetris được xây dựng dựa trên hai nhóm chức năng chính gồm Core và giao diện UI. Trong Core, các lớp quan trọng được triển khai để xử lý toàn bộ logic của trò chơi. BlockFactory chịu trách nhiệm tạo các khối Tetris theo đúng 7 loại chuẩn và có thể sinh ngẫu nhiên khi cần. Board quản lý bảng 2D của game, kiểm tra va chạm, xác định khối có thể di chuyển hoặc xoay, đặt khối xuống bảng khi đã chạm đáy và thực hiện xóa hàng đầy để trả về số hàng cần tính điểm. Mỗi ô trên bảng được biểu diễn bởi lớp Cell, cho phép đánh dấu ô trống hoặc đầy. Các hằng số của trò chơi như kích thước bảng, tốc độ rơi hay bảng điểm đều được gom trong lớp Config để dễ chỉnh sửa và quản lý.

DefaultBlockFactory mở rộng BlockFactory bằng việc cung cấp cơ chế sinh khối ngẫu nhiên theo chuẩn Tetris, giúp GameControl không cần quan tâm đến chi tiết từng loại khối. GameControl là trung tâm điều khiển toàn bộ game, bao gồm quản lý khối hiện tại, khối tiếp theo, xử lý di chuyển trái/phải, xoay, rơi xuống, đặt khối, xóa hàng đầy, cập nhật điểm và kiểm tra điều kiện kết thúc trò chơi. Score quản lý và cập nhật điểm dựa trên số hàng xóa được, còn HighScoreManager đảm nhiệm việc lưu trữ, sắp xếp và ghi/đọc danh sách điểm cao từ file. Vị trí của khối được đại diện bằng lớp Position, hỗ trợ di chuyển theo các hướng.

Về UI, GamePanel đảm nhiệm việc hiển thị bảng chơi, vẽ khối hiện tại, khối tiếp theo và điểm số. MenuPanel cung cấp giao diện menu chính cho người chơi bắt đầu game, xem điểm cao hoặc thoát ứng dụng. Cuối cùng, TetrisApp đóng vai trò điểm khởi chạy của chương trình, tạo menu và điều phối chuyển sang giao diện chơi khi người dùng chọn bắt đầu.

PHẦN 4. KẾT QUẢ & GIAO DIỆN

4.1. Kết quả

Trò chơi mô phỏng lại đúng cách hoạt động kinh điển của Tetris, bao gồm việc tạo khối, điều khiển rơi, kiểm tra va chạm, xóa hàng, tính điểm và quản lý điểm cao. Game Tetris được phát triển với bảng chơi 20 hàng x 10 cột, các khối Tetris (I, J, L, O, S, T, Z) rơi từ trên xuống.

- Người chơi có thể điều khiển khối bằng các thao tác:

  + Di chuyển sang trái hoặc phải (phím mũi tên trái phải).

  + Xoay khối theo chiều kim đồng hồ (phím mũi tên hướng lên).

  + Thả nhanh khối xuống dưới để tăng tốc trò chơi (space).

- Khi khối chạm đáy bảng hoặc va chạm với khối đã đặt, nó sẽ được cố định vào bảng.

- Bảng tự động xóa các hàng đầy và cập nhật điểm theo số hàng xóa:

  + 1 hàng → +100 điểm

  + 2 hàng → +300 điểm

  + 3 hàng → +500 điểm

  + 4 hàng → +800 điểm

- Trò chơi kết thúc khi khối mới không thể đặt vào vị trí bắt đầu vì bảng đã đầy

→ Game Over.

- Điểm cao được quản lý bởi HighScoreManager, lưu lại danh sách điểm cao, sắp xếp giảm dần và hiển thị khi người chơi chọn.

- Người chơi có thể bắt đầu lại game hoặc thoát, tùy thuộc vào lựa chọn trong menu.

- Dự án Tetris đã hoàn thiện đầy đủ cả logic hoạt động lẫn giao diện người dùng, đảm bảo tính mượt mà trong thao tác và tính chính xác trong xử lý va chạm. Trò chơi vận hành ổn định, có khả năng lưu trữ điểm cao, hiển thị thông tin trực quan, đồng thời mang lại trải nghiệm quen thuộc nhưng hấp dẫn cho người chơi.

4.2. Giao diện

- Giao diện được thiết kế thành hai phần chính: Menu và Game Panel.

4.2.1. MenuPanel

- Hiển thị menu chính với các lựa chọn:

    + Start Game – bắt đầu trò chơi mới.

    + High Scores – xem danh sách điểm cao.

Người chơi điều hướng menu bằng phím hoặc chuột, menu nổi bật tùy chọn đang chọn như hình 4.1 dưới đây

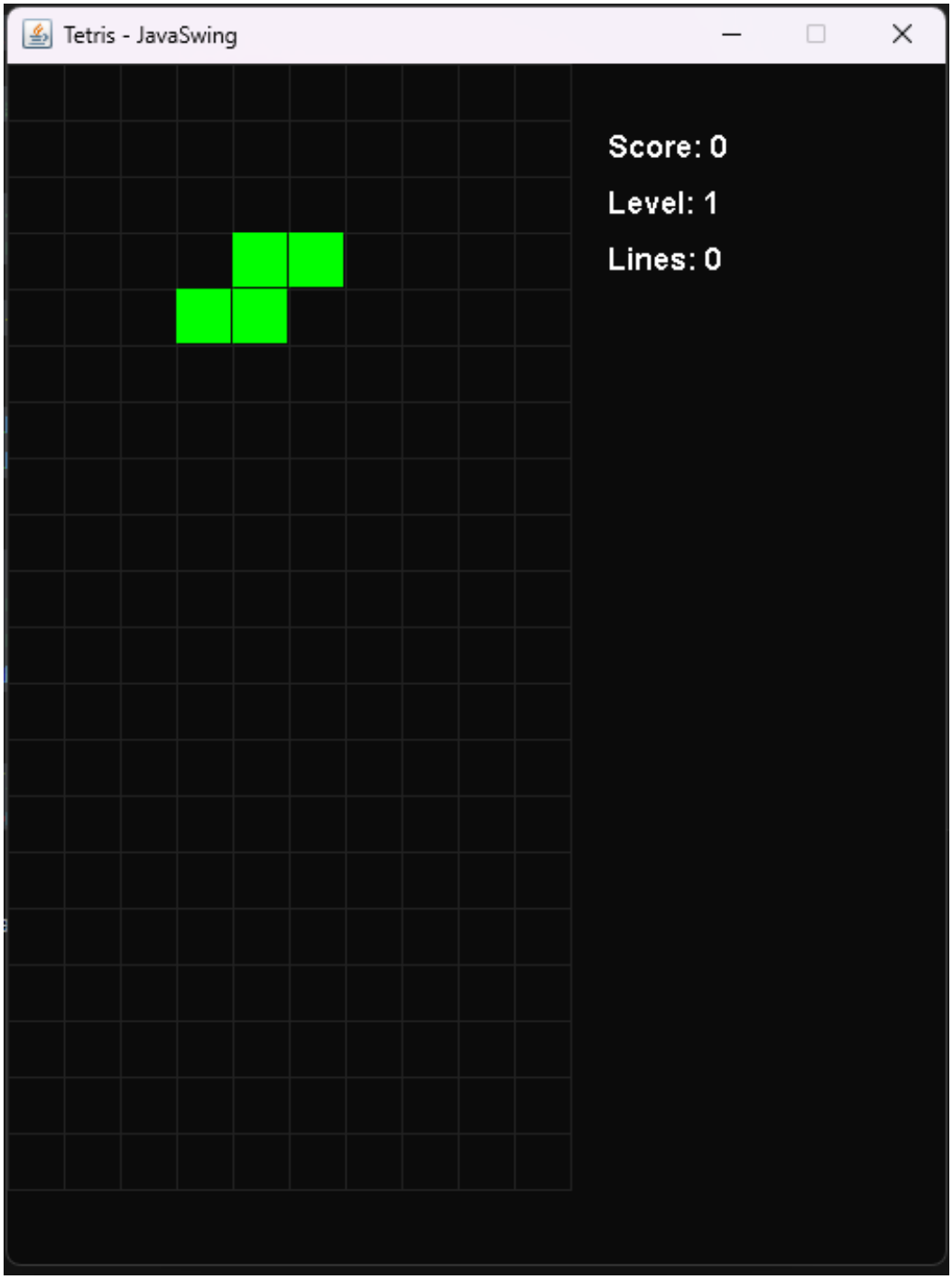
Hình 4.1. Giao diện menu bắt đầu trò chơi

4.2.2. GamePanel

- Hiển thị bảng chính gồm 20x10 ô:

+ Ô trống → chưa có khối.

+ Ô đầy → khối đã được cố định.

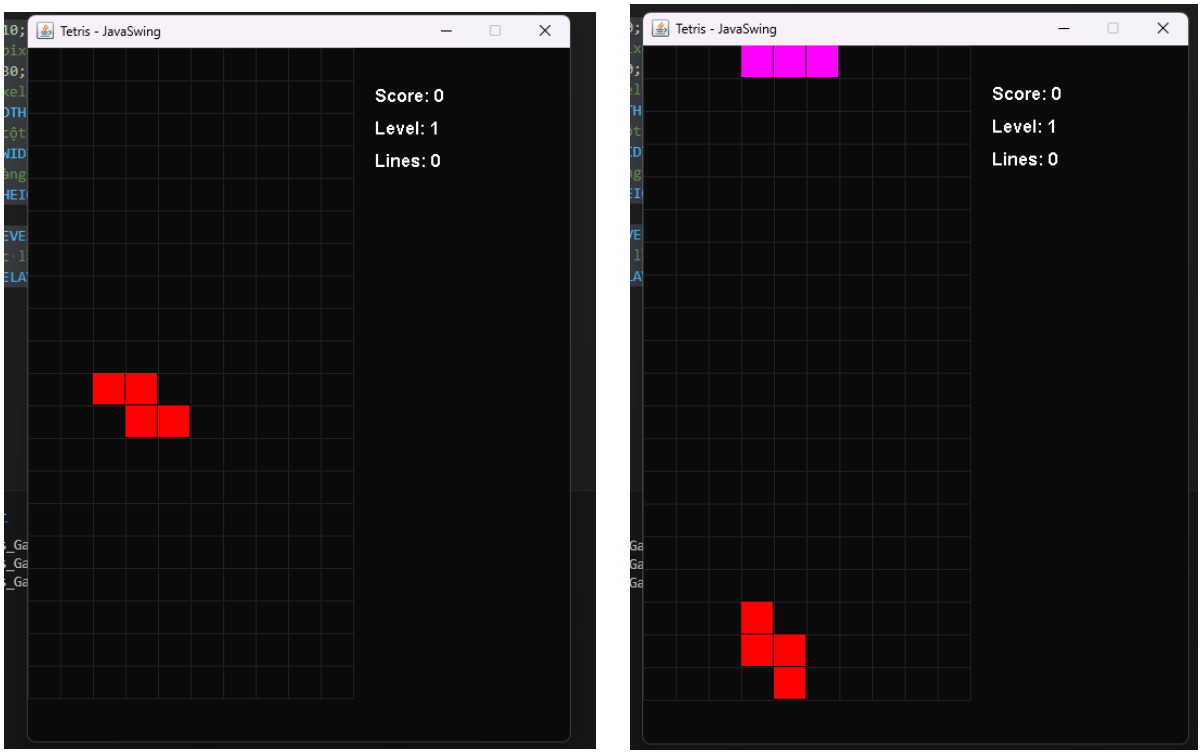


Hình 4.2. Giao diện khi vào GamePanel

- Khối hiện tại:

+ Di chuyển, xoay và rơi tự động.

+ Vị trí được cập nhật liên tục dựa trên GameControl.



Hình 4.3. Di chuyển và xoay Block

- Điểm hiện tại:

+ Cập nhật theo số hàng đã xóa, hiển thị trực tiếp trên màn hình.

+ Giao diện trực quan, dễ nhìn, giúp người chơi theo dõi trạng thái bảng và điểm số ngay cả khi khối đang rơi nhanh.

4.3. Tóm tắt phần 4

Kết quả và giao diện của game Tetris cho phép người chơi trải nghiệm logic rơi khối, di chuyển, xoay, xóa hàng và tính điểm theo chuẩn Tetris.

Menu và bảng hiển thị giúp người chơi dễ dàng bắt đầu, theo dõi trò chơi và quản lý điểm cao, đồng thời giao diện trực quan giúp trải nghiệm game mượt mà và rõ ràng.

Phần kết quả và giao diện cho thấy trò chơi Tetris đã được mô phỏng hoàn chỉnh, đúng theo cơ chế truyền thống: khối rơi từ trên xuống, người chơi có thể di chuyển trái/phải, xoay, thả nhanh và hệ thống tự xử lý va chạm, cố định khối, xóa hàng và tính điểm theo số hàng bị xóa. Trò chơi kết thúc khi không thể sinh khối mới, đồng thời điểm cao được lưu lại và sắp xếp thông qua HighScoreManager.

Giao diện gồm MenuPanel và GamePanel: menu cho phép bắt đầu game và xem điểm cao, trong khi GamePanel hiển thị bảng 20x10, khối hiện tại, khối tiếp theo và điểm đang được cập nhật theo thời gian thực. Hình ảnh minh họa cho thấy bố cục rõ ràng, các thao tác mượt mà và thông tin luôn hiển thị trực quan.

Tổng thể, trò chơi đã hoàn thiện đầy đủ cả mặt logic và giao diện, chạy ổn định, dễ sử dụng và mang lại trải nghiệm quen thuộc, đúng chuẩn của game Tetris cổ điển.

PHẦN 5. KẾT LUẬN & HƯỚNG PHÁT TRIỂN

5.1. Kết luận

Qua quá trình thực hiện đề tài "Mô phỏng trò chơi Tetris", nhóm đã vận dụng hiệu quả các kiến thức đã học trong môn Lập trình hướng đối tượng (OOP) để xây dựng một ứng dụng có tính thực tiễn, trực quan và sinh động.

- Đề tài giúp nhóm hiểu sâu hơn về:

+ Cách phân tích và mô hình hóa đối tượng trong một hệ thống trò chơi.

+ Việc tổ chức và quản lý các lớp (class) theo nguyên lý kế thừa, đa hình, trừu tượng, và đóng gói.

+ Ứng dụng Java GUI (Java Swing/AWT) để hiển thị và xử lý tương tác người dùng theo thời gian thực.

+ Tư duy thiết kế hướng mô-đun, giúp dễ dàng bảo trì, mở rộng và tái sử dụng mã nguồn.

+ Kết quả đạt được là một phiên bản Tetris hoàn chỉnh, có thể:

+ Sinh khối gạch ngẫu nhiên.

+ Di chuyển, xoay và rơi tự động theo thời gian.

+ Xử lý va chạm, xóa hàng đầy và cập nhật điểm số.

+ Có giao diện trực quan và hoạt động ổn định.

Thông qua quá trình làm việc nhóm, các thành viên không chỉ rèn luyện được kỹ năng phân chia công việc, quản lý tiến độ, trao đổi kỹ thuật và kiểm thử phần mềm, mà còn học được cách phối hợp hiệu quả giữa các vai trò trong dự án, cùng nhau giải quyết vấn đề và hỗ trợ lẫn nhau khi gặp khó khăn. Quá trình này giúp mỗi thành viên nâng cao khả năng giao tiếp, tinh thần trách nhiệm, cũng như tích lũy thêm nhiều kinh nghiệm thực tế trong việc phát triển và triển khai phần mềm, từ đó góp phần củng cố năng lực làm việc nhóm và kỹ năng lập trình chuyên nghiệp.

5.2. Hướng phát triển

Trong tương lai, nhóm mong muốn tiếp tục nâng cấp và mở rộng trò chơi với các tính năng mới, bao gồm:

- Chế độ chơi nâng cao: thêm nhiều cấp độ (level) khó hơn, tăng tốc độ rơi hoặc thêm các hiệu ứng đặc biệt.

- Lưu và tải lại trạng thái trò chơi (Save/Load Game).

- Bảng xếp hạng người chơi (Leaderboard) để lưu điểm cao.

- Âm thanh và nhạc nền sinh động hơn, sử dụng thư viện Java Swing/AWT Media hoặc thư viện âm thanh ngoài.

- Nâng cấp giao diện người dùng (UI/UX) với hiệu ứng đồ họa đẹp hơn, hỗ trợ theme tối/sáng.

- Phát triển phiên bản trực tuyến (Multiplayer) để người chơi có thể thi đấu với nhau qua mạng.

Nhóm tin rằng với nền tảng đã xây dựng, game Tetris hoàn toàn có thể trở thành một dự án mở rộng nhằm phục vụ việc học, giảng dạy và nghiên cứu về lập trình hướng đối tượng, xử lý đồ họa và lập trình game trong Java.

5.3. Tóm tắt phần 5

Qua quá trình thực hiện đề tài "Mô phỏng trò chơi Tetris", nhóm đã vận dụng hiệu quả kiến thức lập trình hướng đối tượng và Java GUI để xây dựng một trò chơi hoàn chỉnh, trực quan và ổn định. Dự án giúp nhóm hiểu rõ cách phân tích, mô hình hóa đối tượng, tổ chức và quản lý lớp theo nguyên lý OOP, thiết kế hướng mô-đun, cũng như phát triển giao diện tương tác thời gian thực. Kết quả là phiên bản Tetris có thể sinh khối ngẫu nhiên, di chuyển, xoay, rơi tự động, xử lý va chạm, xóa hàng và cập nhật điểm, đồng thời có giao diện trực quan.

Về hướng phát triển, nhóm dự định nâng cấp game với các tính năng như chế độ chơi nâng cao, lưu/ tải trò chơi, bảng xếp hạng, âm thanh và giao diện sinh động hơn, cũng như phát triển phiên bản trực tuyến nhiều người chơi.

PHỤ LỤC

Mã nguồn của toàn bộ bài tập lớn đã được nhóm tổ chức và lưu trữ trên kho GitHub nhằm thuận tiện cho việc quản lý, theo dõi tiến độ cũng như cộng tác trong quá trình phát triển. Tất cả các tệp mã nguồn, tài nguyên giao diện, cấu trúc thư mục và lịch sử chỉnh sửa đều được cập nhật đầy đủ trên kho lưu trữ này.

Link GitHub của dự án: https://github.com/ngoctu0405/Tetris_Game.git

TÀI LIỆU THAM KHẢO

Tiếng Việt

1. Nguyễn Minh Hải – Bài giảng Lập trình hướng đối tượng (OOP) – Java – Khoa Công nghệ Thông tin, Trường Đại học Sài Gòn, 2025.
2. Trần Văn Hòa – Giáo trình Lập trình Java căn bản và nâng cao – Nhà xuất bản Giáo dục Việt Nam, 2023.
3. Kteam – Lập trình Java Swing/AWT Cơ bản– Kteam Education – năm 2018.
4. Nguyễn Quang Trung – Phát triển trò chơi Tetris bằng Java Swing/AWT – Hướng dẫn kỹ thuật và tối ưu – Tạp chí Khoa học Máy tính, 2024.

Tiếng Anh

5. ZetCode – Java Tetris – Java 2D Game Tutorial – Trang hướng dẫn lập trình Java Game – năm 2023.
6. RyiSnow – How to Code Tetris in Java – Video hướng dẫn trên YouTube – năm 2023.
7. EDUCBA – Tetris Game in Java – Step by Step Tutorial – Trang đào tạo trực tuyến EDUCBA – năm 2023.
8. Oracle – The Java™ Tutorials – Trang tài liệu chính thức của Oracle – năm 2024.
9. Liang, Y. Daniel – Introduction to Java Programming, Comprehensive Version (12th Edition) – Nhà xuất bản Pearson Education – năm 2022.
10. Stack Overflow – Java KeyListener and Game Loop Discussions (Tetris Implementation) – Diễn đàn trao đổi lập trình – năm 2024.