Thẻ 1

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

BÀI TẬP LỚN
ĐỀ TÀI: Game Pacman
BỘ MÔN LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Giảng viên hướng dẫn: TS. Đỗ Thị Liên

NHÓM: 11

CÁC THÀNH VIÊN

Phạm Quốc Hùng	MSV: B23DCVT188
Trần Vũ Tiến Minh	MSV: B23DCCE067
Trần Hoàng Nam	MSV: B23DCCE070
Phạm Thanh An	MSV: B23DCCE004

Hà Nội – 2025

I. Yêu cầu

1. Giao diện người dùng (GUI) thân thiện và dễ hiểu

Người chơi thấy được các màn hình riêng biệt rõ ràng: Main Menu, Playing, Paused, HighScores. Trong khi chơi, người chơi thấy nhân vật Pacman, các Ghost, điểm số và bản đồ mê cung.

2. Gameplay đơn giản

Người chơi có thể điều khiển Pacman di chuyển theo 4 hướng lên, xuống, trái, phải trong một bản đồ mê cung.
Mục tiêu là ăn hết các dấu chấm trong mê cung mà không bị các con ma bắt.. Các dấu chấm nhỏ sẽ được Pacman ăn khi di chuyển qua chúng. Khi tất cả các dấu chấm nhỏ trong mê cung được ăn hết, người chơi sẽ hoàn thành màn chơi và chuyển sang màn tiếp theo.
Có các chấm lớn Power Up giúp Pacman "có sức mạnh tạm thời" là ăn được ma để cộng điểm.
Có bản đồ và các nhân vật Pacman, bốn con ma Blinky, Pinky, Inky, Clyde y như phần game gốc.

3. Tính thử thách và cạnh tranh

Người chơi mong muốn có hệ thống tính điểm để so sánh và vượt qua các điểm số trước đó.
Từng con ma có hành vi thông minh riêng, không di chuyển ngẫu nhiên hoàn toàn.

4. Âm thanh và hiệu ứng sống động

Người chơi kỳ vọng có hiệu ứng âm thanh khi ăn chấm, bị ma bắt, ăn ma, lúc thắng hoặc thua Game. Game cần có nhạc nền cổ điển đặc trưng của Pacman.

5. Hiệu suất ổn định

Game cần chạy mượt mà, không bị giật lag, tối thiểu 60 Khung hình / giây.
Khởi động nhanh, không bị lỗi trong quá trình chơi.

6. Khả năng lưu điểm số cao

Người chơi muốn game lưu lại điểm số cao nhất cho những lần khởi động Game sau này.

7. Khả năng chơi lại dễ dàng

Sau khi thua, người chơi có thể nhanh chóng chơi lại mà không cần khởi động lại toàn bộ game.

Phân công công việc

Thành viên	Công việc
Trần Vũ Tiến Minh	Thiết kế logic các máy trạng thái. Viết package screens gồm 4 màn hình: MainMenu, Playing, Paused, HighScores. Viết Game.java xử lý vòng lặp Game. Tìm file resources ảnh. âm thanh… cho Game. Sửa bug và xử lý xung đột khi merge code vào dự án.
Phạm Quốc Hùng	-Thiết kế và phát triển Package entities: + Thiết kế lớp cơ sở trừu tượng Interface ImmutableEntity. + Thiết kế lớp trừu tượng Entity. + Thiết kế lớp Pacman. + Thiết kế lớp Ghost. + Thiết kế thuật toán đuổi theo Pacman cho mỗi Ghost. + Thiết kế lớp cho từng Ghost(Blinky, Clyde, Inky, Pinky).
Trần Hoàng Nam	-Thiết kế và phát triển package resources: + Thiết kế lớp resource. + Thiết kế lớp ResourcesManager. + Thiết kế lớp SpriteSheet.
Phạm Thanh An	-Thiết kế và phát triển package maps: + Thiết kế lớp interface ImmutableMap. + Thiết kế lớp Map. + Thiết kế lớp Tile.

II. Phân tích thiết kế ra lớp

1. Root package (com.karas.pacman)

Lớp Main

Mục đích: Là điểm khởi đầu của ứng dụng. Mục đích duy nhất của nó là tạo một đối tượng Game và khởi chạy trò chơi.

Lớp Game

Game extends JPanel implements Exitable, Runnable, KeyListener
Mục đích: Là lõi xử lý chính của trò chơi, quản lý vòng lặp chính game loop, cửa sổ frame, xử lý input và điều phối việc cập nhật logic, vẽ hình. Lớp đóng vai trò là máy chuyển đổi trạng thái State Machine giữa các màn hình Screen đại diện cho các trạng thái lớn: Playing, Paused, Menu…

Thuộc tính:
running: Biến boolean theo dõi game vẫn nên tiếp tục chạy.
frame: Đối tượng cửa sổ hiển thị game - JFrame.
state: Trạng thái hiện tại của game - State.
thread: Đối tượng quản lý game thread - Thread.
frameCount: đếm số frame trong 1 giây (để tính FPS).
screenManager: Quản lý các màn hình và chuyển đổi giữa chúng.
resourcesManager: Quản lý tải và lưu trữ tài nguyên game.
updateCount: Đếm số lần update trong 1 giây (để tính UPS).
updateTimer: Bộ đếm thời gian tích lũy cho các hoạt động update.
repaintTimer: Bộ đếm thời gian tích lũy cho các hoạt động repaint.
logTimer: Bộ đếm thời gian tích lũy cho các hoạt động log.
scale: Tỷ lệ scale để điều chỉnh kích thước game theo cửa sổ.

Thuộc tính tĩnh:
LOGGER: Ghi log thông tin hoạt động của game.

Phương thức:
Game(): Khởi tạo game với các thành phần cơ bản.
enter(): Khởi tạo game thread tách biệt với main thread.
exit(): Hợp game thread vào main thread và đóng game.
run(): Thực thi game loop, liên tục gọi updateGame() và repaint().
updateGame(): Cập nhật logic của trạng thái game hiện tại.
paintComponent(): Vẽ các đối tượng của trạng thái game hiện tại lên màn hình. Kế thừa từ lớp JPanel, được gọi bởi repaint() và chạy trên thread riêng để vẽ lên màn hình.
keyPressed(), keyReleased(), keyTyped(): Xử lý input, kế thừa từ interface KeyListener.
logGame(): Cập nhật thông tin UPS/FPS lên tiêu đề cửa sổ.

Lớp Configs

Mục đích: Lưu trữ các hằng số và cấu hình tĩnh của game một cách tập trung, giúp dễ dàng thay đổi các tham số như kích thước ô, tốc độ, điểm số, v.v.

Một số Hằng số:
UPS_TARGET: Số update mỗi giây
FPS_TARGET: Số frame mỗi giây
TITLE: Tiêu đề game
DEFAULT_SCALE: Tỷ lệ scale mặc định

2. Package entities

Interface ImmutableEntity

Mục đích: Là lớp cơ sở trừu tượng cho tất cả các đối tượng động trong game (Pacman, Ghost). Nó định nghĩa các thuộc tính và hành vi chung.
Đặc biệt, trong lớp này có định nghĩa kiểu dữ liệu enum là Stage để định nghĩa các trạng thái hoạt động của entity: IDLE, HUNTER, PREY, DEAD

Lớp Trừu Tượng Entity

Mục đích: Implement ImmutableEntity. Là lớp cơ sở trừu tượng cho tất cả các đối tượng động trong game (Pacman, Ghost). Nó định nghĩa các thuộc tính và hành vi chung.
Đặc biệt, trong lớp này có định nghĩa kiểu dữ liệu enum là Stage để định nghĩa các trạng thái hoạt động của entity: IDLE, HUNTER, PREY, DEAD

Thuộc tính:
Map: Tham chiếu đến bản đồ game.

Sprite: Đối tượng hiển thị hình ảnh

speed: Tốc độ di chuyển.

position: Vị trí hiện tại.

direction: Hướng di chuyển hiện tại.

state: Trạng thái hoạt động hiện tại.

preIdleState: Trạng thái trước khi chuyển sang IDLE.

Phương thức:

Công khai:

update(double deltaTime): Phương thức trừu tượng, cập nhật logic mỗi khung hình.

reset(): Phương thức trừu tượng, khôi phục trạng thái ban đầu.

getState(): Trả về trạng thái hiện tại.

enterState(State nextState): Chuyển đổi trạng thái với cơ chế xử lý chuyển tiếp.

enterPreIdleState(): Khôi phục trạng thái trước khi vào IDLE.

getPosition(): Lấy vị trí thực tế (pixel).

getDirection(): Lấy hướng di chuyển hiện tại.

getGridPosition(): Chuyển đổi vị trí sang hệ tọa độ lưới

collidesWith(Entity other): Kiểm tra va chạm với entity khác dựa trên khoảng cách.

repaint(Graphics2D G): Hiển thị entity lên màn hình.

Được bảo vệ:

Entity(Vector2 gridPosition, Direction direction, int speed, EntitySprite SpriteRef, ImmutableMap MapRef)

handleStateTransition(State nextState): Phương thức trừu tượng, xử lý logic chuyển đổi trạng thái.

isCenteredInTile(): Kiểm tra entity có nằm chính giữa ô lưới không

canMoveInDirection(Direction d): Kiểm tra khả năng di chuyển theo hướng

setDirection(Direction d): Thiết lập hướng di chuyển mới

move(double deltaTime): Thực hiện di chuyển với xử lý đường hầm

setSpeed(int speed): Thiết lập tốc độ di chuyển

setGridPosition(Vector2 gridPosition): Đặt vị trí theo hệ lưới

getSprite() / setSprite(): Truy cập và thay đổi sprite hiển thị

Lớp Pacman

Mục đích: Kế thừa từ Entity, đại diện cho người chơi. Quản lý logic di chuyển, xử lý va chạm, ăn điểm và hoạt ảnh của Pacman.

Thuộc tính riêng:

DeathSound: Âm thanh khi Pacman chết.

baseSprite: Sprite cơ bản của Pacman (khi sống).

deathSprite: Sprite khi Pacman chết.

nextDirection: Hướng di chuyển tiếp theo (volatile).

Phương thức:

Công khai:

Pacman(ImmutableMap, BufferedImage[], BufferedImage[], Sound): Constructor - khởi tạo Pacman với vị trí, hướng, tốc độ, sprite và bản đồ.

setNextDirection(Direction): Thiết lập hướng di chuyển tiếp theo từ input người chơi

update(double deltaTime): (Override) Cập nhật logic Pacman mỗi khung hình dựa trên trạng thái hiện tại

reset(): (Override) Khôi phục Pacman về trạng thái ban đầu

Được bảo vệ:
handleStateTransition(State): (Override) Xử lý logic chuyển đổi trạng thái.

Lớp Ghost

Mục đích: Kế thừa từ Entity, đại diện cho các con ma. Quản lý logic di chuyển ma, trạng thái đuổi theo, sợ hãi và va chạm với Pacman.

Hằng số
HOME_POSITION=Map.toPixelVector2(Configs.Grid.GHOST_HOME);

Thuộc tính riêng:

DeathSound: Âm thanh khi ghost chết.

Pacman: Tham chiếu đến Pacman để truy đuổi/tránh né.

baseSprite: Sprite hiển thị bình thường.

preySprite: Sprite khi ghost sợ hãi.

deathSprite: Sprite khi ghost chết.

baseSpeed: Tốc độ cơ bản của ghost

startGridPosition: Vị trí bắt đầu trên lưới.

startDirection: Hướng bắt đầu.

prevGridPos: Vị trí lưới trước đó để xác định khi nào cần đổi hướng.

deathSprite: Sprite khi ghost chết.

baseSpeed: Tốc độ cơ bản của ghost.

startGridPosition: Vị trí bắt đầu trên lưới.

startDirection: Hướng bắt đầu.

prevGridPos: Vị trí lưới trước đó để xác định khi nào cần đổi hướng.

Phương thức:
Công khai
enableFlashing(): Kích hoạt hiệu ứng nhấp nháy khi ghost ở trạng thái PREY
update(double deltaTime): (Override) Cập nhật logic theo trạng thái với AI phức tạp
reset(): (Override) Khôi phục ghost về trạng thái ban đầu

Bảo vệ

Ghost(Vector2 gridPosition, Direction direction, int speed,
BufferedImage[] BaseImages, BufferedImage[] PreyImages,
BufferedImage[] DeathImages, Sound DeathSound,
ImmutableEntity PacmanRef, ImmutableMap MapRef)
handleStateTransition(State nextState): Xử lý hiệu ứng khi chuyển trạng thái

Phương thức trừu tượng
findHunterTarget(ImmutableEntity PacmanRef): mỗi loại ghost triển khai chiến lược săn mồi riêng

Riêng tư
updateDirection(): Cập nhật hướng di chuyển thông minh dựa trên trạng thái

getValidDirections(): Lấy tập hợp các hướng di chuyển hợp lệ

isAtHomePosition(): Kiểm tra ghost đã về đến vị trí nhà chưa

Lớp Blinky, Pinky, Inky, Clyde
Là bốn con Ghost chính ở trong Game, kế thừa từ Ghost.

Phương thức:
findHunterTarget(ImmutableEntity PacmanRef): (Override) Thực hiện thuật toán chọn mục tiêu để săn Pacman

3. Package screens

Interface Screen
Mục đích: Định nghĩa một giao diện màn hình chung cho tất cả các trạng thái của game. Điều này tuân theo mẫu thiết kế State, cho phép Game xử lý các trạng thái màn hình khác nhau một cách thống nhất.

Phương thức:

void enter(): Xử lý khi vào trạng thái.

void exit(): Xử lý khi thoát trạng thái.

State update(): Cập nhật logic cho trạng thái này và trả về trạng thái hiện tại hoặc trạng thái tiếp theo.

void render(Graphics2D g): Vẽ các đối tượng cho trạng thái này.

void input(KeyEvent e): Xử lý input cho trạng thái này.

Lớp ScreenManager
Mục đích: Quản lý việc chuyển đổi giữa các màn hình game

Thuộc tính:

screens: Map lưu trữ các screen theo class

current: Screen hiện tại đang active

Phương thức:
ScreenManager(ResourcesManager): Khởi tạo với tất cả screens

exit(): Gọi exit() trên tất cả screens

input(KeyEvent): Chuyển input đến screen hiện tại

repaint(Graphics2D): Vẽ screen hiện tại

update(double): Cập nhật screen hiện tại và xử lý chuyển đổi

Lớp Playing

Mục đích: Implement State, định nghĩa logic khi game đang được chơi. Đây là trạng thái chính của game.

Thuộc tính:
LOGGER: Đối tượng ghi log cho lớp

FontMedium: Font chữ hiển thị điểm số.

NewGameSound, NormalSound, PowerupSound, WonSound: Các âm thanh tương ứng trạng thái game.

ScoreDatabase: Tham chiếu đến cơ sở dữ liệu điểm số.

ResourcesManager: Quản lý tài nguyên game.

map: Tham chiếu đến bản đồ game.

pacman: Đối tượng nhân vật Pacman.

ghosts: Danh sách các đối tượng ma.

scores: Danh sách sprite hiển thị điểm.

state: Trạng thái hoạt động hiện tại của game

stateCooldown: Thời gian đếm ngược cho trạng thái hiện tại

totalScore: Tổng điểm hiện tại của người chơi

nextScreen: Màn hình kế tiếp sẽ chuyển đến

Phương thức:
Công khai:
Playing(ResourcesManager): Khởi tạo màn hình chơi game

enter(Class<extends Screen>): Được gọi khi vào màn hình, khôi phục trạng thái

exit(): Được gọi khi rời màn hình, tạm dừng các hoạt động

update(double deltaTime): Cập nhật logic game mỗi khung hình, trả về màn hình kế tiếp

repaint(Graphics2D G): Hiển thị tất cả thành phần game lên màn hình

input(KeyEvent e): Xử lý đầu vào từ bàn phím

Private:
createMap(ResourcesManager): Tạo đối tượng bản đồ game

createPacman(Map, ResourcesManager): Tạo đối tượng Pacman

createGhosts(Pacman, Map, ResourcesManager): Tạo danh sách các ma

createScores(ResourcesManager): Tạo sprite hiển thị điểm

enterState(State): Chuyển đổi trạng thái với cơ chế xử lý chuyển tiếp

updateState(double): Cập nhật logic trạng thái game

handleCollision(Sound): Kiểm tra và xử lý va chạm giữa các entity

paintTotalScore(Graphics2D): Hiển thị tổng điểm lên màn hình

Enum State:
START, NORMAL, POWERUP, LOST, WON: Các trạng thái của game

Lớp Paused

Mục đích: Implement State, định nghĩa logic khi game bị tạm dừng. Cho ra các tùy chọn chỉnh cài đặt.

Thuộc tính:
LOGGER: Đối tượng ghi log cho lớp.

FontLarge: Font chữ hiển thị tiêu đề "PAUSED".

PlayingScreen: Tham chiếu đến màn hình game đang chơi.

navigator: Đối tượng điều hướng menu.

_nextScreen: Màn hình kế tiếp sẽ chuyển đến

Phương thức:
Công khai:
Paused(Screen, ResourcesManager): Khởi tạo màn hình tạm dừng với màn hình game và quản lý tài nguyên

enter(Class< extends Screen>): Được gọi khi vào màn hình, khởi tạo trạng thái

exit(): Được gọi khi rời màn hình

update(double deltaTime): Cập nhật logic mỗi khung hình, trả về màn hình kế tiếp

repaint(Graphics2D G): Hiển thị màn hình game ở nền và vẽ giao diện tạm dừng lên trên

input(KeyEvent e): Xử lý đầu vào từ bàn phím để điều hướng menu

Private:
paintOverlay(Graphics2D): Vẽ lớp phủ mờ lên trên màn hình game

paintPausedText(Graphics2D): Vẽ chữ "PAUSED" lên màn hình

Lớp MainMenu

Mục đích: Implement State, định nghĩa logic khi game ở sảng chờ.

Thuộc tính:
TitleImage: Hình ảnh tiêu đề của game

navigator: Đối tượng điều hướng menu chính

nextScreen: Màn hình kế tiếp sẽ chuyển đến

Phương thức:
Công khai:
MainMenu(ResourcesManager): Khởi tạo màn hình menu chính với quản lý tài nguyên

enter(Class<extends Screen>): Được gọi khi vào màn hình, khởi tạo trạng thái

exit(): Được gọi khi rời màn hình

update(double deltaTime): Cập nhật logic mỗi khung hình, trả về màn hình kế tiếp

repaint(Graphics2D G): Hiển thị nền, hình ảnh tiêu đề và menu điều hướng

input(KeyEvent e): Xử lý đầu vào từ bàn phím để điều hướng menu

Private:
paintBackgroundColor(Graphics2D): Vẽ màu nền cho menu chính.
paintTitleImage(Graphics2D): Vẽ hình ảnh tiêu đề được căn giữa và điều chỉnh kích thước

Lớp HighScores

Mục đích: Implement Screen, định nghĩa logic khi game kết thúc, hiện các thông tin điểm số score và lựa chọn chơi tiếp hoặc chơi lại.

Thuộc tính:
ScoreDatabase: Tham chiếu đến cơ sở dữ liệu điểm số.

HeaderImage: Hình ảnh tiêu đề cho màn hình điểm cao

FontSmall, FontMedium: Các font chữ sử dụng để hiển thị văn bản
highscores: Danh sách các điểm số cao (dạng Iterable của ScoreDatabase.Entry)
nextScreen: Màn hình kế tiếp sẽ chuyển đến

Phương thức:
Công khai:
HighScores(ResourcesManager): Khởi tạo màn hình điểm cao với quản lý tài nguyên
enter(Class<? extends Screen>): Được gọi khi vào màn hình, khởi tạo trạng thái và lấy danh sách điểm cao
exit(): Được gọi khi rời màn hình
update(double deltaTime): Cập nhật logic mỗi khung hình, trả về màn hình kế tiếp
repaint(Graphics2D G): Hiển thị hình ảnh tiêu đề, văn bản và danh sách điểm cao
input(KeyEvent e): Xử lý đầu vào từ bàn phím (nhấn Enter để chuyển sang màn hình Playing)

Private:
paintHeaderImage(Graphics2D): Vẽ hình ảnh tiêu đề được căn giữa và điều chỉnh kích thước
paintText(Graphics2D): Vẽ các chuỗi văn bản "HIGH SCORES" và "NEW GAME"
paintHighScores(Graphics2D): Vẽ danh sách các điểm số cao theo thứ tự

4. Package commons

Interface Exitable

Mục đích: Định nghĩa interface cho các đối tượng cần thực hiện dọn dẹp tài nguyên khi thoát
Phương thức: void exit() - giải phóng tài nguyên

Interface Paintable

Mục đích: Định nghĩa interface cho các đối tượng có thể được vẽ lên màn hình
Phương thức: void repaint(Graphics2D G) - vẽ đối tượng

Enum Direction

Mục đích: Định nghĩa một tập hợp các hướng di chuyển có thể có UP, DOWN, LEFT, RIGHT. Việc sử dụng enum giúp tăng sự rõ ràng và tránh lỗi so với việc dùng số nguyên hoặc chuỗi.

Thuộc tính:
UP, RIGHT, DOWN, LEFT: Các hằng số đại diện cho 4 hướng di chuyển

_VECTORS: Mảng chứa các vector tương ứng với mỗi hướng

Phương thức:
Công khai:
random(): Trả về một hướng ngẫu nhiên từ 4 hướng có sẵn

toVector2(): Chuyển đổi hướng thành vector 2D tương ứng

isVertical(): Kiểm tra xem hướng hiện tại có phải là hướng dọc (lên/xuống) không

opposite(): Trả về hướng ngược lại với hướng hiện tại

Record Vector2

Mục đích: Biểu diễn một vector 2D, được dùng để lưu trữ vị trí (x, y) hoặc hướng di chuyển của các thực thể.
Thuộc tính:
x: Tọa độ x (kiểu double)
y: Tọa độ y (kiểu double)

Phương thức:
Công khai:
ix(): Trả về tọa độ x dạng integer.

iy(): Trả về tọa độ y dạng integer.

abs(): Trả về vector mới với giá trị tuyệt đối của x và y.

ceil(): Trả về vector mới với giá trị làm tròn lên của x và y

floor(): Trả về vector mới với giá trị làm tròn xuống của x và y

add(double n): Cộng vector với một số thực n, trả về vector mới

sub(double n): Trừ vector cho một số thực n, trả về vector mới

mul(double n): Nhân vector với một số thực n, trả về vector mới

div(double n): Chia vector cho một số thực n, trả về vector mới

mod(double n): Lấy phần dư của vector khi chia cho số thực n, trả về vector mới

add(Vector2 other): Cộng vector với vector khác, trả về vector mới

sub(Vector2 other): Trừ vector cho vector khác, trả về vector mới

distance(Vector2 other): Tính khoảng cách Euclidean giữa hai vector

furthestFrom(Vector2 other, EnumSet<Direction> directions): Tìm hướng trong tập directions sao cho vector cộng với hướng đó xa vector other nhất

closestTo(Vector2 other, EnumSet<Direction> directions): Tìm hướng trong tập directions sao cho vector cộng với hướng đó gần vector other nhất

Thuộc tính chính: double x, double y: Tọa độ của vector.
Phương thức chính:
Vector2 add(), sub(), mul(), div(), mod(): Cộng trừ nhân chia với vector còn lại, hoặc là một số double.

5. Package maps

Interface ImmutableMap

Mục đích: Cung cấp interface chỉ đọc cho các thuộc tính của Map

Phương thức:

int getPelletCounts()

boolean isMovableAt(Vector2 gridPosition)

boolean canMoveInDirection(Vector2 position, Direction nextDirection)

Vector2 tryTunneling(Vector2 position, Direction direction)

Lớp Map

Mục đích: Quản lý tất cả dữ liệu về bản đồ game, bao gồm layout tường, vị trí các chấm dots và các chấm năng lượng power pellets.

Thuộc tính:
Thuộc tính không tĩnh (Instance Fields):
WakaSounds: Mảng âm thanh khi ăn pellet

MapImage: Hình ảnh nền của bản đồ

PelletImage: Hình ảnh pellet thường

PowerupImage: Hình ảnh powerup

pelletCounts: Số lượng pellet còn lại trên bản đồ

tiles: Mảng 2D đại diện cho các ô trong bản đồ

Phương thức:
Phương thức tĩnh (Static Methods):
Public:
toGridVector2(Vector2): Chuyển đổi vị trí pixel sang tọa độ lưới

toPixelVector2(Vector2): Chuyển đổi tọa độ lưới sang vị trí pixel

isCenteredInTile(Vector2): Kiểm tra entity có nằm chính giữa ô không

Private:
isXYCenteredInTile(Vector2): Kiểm tra căn giữa theo từng trục X và Y

checkBound(Vector2): Kiểm tra vị trí có trong biên bản đồ không

Phương thức không tĩnh (Instance Methods):
Công khai:
Map(): Constructor - khởi tạo bản đồ với tilemap và resources

reset(): Đặt lại bản đồ với tilemap mới

getPelletCounts(): Trả về số pellet còn lại

tryTunneling(): Xử lý cơ chế đường hầm (teleport)

isMovableAt(): Kiểm tra vị trí có thể di chuyển đến không

canMoveInDirection(): Kiểm tra khả năng di chuyển theo hướng

tryEatAt(): Thử ăn vật phẩm tại vị trí và trả về loại tile

repaint(): Vẽ bản đồ lên Graphics2D

Private:
setTilemap(): Thiết lập tilemap và đếm số pellet

paintConsumables(): Vẽ các vật phẩm (pellet và powerup)

Enum Tile
Mục đích: Định nghĩa các ô tile trong Map
Thuộc tính (Enum Constants):
NONE: Ô trống, không có gì

WALL: Tường, không thể di chuyển qua

PELLET: Pellet thường, Pacman có thể ăn để ghi điểm

POWERUP: Powerup, Pacman ăn để vào trạng thái săn ma

TUNNEL: Đường hầm, cho phép teleport từ đầu này sang đầu kia

Phương thức:
Phương thức tĩnh (Static Methods):
Công khai:
fromChar(char): Chuyển đổi ký tự sang giá trị Tile tương ứng

6. Package resources

Lớp ResourcesManager

Mục đích:

- Chịu trách nhiệm khởi tạo, tải (load) và lưu trữ tất cả các tài nguyên game (Hình ảnh, Âm thanh, Font chữ, Dữ liệu điểm số, Bản đồ) từ ổ cứng vào bộ nhớ khi game bắt đầu.
- Cung cấp các phương thức để các lớp khác truy xuất tài nguyên đã tải.

- Quản lý việc giải phóng tài nguyên hoặc lưu dữ liệu (như High Score) khi thoát game thông qua phương thức exit().

Thuộc tính:

Thuộc tính tĩnh (Static Fields):

LOGGER: Đối tượng ghi log cho lớp (static final)

Thuộc tính không tĩnh (Instance Fields):

soundMap: EnumMap lưu trữ các âm thanh (Resource → Sound)

imageMap: EnumMap lưu trữ các hình ảnh (Resource → BufferedImage)

spriteMap: EnumMap lưu trữ các sprite (SpriteSheet → BufferedImage[])

tilemap: Mảng 2D các Tile đại diện cho bản đồ game

fonts: Mảng các Font chữ với các kích thước khác nhau

database: Đối tượng ScoreDatabase để quản lý điểm số

Phương thức:

Phương thức tĩnh (Static Methods):

Factory Methods:

initSoundMap(): Khởi tạo và tải tất cả âm thanh

initImageMap(): Khởi tạo và tải tất cả hình ảnh

initSpriteMap(BufferedImage): Khởi tạo sprite từ sprite sheet

initTilemap(): Khởi tạo và tải bản đồ game

initFonts(): Khởi tạo và tải font chữ

initDatabase(): Khởi tạo cơ sở dữ liệu điểm số

Resource Loaders:

loadSound(String): Tải file âm thanh từ đường dẫn

loadImage(String): Tải file hình ảnh từ đường dẫn

loadTilemap(String, Vector2): Tải và phân tích file bản đồ

loadFont(String, float): Tải file font chữ

loadOrCreateDatabase(String): Tải hoặc tạo mới database

Utility Methods:

createSquare(int): Tạo hình ảnh vuông (cho pellet/powerup)

handleException(Exception, boolean): Xử lý ngoại lệ thống nhất

Phương thức không tĩnh (Instance Methods):

Constructor & Lifecycle:

ResourcesManager(): Constructor - khởi tạo tất cả tài nguyên

exit(): Giải phóng tài nguyên và lưu database

Getters:

getSound(Resource): Lấy âm thanh theo resource

getImage(Resource): Lấy hình ảnh theo resource

getSprite(SpriteSheet): Lấy mảng sprite theo sprite sheet

getTilemap(): Lấy bản sao của tilemap

getFont(int): Lấy font theo kích thước

getDatabase(): Lấy đối tượng database

Enum Resource

Thuộc tính:

Các hằng số enum (Enum Constants):

WINDOW_ICON: Biểu tượng cửa sổ game

TITLE_IMAGE: Hình ảnh tiêu đề game

HIGHSCORE_IMAGE: Hình ảnh màn hình điểm cao

MAP_IMAGE: Hình ảnh bản đồ game

SPRITE_SHEET: Tập hợp sprite

TILEMAP: Dữ liệu bản đồ dạng text

FONT: Font chữ game

DATABASE_FILE: File cơ sở dữ liệu

EAT_WA_SOUND: Âm thanh ăn pellet (Wa)

EAT_KA_SOUND: Âm thanh ăn pellet (Ka)

PACMAN_DEATH_SOUND: Âm thanh Pacman chết

GHOST_DEATH_SOUND: Âm thanh ma chết

GAME_START_SOUND: Âm thanh bắt đầu game

GAME_NORMAL_SOUND: Âm thanh game bình thường

GAME_POWERUP_SOUND: Âm thanh game powerup

GAME_WON_SOUND: Âm thanh thắng game

Thuộc tính instance (Instance Fields):

path: Đường dẫn đến file tài nguyên (String)

isCritical: Cờ xác định tài nguyên có quan trọng không (boolean)

Phương thức:

Phương thức không tĩnh (Instance Methods):

Công khai:

getPath(): Trả về đường dẫn của tài nguyên

isCritical(): Kiểm tra tài nguyên có quan trọng không

Phương thức tĩnh (Static Methods):

Không có phương thức tĩnh trong enum này

Constructor:

Resource(String, boolean): Constructor private - khởi tạo enum constant với path và critical flag

Enum SpriteSheet

Mục đích: Định nghĩa vị trí và kích thước của các sprite cụ thể nằm trên tấm ảnh lớn spritesheet.png.

Thuộc tính:

Các hằng số enum (Enum Constants):

PACMAN: Sprite sheet cho Pacman sống

DEAD_PACMAN: Sprite sheet cho Pacman chết

BLINKY: Sprite sheet cho ghost Blinky

PREY_GHOST: Sprite sheet cho ghost khi bị săn

PINKY: Sprite sheet cho ghost Pinky

DEAD_GHOST: Sprite sheet cho ghost chết

INKY: Sprite sheet cho ghost Inky

SCORES: Sprite sheet cho hiển thị điểm số

CLYDE: Sprite sheet cho ghost Clyde

PELLETS: Sprite sheet cho pellet và powerup (tự tạo)

Thuộc tính instance (Instance Fields):

row: Vị trí hàng trong sprite sheet (int)

col: Vị trí cột bắt đầu trong sprite sheet (int)

len: Số lượng frame/sprite trong sheet (int)

Công khai:

getRow(): Trả về vị trí hàng trong sprite sheet

getCol(): Trả về vị trí cột bắt đầu

getLen(): Trả về số lượng frame

Constructor:

SpriteSheet(int, int, int): Constructor private - khởi tạo enum constant với row, col, len

7. Package audio

Lớp Sound

Mục đích: Quản lý hiển thị tạm thời các sprite điểm số (score pop-up) khi Pacman ăn ma. Các điểm số này sẽ hiển thị trong một khoảng thời gian ngắn rồi tự động biến mất.

Wrap đối tượng Clip để quản lý âm thanh

Thuộc tính: _clip - đối tượng Clip từ Java Sound API

Phương thức:

play(): Phát âm thanh một lần

loop(): Phát âm thanh lặp lại

pause(): Tạm dừng âm thanh

reset(): Đặt lại vị trí phát

exit(): Giải phóng tài nguyên

getDummy(): Trả về Sound giả cho xử lý lỗi

8. Package graphics

Lớp EntitySprite

Mục đích: Quản lý hiệu ứng hoạt ảnh cho các entity

Thuộc tính:

_Images: Mảng hình ảnh cho animation

_timer: Bộ đếm thời gian cho chuyển frame

_index: Chỉ số frame hiện tại

_offset: Độ lệch trong mảng hình ảnh

_frameCount: Số frame trong animation

_position: Vị trí hiển thị

Phương thức:

isAnimationEnded(): Kiểm tra animation kết thúc

setOffset(), setFrameCount(), setPosition(): Thiết lập tham số

update(double deltaTime): Cập nhật animation

repaint(Graphics2D G): Vẽ frame hiện tại

Lớp ScoreSprite

Thuộc tính:

Image: Hình ảnh hiển thị điểm số (BufferedImage) - final

duration: Thời gian hiển thị còn lại (double)

position: Vị trí hiển thị trên màn hình (Vector2)

Phương thức:
Công khai:
ScoreSprite(BufferedImage): Khởi tạo với hình ảnh điểm số, thời gian hiển thị ban đầu là 0 và vị trí null

setDisplay(Vector2): Kích hoạt hiển thị điểm số tại vị trí cụ thể với thời gian quy định

update(double deltaTime): Cập nhật thời gian hiển thị (giảm dần theo deltaTime)

repaint(Graphics2D G): Vẽ hình ảnh điểm số tại vị trí nếu thời gian hiển thị còn > 0

Lớp MenuNavigator

Thuộc tính:
Font: Font chữ sử dụng để hiển thị menu (Font) - final

options: Mảng các lựa chọn menu (MenuOption[]) - final

position: Vị trí bắt đầu hiển thị menu (Vector2) - final

hovering: Chỉ số của lựa chọn đang được chọn (int)

Phương thức:
Công khai:

MenuNavigator(Vector2, Font, MenuOption...): Khởi tạo navigator với vị trí, font và danh sách các lựa chọn menu

next(): Chuyển đến lựa chọn tiếp theo (vòng tròn)

previous(): Chuyển đến lựa chọn trước đó (vòng tròn)

select(): Trả về lớp màn hình tương ứng với lựa chọn hiện tại

repaint(Graphics2D G): Vẽ menu với lựa chọn hiện tại được highlight

Static Nested Record:

MenuOption:

label: Nhãn hiển thị của lựa chọn (String)

screenClass: Lớp màn hình tương ứng khi chọn (Class<? extends Screen>)

9. Package data

Lớp ScoreDatabase

Thuộc tính:

_fileLength: Số lượng bản ghi đã được đọc từ file (hoặc lưu vào file) (int)

_entries: Danh sách các bản ghi điểm số (ArrayList<Entry>)

_databaseFile: File lưu trữ cơ sở dữ liệu (File)

Phương thức:
Công khai:

ScoreDatabase(): Constructor khởi tạo cơ sở dữ liệu trống (không có file)

ScoreDatabase(File): Constructor khởi tạo từ file, đọc dữ liệu hiện có

addEntry(int): Thêm một bản ghi điểm số mới với ngày hiện tại

getTopEntries(int): Trả về danh sách các bản ghi điểm số cao nhất (số lượng được chỉ định)

save(): Lưu các bản ghi mới (chưa được lưu) vào file

Static Nested Record:

Entry:

score: Điểm số (int)

date: Ngày đạt được điểm số (LocalDate)

III. Quan hệ giữa các lớp

1. Quan hệ KẾ THỪA (is-a):

MainMenu, Playing, Paused, HighScores Implement Screen

Entity, Map, ScoreSprite, MenuNavigator, EntitySprite Implement Paintable

Game, ResourcesManager, ScreenManager, Sound Implement Exitable

Entity Implement ImmutableEntity

Map Implement ImmutableMap

Blinky, Pinky, Inky, Clyde Extends Ghost Extends Entity

Pacman Extends Entity

2. Quan hệ THÀNH PHẦN(has-a):

Quan hệ 1-1:

Game ↔ ScreenManager (1 Game có 1 ScreenManager)

ScreenManager ↔ Screen_current (1 ScreenManager quản lý 1 Screen hiện tại)

Playing ↔ Map (1 Playing có 1 Map)

Playing ↔ Pacman (1 Playing có 1 Pacman)

Pacman ↔ EntitySprite _baseSprite, _deathSprite

Ghost ↔ EntitySprite _baseSprite, _preySprite, _deathSprite

Inky ↔ Blinky (Inky tham chiếu đến Blinky)

Quan hệ 1-n:

ScreenManager ↔ Screen (1 ScreenManager quản lý nhiều Screen)

Playing ↔ Ghost (1 Playing có nhiều Ghost - Blinky, Pinky, Inky, Clyde)

Playing ↔ ScoreSprite (1 Playing có nhiều ScoreSprite)

ResourcesManager ↔ Sound (1 ResourcesManager quản lý nhiều Sound)

ResourcesManager ↔ BufferedImage (1 ResourcesManager quản lý nhiều image)

ResourcesManager ↔ Font (1 ResourcesManager quản lý nhiều font)

ScoreDatabase ↔ Entry (1 ScoreDatabase có nhiều Entry)

Quan hệ n-n:

Map ↔ Tile (Map chứa grid 2D của Tile - quan hệ ma trận)

ResourcesManager ↔ SpriteSheet (thông qua EnumMap - 1 manager quản lý nhiều sprite sheet

3. Quan hệ SỬ DỤNG (use-a):

Dependency:

Ghost ↔ Pacman (Ghost tham chiếu đến Pacman để tracking)

Entity ↔ ImmutableMap (Entity cần Map để di chuyển)

Playing ↔ ResourcesManager (Playing sử dụng ResourcesManager)

Paused ↔ Playing (Paused tham chiếu đến Playing screen)

Composition vs Aggregation:

Composition (mạnh - cùng lifecycle):

Game tạo và sở hữu ScreenManager, ResourcesManager

Playing tạo và sở hữu Map, Pacman, Ghosts

Aggregation (yếu - độc lập lifecycle):

ScreenManager quản lý các Screen nhưng không sở hữu

Ghost tham chiếu đến Pacman nhưng không sở hữu

IV. Xây dựng biểu đồ lớp UML

Chú thích:

I: Interface

C: Class

E: Enum

A: Abstract

: Quan hệ sở hữu

: Quan hệ sử dụng

: Quan hệ kế thừa

: Quan hệ implement

Diagram 1: Resources and Data

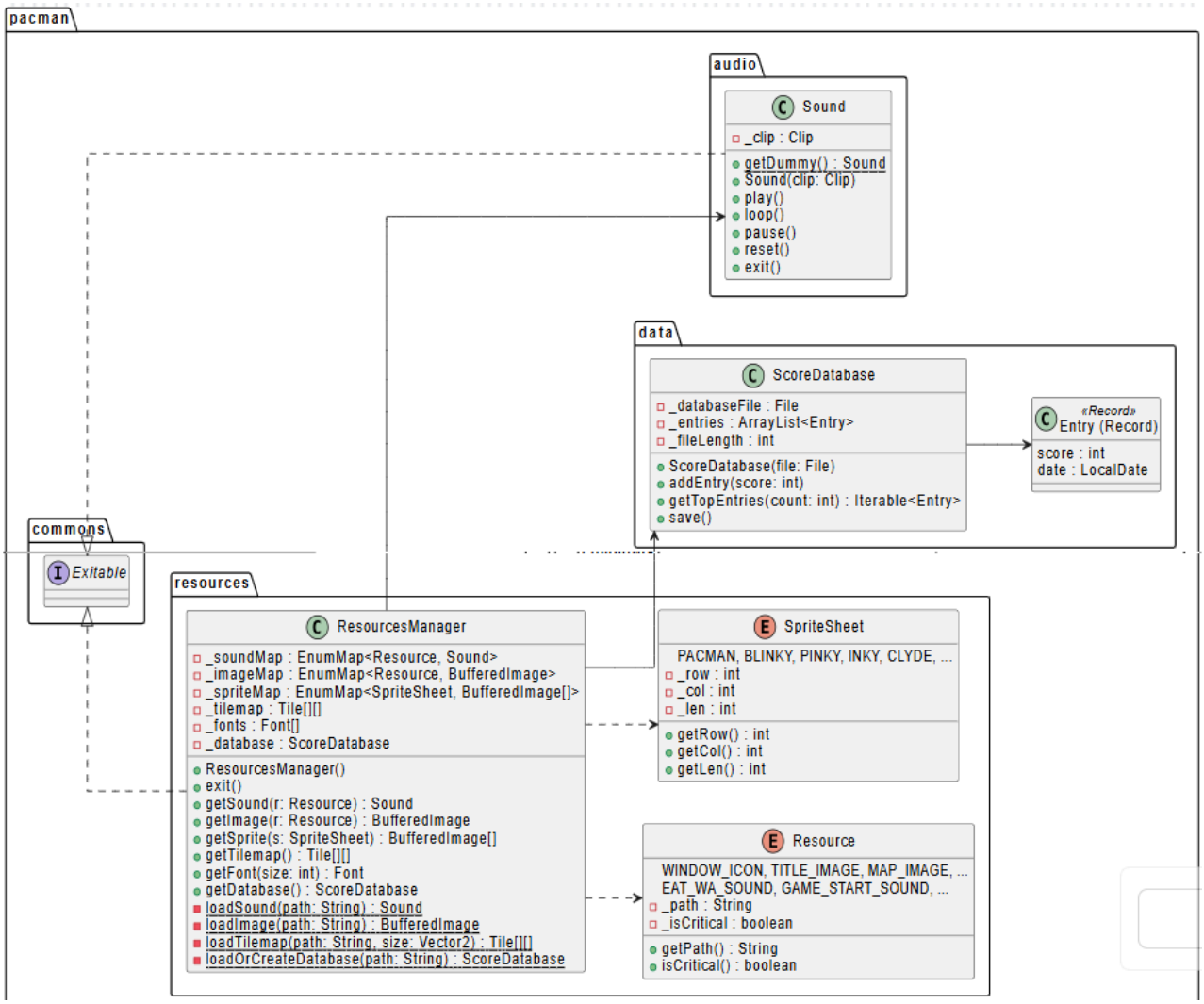


Diagram 2: Core & Commons

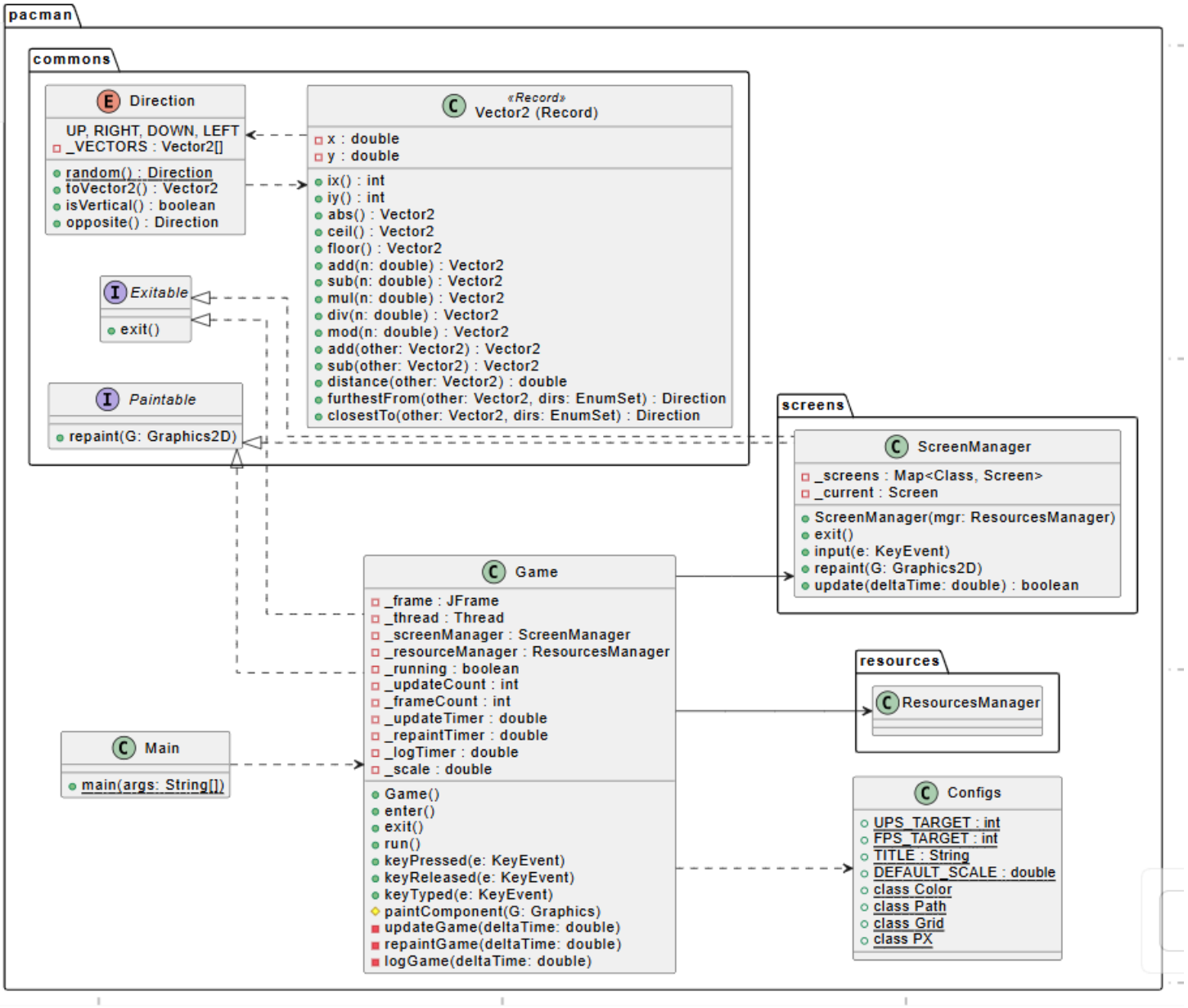


Diagram 3: Graphics & Maps

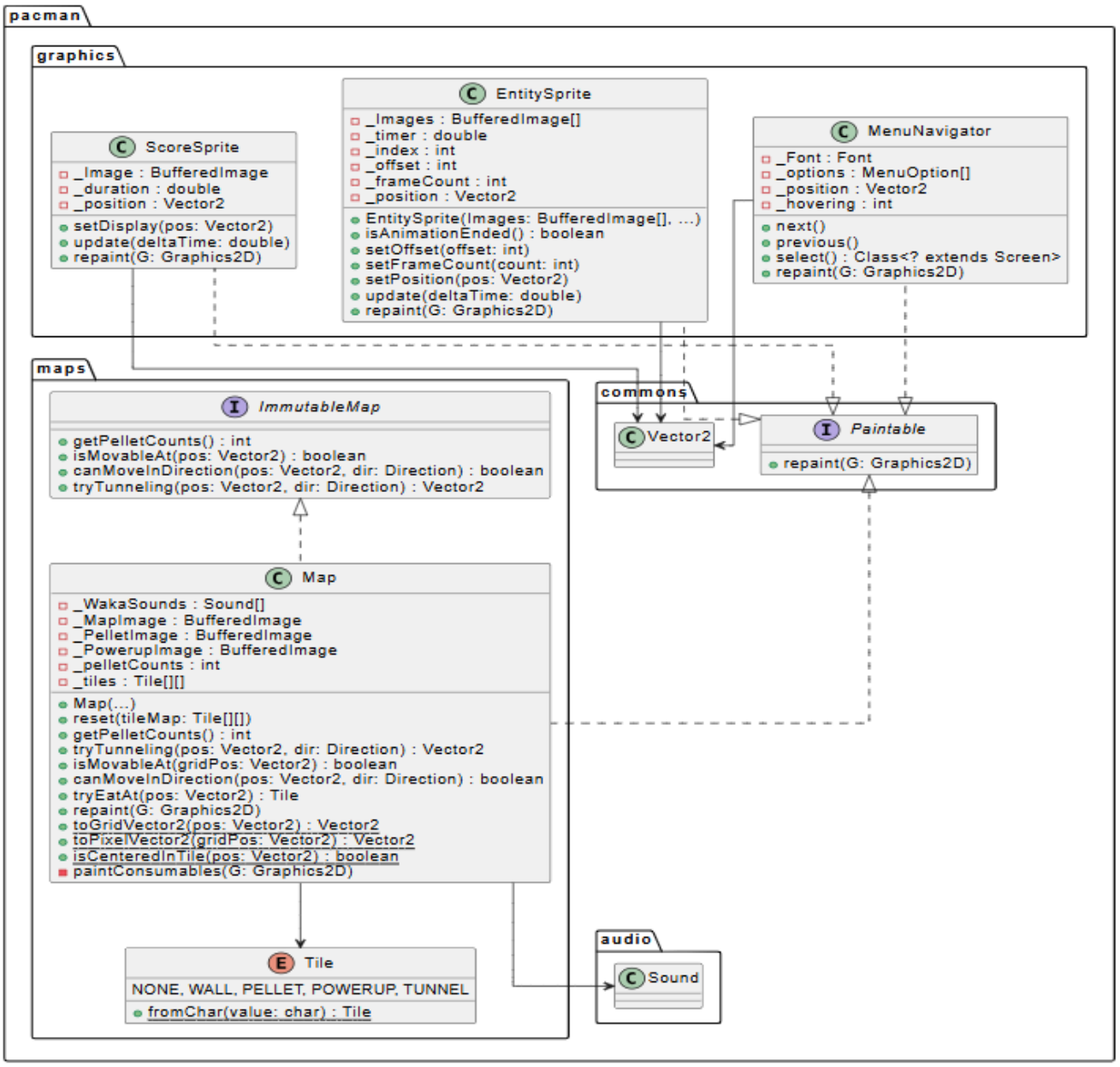


Diagram 4: Entities & Ghosts

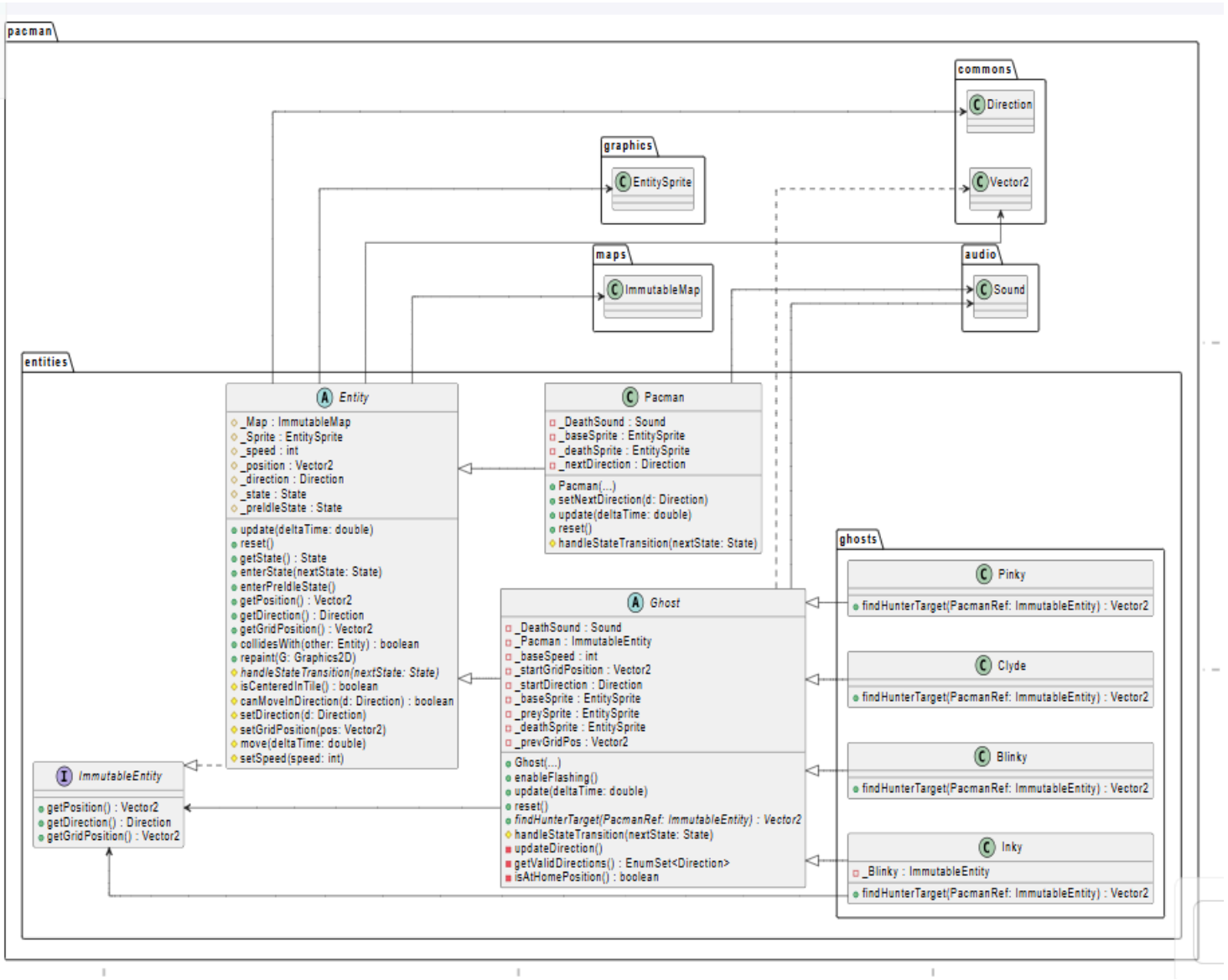
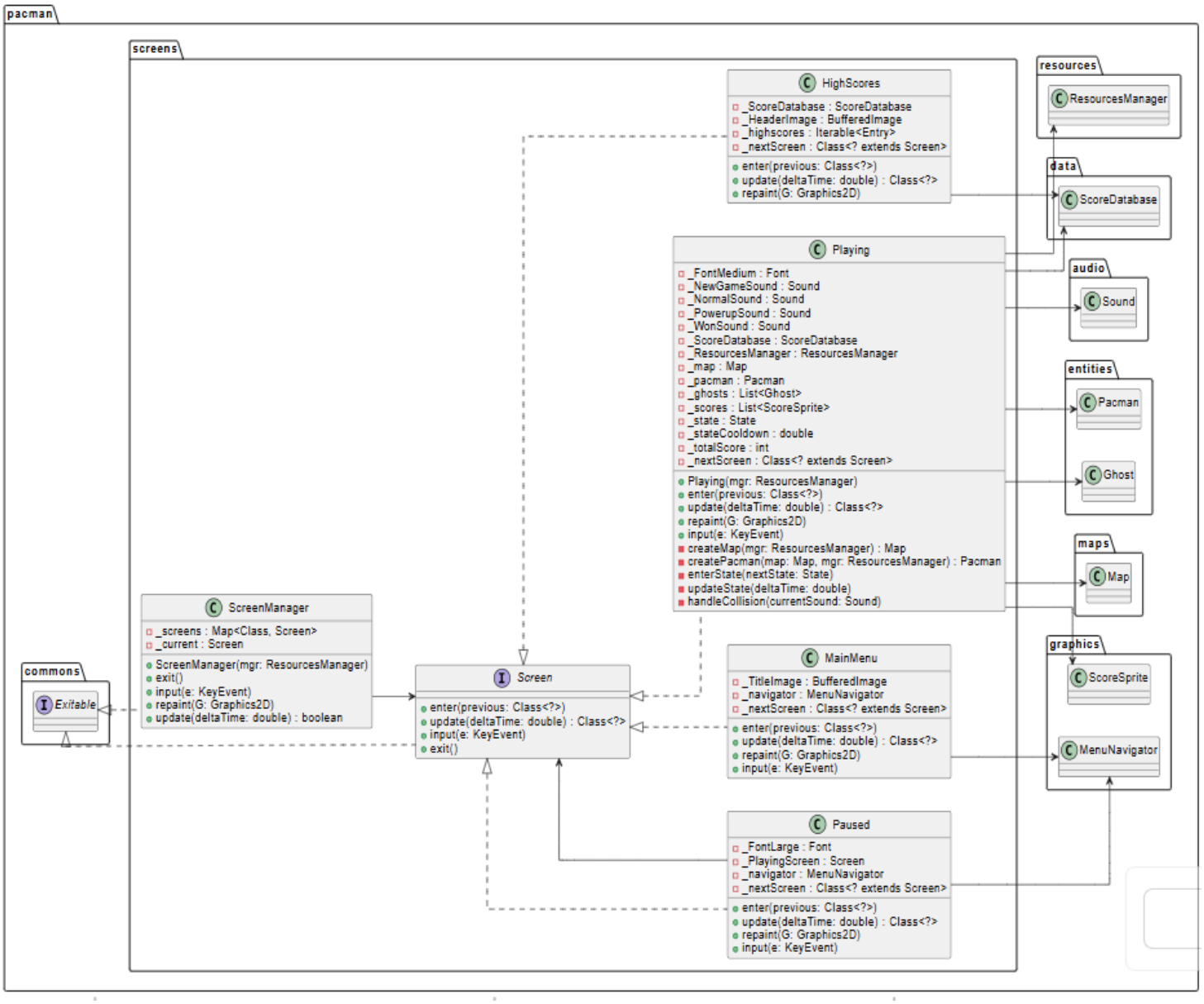

pacman
commons
Direction
Vector2
graphics
EntitySprite
maps
ImmutableMap
audio
Sound
entities
Entity
_Map : ImmutableMap
_Sprite : EntitySprite
_speed : int
_position : Vector2
_direction : Direction
_state : State
_preIdleState : State
update(deltaTime: double)
reset()
getState() : State
enterState(nextState: State)
enterPreIdleState()
getPosition() : Vector2
getDirection() : Direction
getGridPosition() : Vector2
collidesWith(other: Entity) : boolean
repaint(G: Graphics2D)
handleStateTransition(nextState: State)
isCenteredInTile() : boolean
canMoveInDirection(d: Direction) : boolean
setDirection(d: Direction)
setGridPosition(pos: Vector2)
move(deltaTime: double)
setSpeed(speed: int)
Pacman
_DeathSound : Sound
_baseSprite : EntitySprite
_deathSprite : EntitySprite
_nextDirection : Direction
Pacman(...)
setNextDirection(d: Direction)
update(deltaTime: double)
reset()
handleStateTransition(nextState: State)
ImmutableEntity
getPosition() : Vector2
getDirection() : Direction
getGridPosition() : Vector2
Ghost
_DeathSound : Sound
_Pacman : ImmutableEntity
_baseSpeed : int
_startGridPosition : Vector2
_startDirection : Direction
_baseSprite : EntitySprite
_preySprite : EntitySprite
_deathSprite : EntitySprite
_prevGridPos : Vector2
Ghost(...)
enableFlashing()
update(deltaTime: double)
reset()
findHunterTarget(PacmanRef: ImmutableEntity) : Vector2
handleStateTransition(nextState: State)
updateDirection()
getValidDirections() : EnumSet<Direction>
isAtHomePosition() : boolean
ghosts
Pinky
findHunterTarget(PacmanRef: ImmutableEntity) : Vector2
Clyde
findHunterTarget(PacmanRef: ImmutableEntity) : Vector2
Blinky
findHunterTarget(PacmanRef: ImmutableEntity) : Vector2
Inky
_Blinky : ImmutableEntity
findHunterTarget(PacmanRef: ImmutableEntity) : Vector2

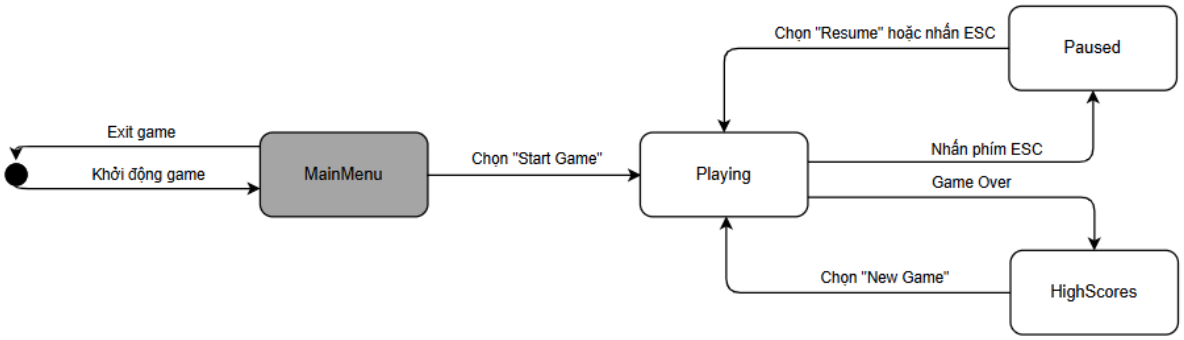
Diagram 5: Screens



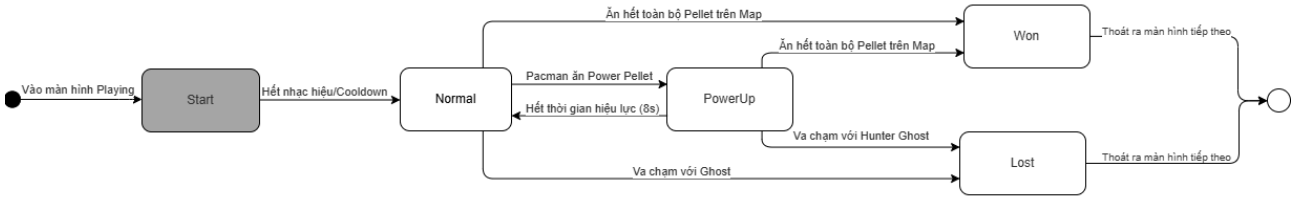
V. Đồ Hình Trạng thái - State Diagram

Đồ Hình Trạng Thái cho thấy một luồng trạng thái rõ ràng, phản ánh lối chơi cổ điển của dòng game Pacman. Đồ hình của các Lớp sử dụng Máy Trạng Thái State Machine như sau:

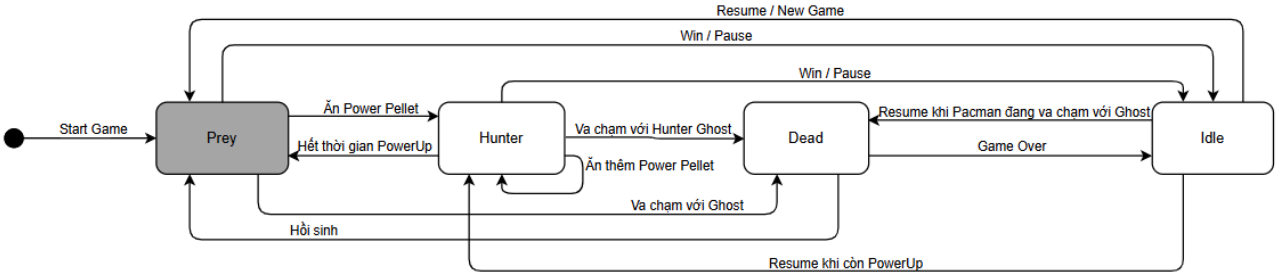
1. Lớp Game



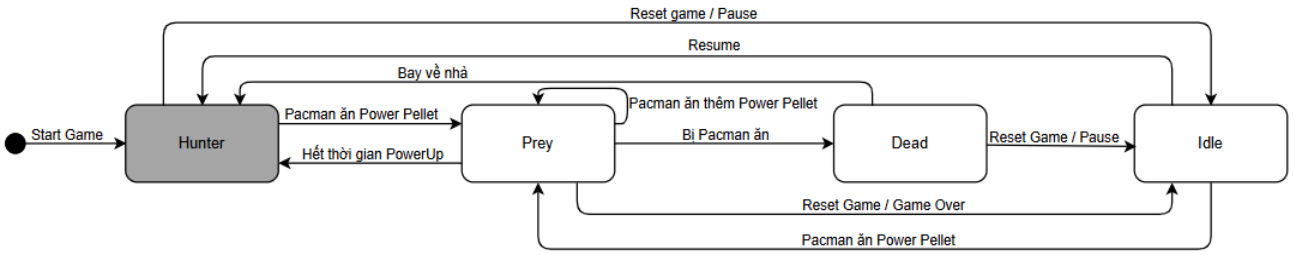
2. Lớp Playing



3. Lớp Pacman



4. Lớp Ghost



VI. File & Exception

1. File

Lớp ScoreDatabase phụ trách cơ sở dữ liệu (NoSQL)

Có thuộc tính File để đại diện file cơ sở dữ liệu điểm.
Khi tải file, đọc dòng bằng BufferedReader, parse bằng .split(",") và LocalDate.parse() theo (05d,YYYY-MM-DD)
phương thức save() Lưu các Điểm HighScores mới vào File.
File cơ sở dữ liệu được lưu dưới dạng .txt, lưu ở bên ngoài, ngang hàng với đường dẫn của file pacman.jar

Lớp ResourcesManager sở hữu, đọc các File tài nguyên và đưa các phương thức getter.

Đây là lớp quản lý tài nguyên tập trung, tất cả các File tài nguyên của chương trình đều do ResourcesManager sở hữu. Điều này giúp giảm thiểu cấp phát thừa các tài nguyên. Nâng cao tính sử dụng lại tài nguyên sẵn có.

2. Exception

Lớp ResourcesManager:

Khi đọc File tài nguyên nếu gặp Exception thì:
- Wrap nó trong một RuntimeException với thông điệp mô tả rõ ràng hơn.
- Kiểm tra tính quan trọng của tài nguyên, nếu không quan trọng thì Cảnh cáo WARNING, nếu quan trọng thì hiện hộp thoại ERROR và thoát phần mềm.
- Khi ngoại lệ xảy ra, luôn cố gắng tìm cách Recover, không để chương trình thoát mà phải tiếp tục chạy
- Logger ghi lại các thông tin info, warning, error quan trọng

VII. Giao diện GUI

Dự án này sử dụng thư viện Java Swing cơ bản có sẵn trong Java để xây dựng giao diện GUI, nhưng không dùng các Linh kiện Component sẵn có như Button hay TextField v.v… của Swing, nhóm đã thực hiện vẽ tùy chỉnh thủ công đơn giản.

1. Lớp Game kế thừa JPanel và sở hữu JFrame

JPanel Game là linh kiện hình thấy duy nhất trên JFrame.
Hàm paintComponent(Graphics G) kế thừa từ JPanel sẽ được Override và biến G được truyền cho Màn Hình, Pacman, Bản đồ… để vẽ thủ công trên chính JPanel Game đó

Tất cả bắt đầu từ hàm paintComponent(Graphics G) trong lớp Game.java. Đây là nơi duy nhất giao tiếp trực tiếp với hệ thống hiển thị của Java.

2. Interface Paintable

Dự án sử dụng một Interface tên là Paintable chỉ chứa duy nhất một hàm:

void repaint(Graphics2D G).

Mục đích thiết kế:
Bất kỳ lớp nào muốn hiện lên màn hình đều implements interface này, nhằm đồng bộ hóa code trong dự án và nhấn mạnh rằng Lớp đó có thể vẽ ra màn hình.

3. Kiến trúc Phân tầng
Việc vẽ được tổ chức theo mô hình cây hay phân tầng, sử dụng tính đa hình để ủy quyền việc vẽ xuống các cấp thấp hơn.

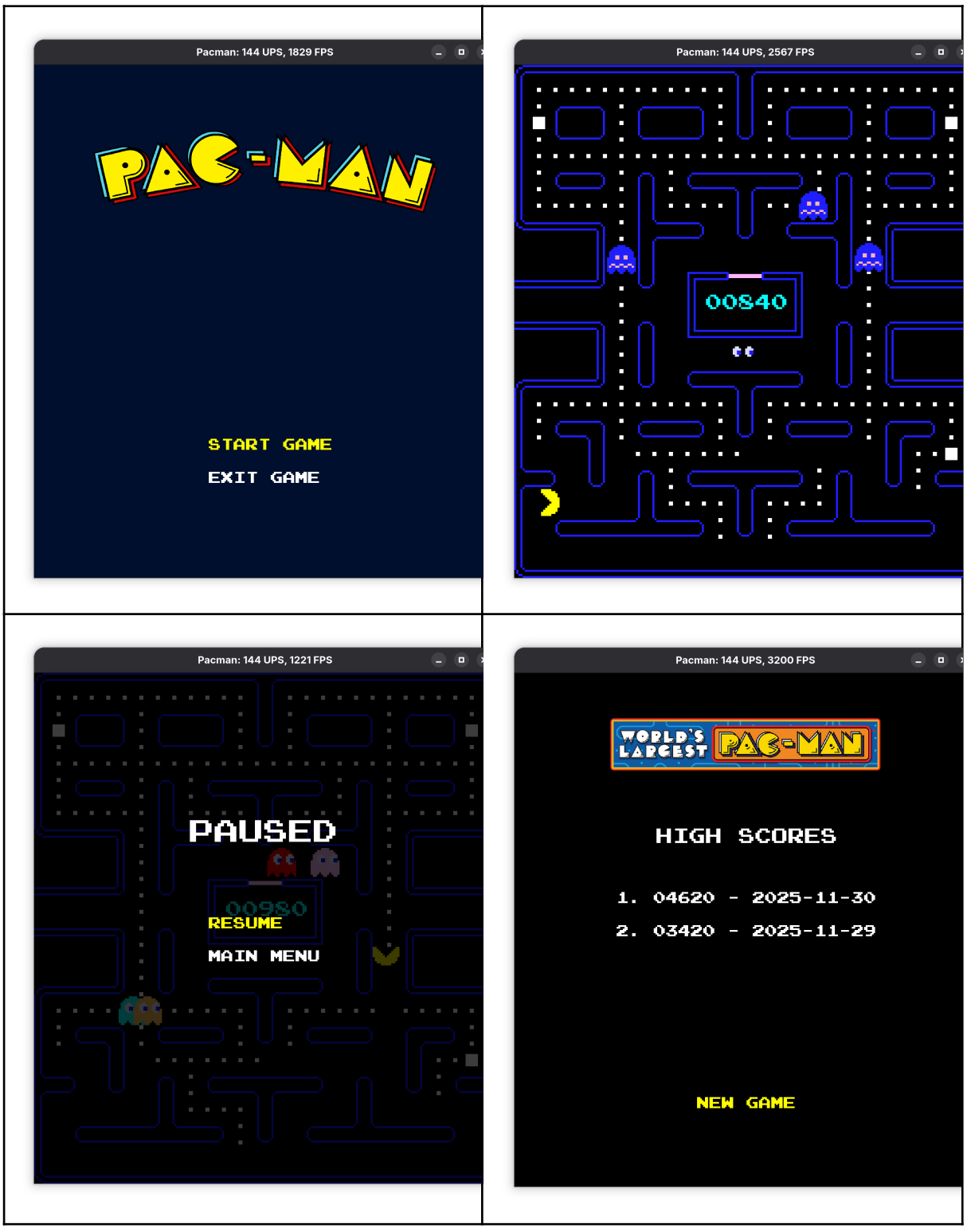
Quy trình vẽ một khung hình diễn ra như sau:

Cấp 1: Game.java:
+ Thực hiện phóng to toàn bố cục tranh (hàm G2D.scale()) để game luôn hiển thị đúng tỷ lệ dù cửa sổ to hay nhỏ.
+ Chuyển cọ vẽ Graphics2D cho ScreenManager vẽ tiếp.

Cấp 2: ScreenManager.java:
Lớp chuyển cọ vẽ cho màn hình hiện tại mà nó đang quản lý: current.repaint(G).

Cấp 3 - Màn hình cụ thể ví dụ Playing.java:
Đây là nơi quy định thứ tự vẽ để tạo ra các lớp:
Vẽ Bản đồ ở dưới cùng.
Vẽ Điểm số sau đó.
Vẽ Pacman.
Vẽ các con Ma.

Cấp 4 - Entity và Map:
Cuối cùng, các lớp như Pacman, Ghost, Map mới thực sự thực hiện lệnh vẽ ảnh (drawImage) từ Spritesheet lên màn hình tại toạ độ của chúng.



Hướng Dẫn Điều khiển:

- **Di chuyển:** phím W A S D hoặc phím mũi tên ← ↑ → ↓
- **Dừng Game khi đang chơi:** phím Esc
- **Lựa chọn:** Enter