ĐẠI HỌC BÁCH KHOA HÀ NỘI

Báo cáo bài tập lớn
Lập trình Hướng đối tượng IT3103

Monster Chase: Coin Collector

Nhóm 7.3

Danh sách thành viên:

Lê Khánh Linh	20225731
Đỗ Hoàng Minh Hiếu	20225837

Giảng viên hướng dẫn: PGS. TS. Lê Đức Hậu

HÀ NỘI, 12/2024

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong thời đại phát triển mạnh mẽ của công nghệ, các trò chơi điện tử đã trở thành một phần không thể thiếu trong đời sống giải trí của con người. Sự phát triển của trò chơi điện tử không chỉ dựa vào yếu tố giải trí mà còn phụ thuộc vào tính tương tác, độ phức tạp và khả năng mở rộng. Để hiện thực hóa được những điều này, lập trình hướng đối tượng (Object-Oriented Programming - OOP) là một trong những phương pháp thiết kế và phát triển trò chơi phổ biến, giúp tạo ra các cấu trúc phần mềm có tổ chức và dễ quản lý.

Trong bài báo cáo này, chúng tôi sẽ phân tích quá trình phát triển một trò chơi điện tử có các tính năng phức tạp như di chuyển, chiến đấu, quản lý kho đồ, vượt qua chướng ngại vật và đối mặt với kẻ thù. Trò chơi áp dụng mô hình lập trình hướng đối tượng để đảm bảo tính dễ mở rộng và tái sử dụng mã lệnh, đồng thời tạo ra một trải nghiệm người chơi mượt mà và cuốn hút.

1.2 Giới thiệu về trò chơi

Trò chơi được xây dựng với lối chơi phiêu lưu hành động, nơi người chơi sẽ nhập vai vào một nhân vật và đối đầu với nhiều loại quái khác nhau trong môi trường có nhiều chướng ngại vật. Người chơi có thể di chuyển theo 4 hướng, tấn công quái vật, quản lý kho đồ và tương tác với các vật phẩm cũng như môi trường. Trò chơi cũng có cơ chế lưu và tải trạng thái, cho phép người chơi tiếp tục từ điểm đã dừng.

Một trong những điểm đặc sắc của trò chơi là hệ thống kẻ thù phong phú, từ quái vật thông thường, quái vật tinh khôn có khả năng đuổi theo người chơi, cho đến boss với nhiều giai đoạn chiến đấu khác nhau. Điều này đòi hỏi trò chơi phải có khả năng xử lý các hành vi phức tạp, tạo ra một thế giới game đa dạng và thú vị.

Cách thức thực hiện trò chơi trên sẽ được đề cập ở các chương sau.

1.3 Mục tiêu của trò chơi

Từ việc xác định rõ nhiệm vụ cần giải quyết ở phần 1.2, mục tiêu chính của trò chơi là tạo ra một trải nghiệm người chơi phong phú, nơi người chơi có thể tự do khám phá, tương tác với môi trường, chiến đấu với kẻ thù, và quản lý tài nguyên của mình. Việc sử dụng lập trình hướng đối tượng cho phép mở rộng trò chơi dễ dàng bằng cách thêm vào các tính năng mới, nâng cao trí tuệ nhân tạo của kẻ thù, hoặc cải thiện trải nghiệm người chơi mà không cần phải thay đổi toàn bộ cấu trúc phần mềm.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

2.1 Tổng quan

2.1.1 Yêu cầu bài toán

- Người chơi điều khiển một nhân vật trong một bản đồ được sinh ngẫu nhiên vị trí của các vật thể.
- Nhân vật người chơi điều khiển có chỉ số xác định lượng máu.
- Người chơi sẽ thua nếu bị quái vật bắt được hoặc đạp bẫy nhiều hơn số mạng hiện có

Đáp ứng yêu cầu bài toán bao gồm:

- Thể hiện bản đồ và các đối tượng trên bản đồ trên giao diện đồ họa
- Điều khiển nhân vật di chuyển
- Quái vật có khả năng di chuyển và đuổi theo người chơi
- Xử lý được các sự kiện xảy ra khi tác động vào các vật thể trên bản đồ.

2.1.2 Các chức năng chính

a, Quản lý nhân vật

- Người chơi (Player) là trung tâm của trò chơi. Người chơi có thể điều khiển nhân vật của mình để di chuyển qua các ô trong bản đồ.
- Các quái vật (Monster) đóng vai trò chướng ngại vật, xuất hiện để truy đuổi hoặc gây khó khăn cho người chơi. Chúng có thể di chuyển theo logic riêng và va chạm với người chơi.

Chức năng này bao gồm:

- Di chuyển nhân vật: Nhân vật của người chơi di chuyển bằng cách sử dụng các phím điều khiển.
- Va chạm: Xử lý các va chạm giữa người chơi và quái vật, hoặc giữa người chơi và vật phẩm.

b, Quản lý vật phẩm và đối tượng

Các vật phẩm và đối tượng trong game bao gồm:

- Tiền xu (Coin): Khi thu thập, người chơi nhận được điểm thưởng.
- Bẫy (Trap): Làm giảm máu người chơi nếu người chơi chạm vào.
- Tường (Wall): Ngăn chặn đường đi của người chơi và quái vật.

Hệ thống đảm bảo các vật phẩm được tạo ngẫu nhiên tại các vị trí phù hợp trên bản đồ.

Chức năng này bao gồm việc quản lý tương tác: Kiểm tra các sự kiện khi người chơi tương tác với các đối tượng này.

c, Hệ thống điểm số

Ghi nhận và quản lý điểm số của người chơi thông qua:

- Điểm số hiện tại: Tăng điểm khi thu thập vật phẩm.
- Điểm số cao nhất: Lưu điểm cao nhất đạt được qua các lần chơi.

Chức năng bao gồm:

- Hiển thị điểm số: Thông qua TopScorePanel, người chơi có thể xem điểm số hiện tại và điểm cao nhất.
- Lưu trữ điểm số: Sử dụng ScoreService để lưu điểm cao nhất

d, Hiển thị và cập nhật giao diện

Trò chơi sử dụng GameRenderer để hiển thị tất cả các đối tượng trên màn hình:

- Nhân vật.
- Quái vật.
- Các vật phẩm và chướng ngại vật.
- Các cập nhật về trạng thái trò chơi cũng được phản ánh trực tiếp lên giao diện.

Chức năng bao gồm:

- Vẽ bản đồ: Hiển thị các ô và các đối tượng trong trò chơi.
- Cập nhật giao diện: Thay đổi giao diện theo thời gian thực khi người chơi thực hiện các hành động.

e, Quản lý logic trò chơi

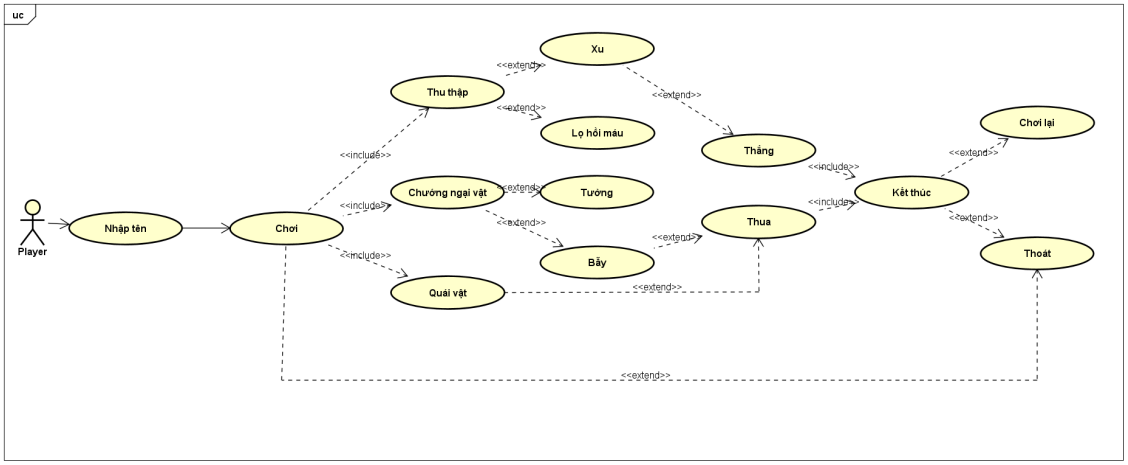
Điều khiển luồng hoạt động chính của trò chơi:

- Sinh ngẫu nhiên quái vật và vật phẩm thông qua GameObjectSpawner.
- Xử lý va chạm giữa các đối tượng trong trò chơi. Đảm bảo logic va chạm được kiểm tra chính xác.
- Quản lý trạng thái trò chơi: Dừng trò chơi khi người chơi thắng hoặc thua.

2.2 Biểu đồ use case tổng quát

Biểu đồ Use Case trong Hình **??** mô tả các chức năng chính và mối quan hệ giữa tác nhân *Player* với hệ thống. Các chức năng bao gồm:

- Nhập tên người chơi.
- Thực hiện các hành động chơi, thu thập vật phẩm, né tránh chướng ngại vật.
- Kết thúc trò chơi qua các trạng thái thắng, thua, hoặc kết thúc.
- Lựa chọn chơi lại hoặc thoát khỏi trò chơi.



2.3 Đặc tả chức năng

Use Case: Nhập tên

- **Tác nhân:** Người chơi.
- **Mô tả:** Người chơi nhập tên trước khi bắt đầu trò chơi.
- **Luồng công việc chính:**
  1. Hệ thống hiển thị màn hình yêu cầu nhập tên.
  2. Người chơi nhập tên của mình.
  3. Hệ thống lưu lại tên và chuyển sang màn hình chơi.
- **Điều kiện tiên quyết:** Trò chơi khởi động thành công.
- **Kết quả:** Hệ thống lưu lại tên người chơi.

Use Case: Chơi

- **Tác nhân:** Người chơi.
- **Mô tả:** Người chơi điều khiển nhân vật di chuyển trong bản đồ để thu thập vật phẩm và né tránh chướng ngại vật.
- **Luồng công việc chính:**
  1. Người chơi bắt đầu di chuyển nhân vật.
  2. Người chơi thu thập các vật phẩm như tiền xu và lọ hồi máu.
  3. Người chơi né tránh quái vật và bẫy.

4. Hệ thống cập nhật điểm số và trạng thái trò chơi theo thời gian thực.
5. Trò chơi kết thúc khi người chơi thắng hoặc thua.

- **Điều kiện tiên quyết:** Người chơi đã nhập tên thành công.
- **Kết quả:** Trạng thái trò chơi được xác định (thắng, thua hoặc kết thúc).

Use Case: Thắng

- **Tác nhân:** Người chơi.
- **Mô tả:** Người chơi thắng khi đạt điều kiện cụ thể (như thu thập đủ vật phẩm hoặc hoàn thành thời gian chơi).
- **Luồng công việc chính:**
  1. Hệ thống kiểm tra điều kiện thắng trong suốt quá trình chơi.
  2. Khi đạt điều kiện, hệ thống thông báo trạng thái "Thắng".
  3. Hiển thị điểm số và tùy chọn chơi lại hoặc thoát.
- **Điều kiện tiên quyết:** Người chơi đang trong trạng thái "Chơi".
- **Kết quả:** Người chơi nhận thông báo chiến thắng.

Use Case: Thua

- **Tác nhân:** Người chơi.
- **Mô tả:** Người chơi thua khi chạm phải chướng ngại vật hoặc không đạt được mục tiêu.
- **Luồng công việc chính:**
  1. Hệ thống kiểm tra các điều kiện thua (chạm bẫy, bị quái vật tấn công, hoặc hết thời gian).
  2. Khi một điều kiện thua xảy ra, hệ thống thông báo trạng thái "Thua".
  3. Hiển thị điểm số và tùy chọn chơi lại hoặc thoát.
- **Điều kiện tiên quyết:** Người chơi đang trong trạng thái "Chơi".
- **Kết quả:** Người chơi nhận thông báo thất bại.

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Trong dự án "Monster Chase: Coin Collector", các công nghệ chính được sử dụng bao gồm Java và Java Swing. Các công nghệ này được lựa chọn vì tính linh hoạt, dễ sử dụng, và khả năng hỗ trợ phát triển ứng dụng đồ họa trên nền tảng máy tính cá nhân. Dưới đây là mô tả chi tiết về các công nghệ và lý do chọn chúng:

3.1 Java

Java là ngôn ngữ lập trình hướng đối tượng phổ biến, mạnh mẽ và có tính tương thích cao, được sử dụng rộng rãi trong phát triển phần mềm và ứng dụng. Dưới đây là những lý do lựa chọn Java:

- Hỗ trợ đa nền tảng: Java có khả năng chạy trên bất kỳ hệ điều hành nào có cài đặt Java Runtime Environment (JRE), giúp ứng dụng có thể được triển khai trên nhiều nền tảng mà không cần thay đổi mã nguồn.
- Hệ sinh thái phong phú: Java cung cấp một loạt các thư viện và công cụ hỗ trợ phát triển ứng dụng, từ việc xử lý dữ liệu, giao diện người dùng, đến quản lý tài nguyên hệ thống.
- Quản lý bộ nhớ tự động: Java có tính năng garbage collection (quản lý bộ nhớ tự động), giúp giảm bớt các vấn đề về quản lý bộ nhớ và tăng tính ổn định của ứng dụng.
- Môi trường phát triển mạnh mẽ: Các IDE như IntelliJ IDEA, Eclipse, và NetBeans hỗ trợ phát triển Java, giúp quá trình lập trình trở nên hiệu quả và dễ dàng hơn.

Trong dự án này, Java được sử dụng để xây dựng các thành phần logic của trò chơi, bao gồm quản lý trạng thái người chơi, các đối tượng trong trò chơi (coin, quái vật, bẫy...), và xử lý các sự kiện như di chuyển của người chơi và tương tác với các đối tượng trong trò chơi.

3.2 Java Swing

Java Swing là một thư viện GUI (giao diện người dùng đồ họa) được tích hợp sẵn trong Java, giúp phát triển các ứng dụng desktop với giao diện đồ họa. Swing cung cấp các thành phần GUI như cửa sổ, nút bấm, bảng điều khiển, thanh cuộn, v.v. để xây dựng giao diện người dùng một cách dễ dàng và hiệu quả. Dưới đây là lý do chọn Java Swing:

- Dễ sử dụng và linh hoạt: Swing cung cấp các thành phần GUI có thể dễ dàng tùy chỉnh và mở rộng. Điều này giúp việc thiết kế giao diện trò chơi trở nên

đơn giản và dễ dàng hơn.

- Tính tương thích cao: Swing là một phần của thư viện chuẩn Java, không yêu cầu bất kỳ thư viện bên ngoài nào, giúp giảm thiểu sự phức tạp khi triển khai và đảm bảo tính tương thích cao giữa các nền tảng.
- Được hỗ trợ rộng rãi: Swing có tài liệu phong phú và cộng đồng phát triển lớn, giúp các lập trình viên dễ dàng tìm kiếm sự hỗ trợ và học hỏi khi gặp phải vấn đề trong quá trình phát triển.

Java Swing được sử dụng trong dự án để xây dựng giao diện người dùng của trò chơi, bao gồm các yếu tố như:

- Cửa sổ trò chơi: Hiển thị bản đồ, các đối tượng (coin, quái vật, bẫy...), thông tin người chơi (tên, HP, số coin).
- Các điều khiển người chơi: Các nút điều khiển di chuyển nhân vật, bắt đầu lại trò chơi, thoát trò chơi.
- Cập nhật trạng thái: Cập nhật thông tin trạng thái của trò chơi, bao gồm điểm số, kết quả thắng thua.

3.3 Mối quan hệ giữa Java và Java Swing

Java cung cấp nền tảng để viết logic xử lý và quản lý các trạng thái trong trò chơi, trong khi Java Swing hỗ trợ xây dựng giao diện người dùng, giúp người chơi có thể tương tác với trò chơi dễ dàng. Các đối tượng và logic trong Model (như người chơi, quái vật, coin, v.v.) được xử lý trong Java, sau đó hiển thị qua các thành phần giao diện của Swing. Swing cũng chịu trách nhiệm nhận các đầu vào từ người chơi (như nhấn phím hoặc chuột) và chuyển chúng đến Controller để xử lý.

3.4 Lý do lựa chọn công nghệ

- Java và Java Swing là lựa chọn tối ưu cho các ứng dụng đồ họa không yêu cầu hiệu suất quá cao (như các trò chơi nhỏ hoặc ứng dụng giáo dục), giúp phát triển nhanh chóng và hiệu quả mà không cần quá nhiều cấu hình phức tạp.
- Các công cụ phát triển như IntelliJ IDEA, Eclipse giúp hỗ trợ việc lập trình và debug dễ dàng, tiết kiệm thời gian trong quá trình phát triển.
- Swing giúp tạo giao diện trực quan và dễ sử dụng mà không cần tích hợp thêm các thư viện bên ngoài, giúp giảm thiểu độ phức tạp của hệ thống. Với sự kết hợp giữa Java và Java Swing, dự án trò chơi này không chỉ dễ phát triển mà còn dễ bảo trì và mở rộng trong tương lai.

CHƯƠNG 4. THIẾT KẾ VÀ ĐÁNH GIÁ HỆ THỐNG

4.1 Thiết kế chi tiết

4.1.1 Thiết kế giao diện

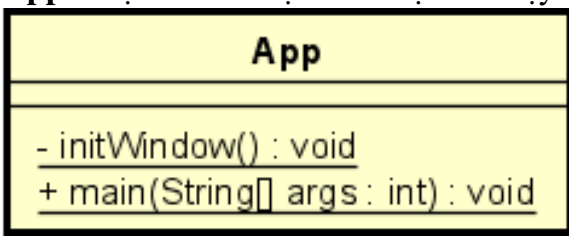
Giao diện của hệ thống được thiết kế dựa trên nguyên tắc thân thiện với người dùng và tối ưu hoá trải nghiệm người dùng. Các thành phần giao diện chính bao gồm:

- Menu chính: Bao gồm các tuỳ chọn để bắt đầu trò chơi, xem bảng xếp hạng, và thoát khỏi trò chơi.
- Giao diện chơi: Hiển thị nhân vật, các vật phẩm, chướng ngại vật và các thông tin như điểm số, thời gian còn lại.
- Giao diện kết thúc: Thông báo thắng/thua và tuỳ chọn chơi lại hoặc thoát.

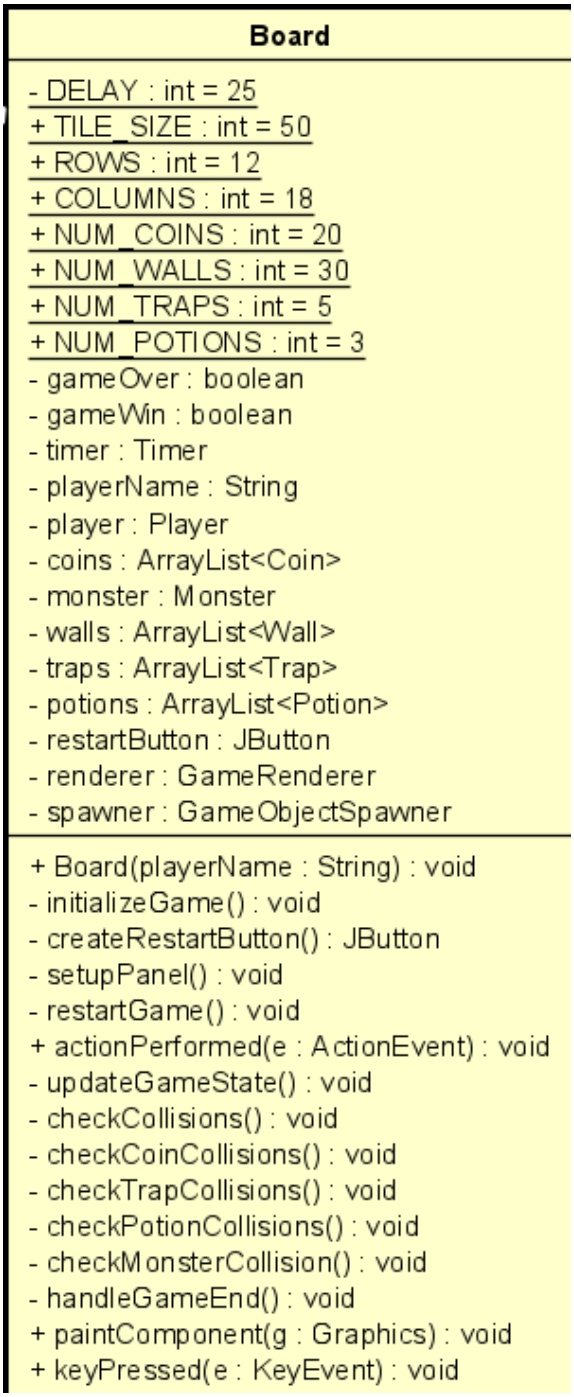
4.1.2 Thiết kế lớp

Hệ thống được chia thành các lớp chính để đảm bảo tính module hoá và dễ bảo trì. Các lớp bao gồm:

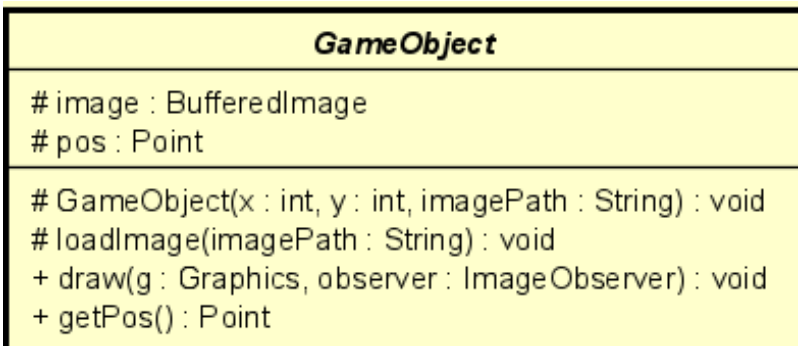
- **App**: Chịu trách nhiệm khởi tạo và chạy trò chơi.



- **Board**: Quản lý trạng thái của các đối tượng trên màn chơi như người chơi, vật phẩm, và chướng ngại vật.

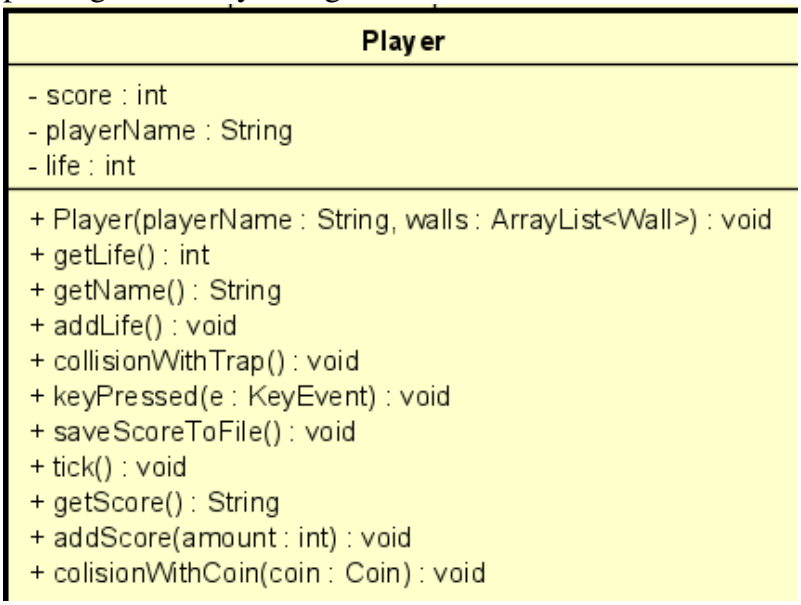


- **GameObject**: Lớp cha của tất cả các đối tượng trong trò chơi, bao gồm người chơi, quái vật, xu, lọ hồi máu, tường, và bẫy.

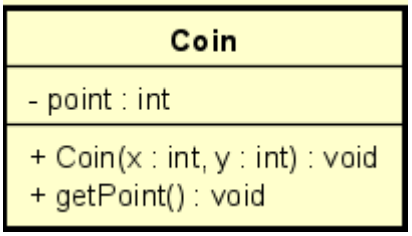


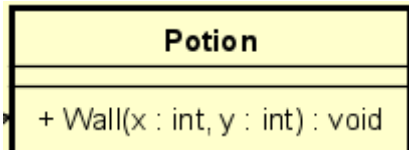
- **Player**: Đại diện cho người chơi, bao gồm các thuộc tính như điểm số và các

phương thức xử lý tương tác.

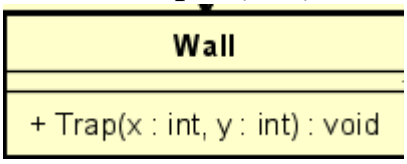


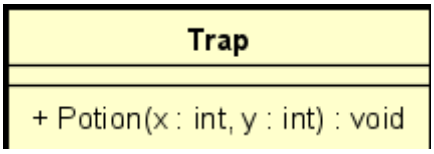
- **Coin** và **Potion**: Đại diện cho các vật phẩm có thể thu thập trong trò chơi.



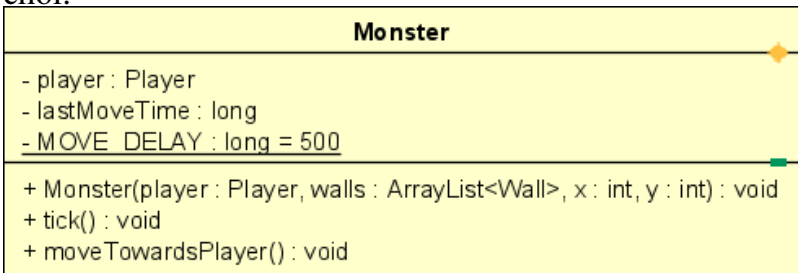


- **Wall** và **Trap**: Đại diện cho các chướng ngại vật trong trò chơi.





- **Monster**: Đại diện cho các quái vật, có hành vi di chuyển và tấn công người chơi.



4.2 Xây Dựng Mô Hình Lớp

4.2.1 Các Lớp Chính

Các lớp chính bao gồm:

- **App**: Xử lý vòng đời của trò chơi.
- **Board**: Điều phối các đối tượng và xử lý logic chính của trò chơi.

- **Player**: Có các phương thức như di chuyển, thu thập vật phẩm, và xử lý va chạm với chướng ngại vật hoặc quái vật.
- **GameObject**: Là lớp cơ sở của tất cả các đối tượng như **Wall**, **Trap**, **Coin**, **Potion**, **Monster**.
- **Monster**: Được thiết kế để tương tác và gây cản trở cho người chơi.

4.2.2 Quan Hệ Các Lớp

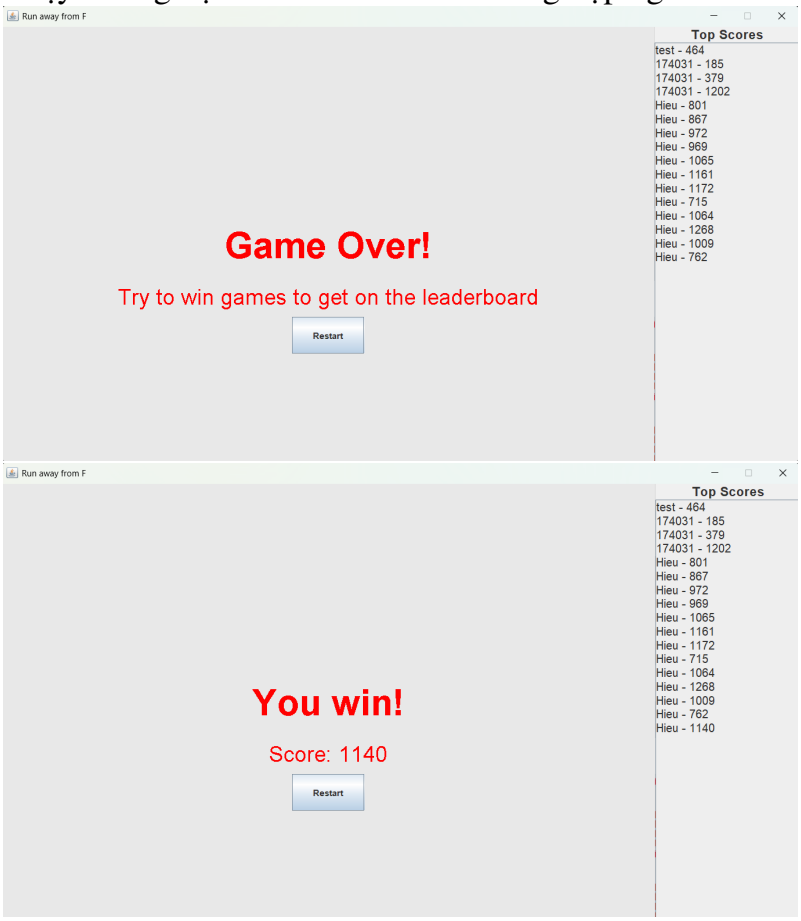
Quan hệ giữa các lớp được xây dựng như sau:

- **Board** điều phối và quản lý tất cả các đối tượng trên màn chơi, bao gồm **Player**, **Monster**, **Coin**, **Potion**, **Wall**, và **Trap**.
- **Player** tương tác với các lớp **Coin** và **Potion** để thu thập vật phẩm, đồng thời xử lý va chạm với **Wall**, **Trap**, và **Monster**.
- **Monster** tương tác với **Player** để thực hiện các hành vi tấn công hoặc cản trở.

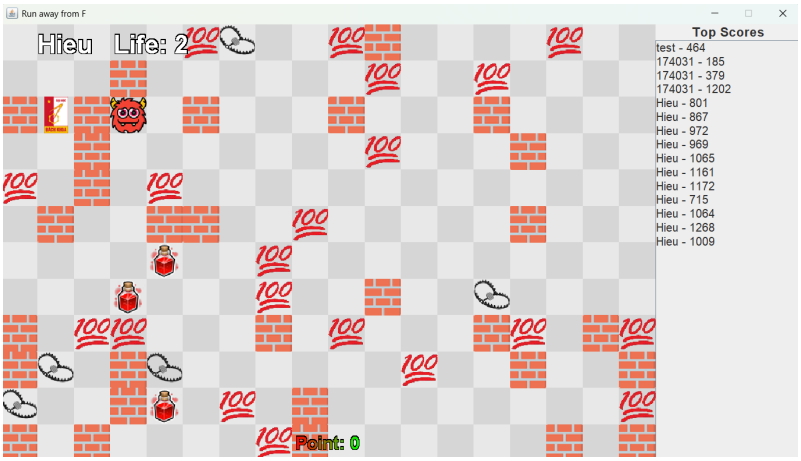
4.3 Demo Chương Trình

Phần demo chương trình bao gồm:

- Chạy thử nghiệm trò chơi với các trường hợp người chơi thắng, thua.

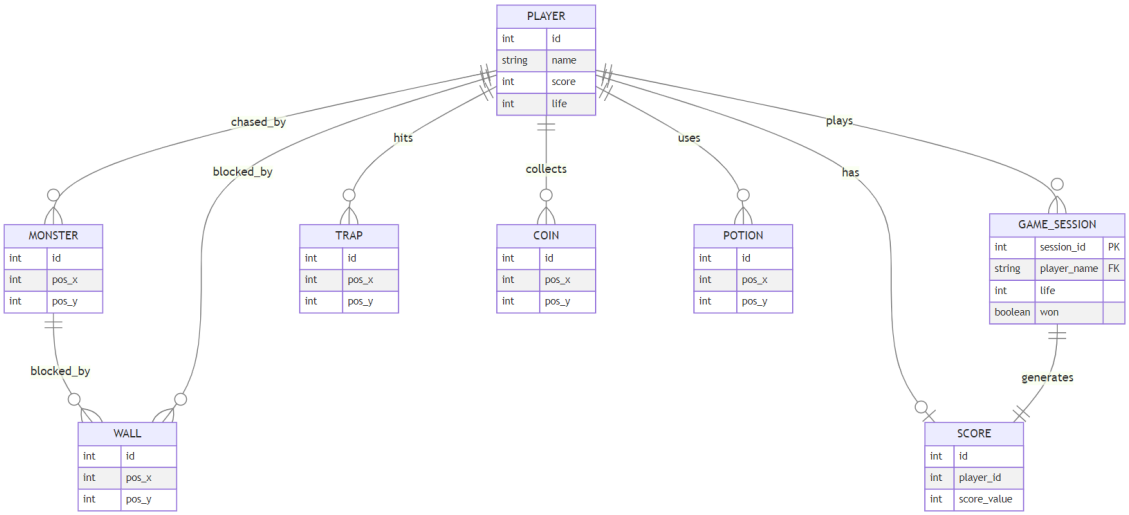


- Hiển thị giao diện, bao gồm các yếu tố động như di chuyển nhân vật, xuất hiện vật phẩm và chướng ngại vật.


Run away from F
Hieu
Life: 2
Point: 0
Top Scores
test - 464
174031 - 185
174031 - 379
174031 - 1202
Hieu - 801
Hieu - 867
Hieu - 972
Hieu - 969
Hieu - 1065
Hieu - 1161
Hieu - 1172
Hieu - 715
Hieu - 1064
Hieu - 1268
Hieu - 1009

CHƯƠNG 5. SƠ ĐỒ THỰC THỂ LIÊN KẾT

5.1 Các Thực Thể Chính



- **Thực thể Player**:
  - **name**: Tên của người chơi.
  - **score**: Điểm số hiện tại của người chơi trong trò chơi (có thể được cập nhật qua nhiều phiên chơi).
  - **life**: Số mạng còn lại của người chơi (dùng để theo dõi trạng thái của người chơi trong quá trình chơi).
- **Thực thể Game Session**:
  - **session_id**: Mã định danh duy nhất của một phiên chơi.
  - **player_name**: Liên kết đến tên của người chơi tham gia phiên chơi này.
  - **life**: Số mạng còn lại của người chơi trong phiên chơi này.
  - **won**: Xác định người chơi thắng hay thua.
- **Thực thể Score**:
  - **id**: Mã định danh duy nhất của điểm số.
  - **player_id**: Liên kết đến Player, cho biết điểm số này thuộc về người chơi nào.
  - **score_value**: Giá trị điểm mà người chơi đạt được trong trò chơi hoặc phiên chơi.
- **Thực thể Monster**:

  - **id**: Mã định danh duy nhất của quái vật.
  - **pos_x, pos_y**: Vị trí của quái vật.
- **Thực thể Wall**:
  - **id**: Mã định danh duy nhất của tường.
  - **pos_x, pos_y**: Vị trí của tường.
- **Thực thể Trap**:
  - **id**: Mã định danh duy nhất của bẫy.
  - **pos_x, pos_y**: Vị trí của bẫy.
- **Thực thể Coin**:
  - **id**: Mã định danh duy nhất của đồng xu.
  - **pos_x, pos_y**: Vị trí của đồng xu.
- **Thực thể Potion**:
  - **id**: Mã định danh duy nhất của thuốc.
  - **pos_x, pos_y**: Vị trí của thuốc.

5.2 Quan Hệ Giữa Các Thực Thể

- **PLAYER ||–o COIN: collects**
  - Một Player có thể thu thập nhiều Coin trong trò chơi.
  - Quan hệ này cho thấy rằng mỗi người chơi có thể thu thập nhiều đồng xu khác nhau trong quá trình chơi.
- **PLAYER ||–o{ TRAP: hits**
  - Một Player có thể dính phải nhiều Trap trong trò chơi.
  - o Đây là quan hệ mô tả việc người chơi có thể gặp nhiều bẫy, dẫn đến giảm mạng hoặc ảnh hưởng tiêu cực.
- PLAYER ||–o{ POTION : uses:
  - Một Player có thể sử dụng nhiều Potion (thuốc) trong trò chơi.
  - Quan hệ này cho thấy rằng người chơi có thể sử dụng nhiều loại thuốc để phục hồi sức khỏe hoặc tăng cường các khả năng.
- PLAYER ||–o{ MONSTER : chased_by
  - Một Player có thể bị quái vật truy đuổi.
  - o Quan hệ này mô tả tình huống người chơi bị quái vật đuổi theo trong

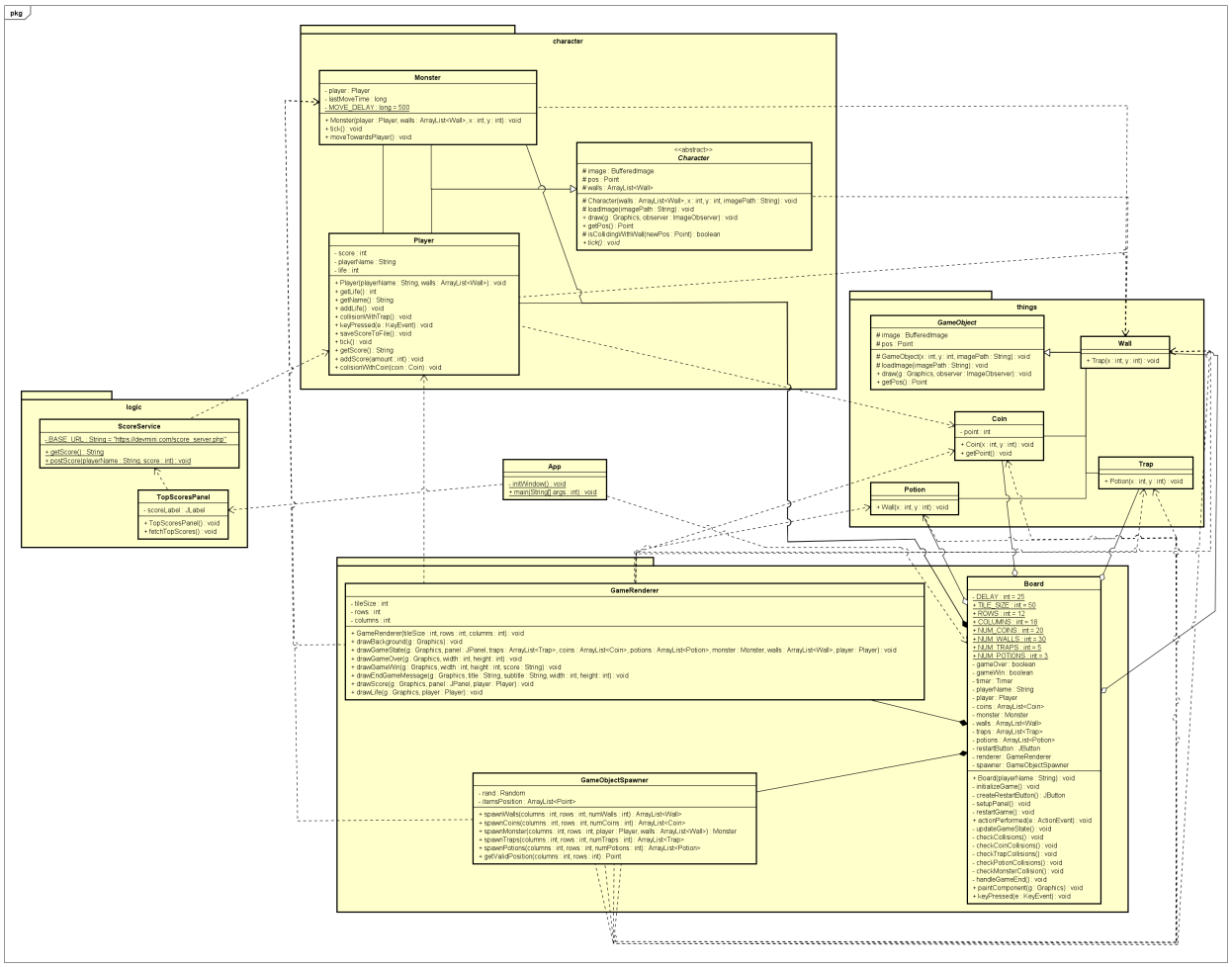
quá trình chơi.

- MONSTER ||–o{ WALL : blocked_by:
  - o Một Monster có thể bị chặn bởi Wall (tường).
  - o Điều này mô tả rằng các quái vật có thể bị tường ngăn cản hoặc chặn lại trong quá trình di chuyển trên bản đồ.
- PLAYER ||–o{ WALL : blocked_by
  - Một Player có thể bị chặn bởi Wall (tường).
  - Quan hệ này chỉ ra rằng người chơi cũng có thể bị tường cản trở trong trò chơi.
- PLAYER ||–o{ SCORE : has
  - Một Player có thể có một hoặc nhiều Score.
  - Quan hệ này cho thấy rằng mỗi người chơi có thể có nhiều điểm số trong trò chơi, nhưng mỗi điểm số liên kết với một người chơi duy nhất.
- GAME_SESSION ||–|| SCORE : generates:
  - Một Game Session tạo ra một Score duy nhất.
  - Mỗi phiên chơi sẽ ghi nhận một điểm số, và điểm số này liên kết trực tiếp với phiên chơi đó.
- PLAYER ||–o{ GAME_SESSION : plays:
  - Một Player có thể tham gia vào nhiều Game Session.
  - Quan hệ này cho thấy rằng mỗi người chơi có thể chơi nhiều phiên chơi khác nhau, mỗi phiên lại có thể có một trạng thái thắng hay thua khác nhau.

Biểu đồ ER này giúp theo dõi lịch sử chơi game của người chơi, Lưu trữ và quản lý điểm số, Phân tích hiệu suất người chơi qua các phiên, Xây dựng bảng xếp hạng điểm cao

CHƯƠNG 6. BIỂU ĐỒ LỚP

Phần 4.1.2 đã trình bày những lớp chính của chương trình này. Do đó tại phần này, chúng ta sẽ đi chi tiết về các lớp của game.



6.1 Chi Tiết Các Lớp

Phần này trình bày chi tiết về các lớp chưa được đề cập hoặc mở rộng thêm nội dung từ phần 4.2.

6.1.1 Lớp Coin

Lớp **Coin** đại diện cho các đồng xu trong trò chơi mà người chơi có thể thu thập để tăng điểm số.

- **Thuộc tính:**
  - `id`: Mã định danh duy nhất cho từng đồng xu.
  - `posX`, `posY`: Tọa độ vị trí của đồng xu trên màn hình.
- **Phương thức:**

- `getCollectedByPlayer()`: Xử lý khi người chơi thu thập đồng xu.

6.1.2 Lớp Potion

Lớp **Potion** đại diện cho các lọ thuốc mà người chơi có thể sử dụng để hồi phục sức khỏe hoặc tăng cường khả năng.

- **Thuộc tính:**
  - `id`: Mã định danh duy nhất của lọ thuốc.
  - `posX`, `posY`: Tọa độ vị trí của lọ thuốc trên màn hình.
  - `effect`: Loại hiệu ứng của lọ thuốc (ví dụ: hồi máu, tăng tốc độ, v.v.).
- **Phương thức:**
  - `applyEffect()`: Áp dụng hiệu ứng của lọ thuốc cho người chơi.

6.1.3 Lớp Wall

Lớp **Wall** đại diện cho các bức tường trên màn chơi, đóng vai trò cản trở di chuyển của người chơi và quái vật.

- **Thuộc tính:**
  - `id`: Mã định danh duy nhất của tường.
  - `posX`, `posY`: Tọa độ vị trí của tường.

6.1.4 Lớp Trap

Lớp **Trap** đại diện cho các bẫy có thể gây hại cho người chơi.

- **Thuộc tính:**
  - `id`: Mã định danh duy nhất của bẫy.
  - `posX`, `posY`: Tọa độ vị trí của bẫy trên màn hình.
  - `damage`: Lượng sát thương gây ra cho người chơi.
- **Phương thức:**
  - `trigger()`: Kích hoạt bẫy khi người chơi va chạm.

6.1.5 Lớp Monster

Lớp **Monster** đại diện cho các quái vật trong trò chơi, có hành vi cản trở và truy đuổi người chơi.

- **Thuộc tính:**
  - `id`: Mã định danh duy nhất của quái vật.
  - `posX`, `posY`: Tọa độ vị trí của quái vật.

  - `behaviorPattern`: Mẫu hành vi của quái vật (ví dụ: tuần tra, truy đuổi người chơi).

- **Phương thức:**
  - `move()`: Di chuyển quái vật theo mẫu hành vi đã định.
  - `attackPlayer()`: Thực hiện hành vi tấn công người chơi.

CHƯƠNG 7. KIẾN TRÚC HỆ THỐNG

Trò chơi "Monster Chase: Coin Collector" được xây dựng dựa trên mô hình MVC (Model-View-Controller), một mô hình thiết kế phần mềm giúp phân tách rõ ràng giữa các thành phần quản lý dữ liệu, giao diện, và điều khiển. Với việc không sử dụng server và cơ sở dữ liệu, toàn bộ logic và trạng thái trò chơi được xử lý trực tiếp trên máy cục bộ.

1. Mô hình (Model)
    - Vai trò: Quản lý toàn bộ dữ liệu và trạng thái của trò chơi.
    - Thành phần chính:
        - Trạng thái người chơi: Tên nhân vật, vị trí trên bản đồ, số lượng coin đã thu thập, điểm HP hiện tại.
        - Trạng thái môi trường: Vị trí các đối tượng như coin, tường, bẫy, potion, và quái vật.
        - Quy tắc trò chơi: Điều kiện thắng (thu thập tất cả coin) và điều kiện thua (mất hết HP hoặc bị quái vật bắt).
    - Lưu trữ: Dữ liệu được lưu trữ trong bộ nhớ dưới dạng các biến và cấu trúc dữ liệu (mảng, đối tượng).
2. Hiển thị (View)
    - Vai trò: Hiển thị trạng thái của trò chơi lên màn hình giao diện người dùng.
    - Chức năng chính:
        - Render bản đồ trò chơi với các yếu tố như coin, quái vật, tường, bẫy, và potion.
        - Hiển thị trạng thái người chơi như điểm HP còn lại, số coin đã thu thập.
        - Thông báo kết quả trò chơi khi người chơi thắng hoặc thua.
    - Kỹ thuật: View được cập nhật mỗi khi trạng thái trong Model thay đổi, đảm bảo hiển thị nhất quán và đúng thời điểm.
3. Điều khiển (Controller)
    - Vai trò: Nhận và xử lý đầu vào từ người dùng, điều chỉnh dữ liệu trong Model và thông báo cho View cập nhật giao diện.
    - Thành phần chính:

  - Xử lý các thao tác di chuyển của Character (Người chơi và quái vật theo các hướng lên, xuống, trái, phải.
  - Kiểm tra tương tác giữa người chơi và các đối tượng trên bản đồ (thu thập coin, dẫm phải bẫy, hồi máu, hoặc bị quái vật bắt).
- Lưu trữ: Dữ liệu được lưu trữ trong bộ nhớ dưới dạng các biến và cấu trúc dữ liệu (mảng, đối tượng).

4. Cách phối hợp
   - Khi người chơi thực hiện hành động (nhấn phím di chuyển), Controller sẽ nhận tín hiệu đầu vào, cập nhật trạng thái trong Model, và yêu cầu View hiển thị trạng thái mới.
   - Model đóng vai trò trung tâm, nơi lưu trữ mọi thông tin, và các thay đổi trong Model sẽ được truyền đến View một cách tự động hoặc thông qua các sự kiện do Controller kích hoạt.

Với kiến trúc này, trò chơi đạt được sự tách biệt rõ ràng giữa các thành phần, giúp code dễ bảo trì, dễ mở rộng và quản lý hiệu quả hơn.

CHƯƠNG 8. CÁC CÔNG NGHỆ HƯỚNG ĐỐI TƯỢNG ĐÃ SỬ DỤNG

8.1 Tính đóng gói

Tính đóng gói đề cập đến việc giới hạn truy cập vào dữ liệu bên trong đối tượng và chỉ cung cấp quyền truy cập thông qua các phương thức công khai. Nó giúp bảo vệ trạng thái nội bộ của đối tượng và đảm bảo rằng chúng chỉ được thay đổi theo cách được kiểm soát.

- Thuộc tính private:
  - Các lớp như Player, Monster, Wall, Coin đều sử dụng các thuộc tính private như pos, image, score để bảo vệ dữ liệu.
  - Người dùng chỉ có thể truy cập hoặc thay đổi thông qua các phương thức công khai (public) như getPos(), addScore(int amount), draw(Graphics g, ImageObserver observer).

Bằng cách sử dụng các phương thức như isCollidingWithWall() hoặc tick(), các hành động liên quan được xử lý nội bộ, không cho phép truy cập trực tiếp

```java
public abstract class Character {
    protected BufferedImage image;
    protected Point pos;
    protected final ArrayList<Wall> walls;

    protected Character(ArrayList<Wall> walls, int x, int y, String imagePath) {
        this.walls = walls;
        this.pos = new Point(x, y);
        loadImage(imagePath);
    }

    protected void loadImage(String imagePath) {
        try {
            image = ImageIO.read(new File(imagePath));
        } catch (IOException exc) {
            System.out.println("Error opening image file: " + exc.getMessage());
        }
    }

    public void draw(Graphics g, ImageObserver observer) {
        g.drawImage(
                image,
                pos.x * Board.TILE_SIZE,
                pos.y * Board.TILE_SIZE,
                observer
        );
    }

    public Point getPos() {
        return pos;
    }

    protected boolean isCollidingWithWall(Point newPos) {
        return walls.stream().anyMatch(wall -> wall.getPos().equals(newPos));
    }

    public abstract void tick();
}
```

8.2 Tính kế thừa

Tính kế thừa cho phép một lớp (lớp con) thừa hưởng các thuộc tính và phương thức của một lớp khác (lớp cha). Trong các đoạn mã trên, không thấy trực tiếp có tính kế thừa được sử dụng. Tuy nhiên, các lớp như Wall, Monster, Player đều gián tiếp kế thừa từ lớp Object (mặc định trong Java).
Nếu thêm yêu cầu mở rộng, có thể sử dụng kế thừa như sau:

- Một lớp GameObject chứa các thuộc tính chung như pos, image và các phương thức như draw(Graphics g, ImageObserver observer).
- Các lớp Wall, Player, Monster, Coin có thể kế thừa từ lớp này để giảm trùng lặp mã.

```java
public class Monster extends Character {
    private final Player player;
    private long lastMoveTime;
    private static final long MOVE_DELAY = 500;

    public Monster(Player player, ArrayList<Wall> walls, int x, int y) {
        super(walls, x, y, imagePath:"images/monster.png");
        this.player = player;
        this.lastMoveTime = System.currentTimeMillis();
    }

    @Override
    public void tick() {
        long currentTime = System.currentTimeMillis();
        if (currentTime - lastMoveTime >= MOVE_DELAY) {
            moveTowardsPlayer();
            lastMoveTime = currentTime;
        }
    }

    private void moveTowardsPlayer() {
        Point newPos = new Point(pos.x, pos.y);

        if (pos.x < player.getPos().x) {
            newPos.x++;
        } else if (pos.x > player.getPos().x) {
            newPos.x--;
        }

        if (!isCollidingWithWall(newPos)) {
            pos.x = newPos.x;
        }

        newPos = new Point(pos.x, pos.y);
        if (pos.y < player.getPos().y) {
            newPos.y++;
        } else if (pos.y > player.getPos().y) {
            newPos.y--;
        }

        if (!isCollidingWithWall(newPos)) {
            pos.y = newPos.y;
        }
    }
}
```

8.3 Tính đa hình

Tính đa hình cho phép các lớp con cung cấp cách triển khai riêng cho các phương thức được định nghĩa trong lớp cha hoặc interface.

Trong các đoạn mã:

Ví dụ như phương thức tick() được ghi đè theo các cách thức thực hiện khác nhau trong Player và Monster

```java
public class Player extends Character {
    private int score;
    private final String playerName;
    private int life;
    @Override
    public void tick() {
        pos.x = Math.max(a:0, Math.min(pos.x, Board.COLUMNS - 1));
        pos.y = Math.max(a:0, Math.min(pos.y, Board.ROWS - 1));
    }
    public Player(String playerName, ArrayList<Wall> walls) {
```

```java
public class Monster extends Character {
    private final Player player;
    private long lastMoveTime;
    private static final long MOVE_DELAY = 500;

    public Monster(Player player, ArrayList<Wall> walls, int x, int y) {
        super(walls, x, y, imagePath:"images/monster.png");
        this.player = player;
        this.lastMoveTime = System.currentTimeMillis();
    }

    @Override
    public void tick() {
        long currentTime = System.currentTimeMillis();
        if (currentTime - lastMoveTime >= MOVE_DELAY) {
            moveTowardsPlayer();
            lastMoveTime = currentTime;
        }
    }
```

8.4 Tính trừu tượng

Tính trừu tượng là việc ẩn các chi tiết triển khai và chỉ hiển thị các chức năng quan trọng với người dùng.

Trong các đoạn mã:

- Do PLayer và Monster đều là các thực thể như nhau, cùng là nhân vật, cho nên chúng sẽ có những thuộc tính, phương thức chung. Do đó ta định nghĩa lớp trừu tượng Character.
- Tương tự, 4 vật thể Coin, Wall, Trap, Potion đều là vật thể, cùng kế thừa từ lớp trừu tượng GameObject.
- Người dùng chỉ cần gọi các phương thức như draw(), addScore() mà không cần biết chi tiết bên trong.

```java
public abstract class Character {
    protected BufferedImage image;
    protected Point pos;
    protected final ArrayList<Wall> walls;
    public abstract void tick();
```

CHƯƠNG 9. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

9.1 Kết luận

Trò chơi "Monster Chase: Coin Collector" là một sản phẩm giải trí sáng tạo, kết hợp hiệu quả giữa yếu tố hành động và chiến thuật. Người chơi không chỉ phải hoàn thành nhiệm vụ thu thập các đồng coin mà còn cần đối mặt với nhiều thử thách khác nhau như né tránh quái vật, vượt qua chướng ngại vật, và quản lý nguồn tài nguyên HP một cách hợp lý. Trò chơi đã đạt được mục tiêu khuyến khích người chơi phát triển khả năng tư duy, phản xạ nhanh nhạy cũng như lập kế hoạch hiệu quả. Qua đó, "Monster Chase: Coin Collector" không chỉ đơn thuần là một trò chơi giải trí mà còn mang tính giáo dục và rèn luyện kỹ năng cho người tham gia.

9.2 Hướng phát triển

Để nâng cao chất lượng và tăng tính hấp dẫn cho trò chơi trong tương lai, các hướng phát triển sau đây được đề xuất:

1. Mở rộng và tùy biến bản đồ: Việc bổ sung các bản đồ mới với các chủ đề đa dạng như rừng rậm, sa mạc, không gian vũ trụ, hoặc các môi trường thay đổi theo thời tiết sẽ giúp tăng sự phong phú cho trò chơi. Các yếu tố như lối đi ẩn, cửa bí mật hoặc vùng nguy hiểm sẽ tạo thêm thách thức cho người chơi.
2. Đa dạng hóa nhân vật và đối thủ: Giới thiệu thêm nhiều loại quái vật với đặc tính riêng như tốc độ di chuyển, tầm nhìn, hoặc khả năng truy đuổi sẽ khiến trò chơi thêm phần thú vị. Đồng thời, cho phép người chơi tùy chỉnh nhân vật của mình từ diện mạo đến các kỹ năng đặc biệt sẽ tăng cường tính cá nhân hóa.
3. Tăng tính tương tác xã hội: Tích hợp chế độ chơi nhiều người (multiplayer), cho phép người chơi hợp tác hoặc cạnh tranh trong việc thu thập coin. Bên cạnh đó, hệ thống xếp hạng toàn cầu hoặc theo khu vực sẽ tạo động lực để người chơi không ngừng cải thiện thành tích.
4. Cải tiến đồ họa và âm thanh: Tăng cường trải nghiệm bằng cách nâng cấp đồ họa với các hiệu ứng động sống động hơn và âm thanh đa dạng để phản ánh chính xác các sự kiện trong trò chơi.
5. Ứng dụng trí tuệ nhân tạo (AI): Sử dụng AI để tối ưu hành vi của quái vật, giúp chúng trở nên thông minh và khó lường hơn, từ đó tạo thêm thách thức cho người chơi. Ngoài ra, AI có thể được tích hợp để điều chỉnh độ khó dựa trên kỹ năng của từng người chơi, đảm bảo trải nghiệm luôn phù hợp và thú vị.

6. Tích hợp câu chuyện: Bổ sung một cốt truyện chi tiết giúp trò chơi có chiều sâu hơn, chẳng hạn như nhân vật chính có lý do cụ thể để thu thập coin, hoặc phải vượt qua các thử thách để đạt được mục tiêu lớn hơn.

Với những hướng phát triển trên, "Monster Chase: Coin Collector" không chỉ là một trò chơi đơn thuần mà còn có tiềm năng trở thành một sản phẩm chất lượng cao, phù hợp với nhu cầu giải trí ngày càng đa dạng của người chơi trong thời đại hiện nay.