TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

---------------o0o---------------



**Bài tập lớn môn học**

**CÔNG NGHỆ JAVA**

**Đề tài: Lập Trình Game Pirate Adventure**

|  |  |
| --- | --- |
| Sinh viên thực hiện: | Phạm Huy Hoàng 212606012 |
|  |
| Lớp : | CNTTVA2\_K62 |

**Hà Nội tháng 5 năm 2023**

Mục lục

[**I.** **Lời cảm ơn** 2](#_Toc134399931)

[**II.** **Mở đầu** 2](#_Toc134399932)

[1. Lý do chọn đề tài 2](#_Toc134399933)

[2. Mục tiêu của đề tài 3](#_Toc134399934)

[3. Công nghệ sử dụng 3](#_Toc134399935)

[4. Phạm vi nghiên cứu 4](#_Toc134399936)

[**III.** **Thiết kế hệ thống** 4](#_Toc134399937)

[**IV.** **Các công nghệ hướng đối tượng được sử dụng trong ứng dụng** 6](#_Toc134399938)

[1. Kế thừa 6](#_Toc134399939)

[2. Bắt sự kiện trong Java AWT 6](#_Toc134399940)

[3. Giao diện tương tác với người dùng 8](#_Toc134399941)

[**V.** **Chi tiết hệ thống** 9](#_Toc134399942)

[1. Hệ thống file 9](#_Toc134399943)

[2. Chi tiết chương trình hoạt động 12](#_Toc134399944)

[**VI.** **Kết luận và kiến nghị** 39](#_Toc134399945)

[**VII.** **Danh mục tài liệu tham khảo** 39](#_Toc134399946)

[**VIII.** **Phụ lục** 39](#_Toc134399947)

1. **Lời cảm ơn**

Em xin cảm ơn cô Thủy đã hỗ trợ em trong quá trình học tập rất nhiều.

1. **Mở đầu**
2. Lý do chọn đề tài

Công nghệ thông tin (CNTT) đang đóng vai trò vô cùng quan trọng trong cuộc sống hiện đại ngày nay. Việc ứng dụng CNTT vào các lĩnh vực khác nhau giúp chất lượng việc làm và đời sống được nâng cao lên rất nhiều so với thời đại trước. Một trong số đó là game – một ứng dụng mang lại rất nhiều giá trị về mặt tinh thần cũng như về mặt kinh tế.

Em chọn đề tài “Lập trình game Pirate’s Adventure bằng ngôn ngữ Java” nhằm tìm hiểu sâu hơn về ngôn ngữ Java và các kĩ thuật của chúng, từ đó viết một ứng dụng cụ thể để thử nghiệm nhằm củng cố kiến thức cũng như định hướng, lên kế hoạch xây dựng những ứng dụng game cụ thể, phát triển theo hướng dịch vụ trong tương lai.

1. Mục tiêu của đề tài

Hiểu khái niệm phân tích và giải quyết bài toán lập trình hướng đối tượng: Xác định các đối tượng lớp, mối quan hệ giữa các lớp trong một hệ thống thông tin.

Mục tiêu đề tài là giúp em nghiên cứu môi trường phát triển của ngôn ngữ Java, qua đó có khả năng xây dựng những ứng dụng cụ thể.

1. Công nghệ sử dụng

Trong xu thế phát triển CNTT nhanh như hiện nay, đặc biệt là trong ngành Công nghệ phần mềm (CNPM), lập trình viên (LTV) phải có trình độ cao trong kĩ thuật lập trình. Vì vậy, phương pháp lập trình hướng thủ tục cổ điểm trước đây không thể đáp ứng được nhu cầu đặt ra của các chương trình phức tạp hiện nay. Một phương pháp lập trình mới được xây dựng nhằm đáp ứng những nhu cầu đó: “Phương pháp Lập trình Hướng đối tượng (LTHDT)”.

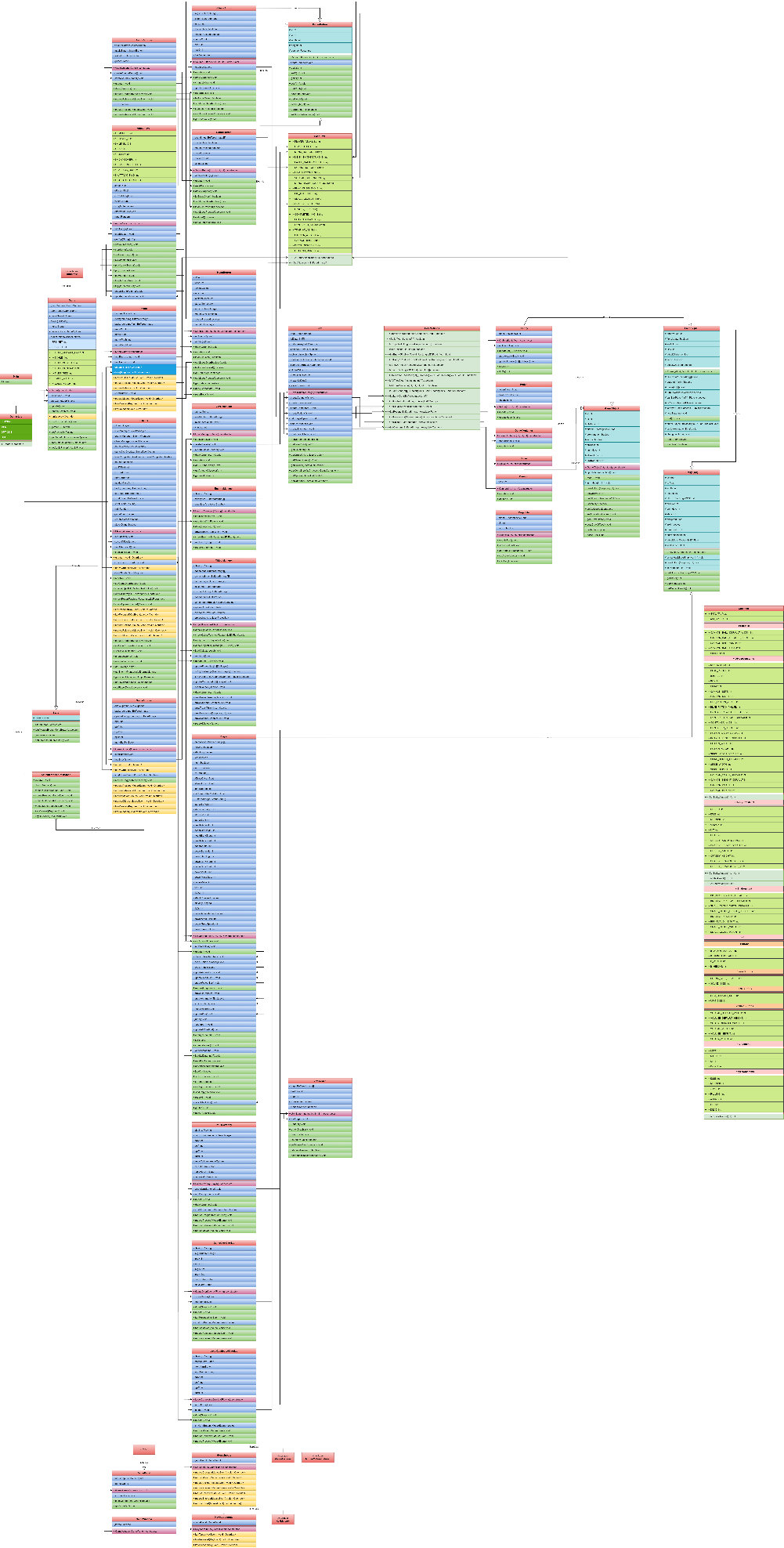
Chương trình này được thiết kế theo phương pháp LTHDT bằng ngôn ngữ Java, do SunMicroSystem xây dựng và phát triển vào năm 1991. Những thế mạnh đặc trưng của Java là:

* Mã chương trình rõ ràng, dễ đọc, dễ hiểu và cô đọng.
* Hướng đối tượng: Hỗ trợ tất cả các đặc trưng của HDT như: class, object, đóng gói, kế thừa, đa hình, nạp chồng, ghi đè, …
* Độc lập với nền tảng (phần cứng và hệ điều hành): Khi biên dịch, code được chuyển thành file \*.class (file chứa mã bytecode để máy ảo thực thi). File này có thể chạy ở bất kỳ đâu có máy ảo Java.
* Bảo mật: Quản lý thực thi chương trình ở nhiều mức:
  + Dữ liệu và phương thức được đóng gói bên trong lớp.
  + Trình thông dịch kiểm soát bytecode đảm bảo quy tắc an toàn trước khi thực thi.
* Mạnh mẽ: Java kiểm soát chặt chẽ các vấn đề như kiểu dữ liệu phải tường minh, không hỗ trợ con trỏ, đảm bảo không tràn mảng, tự động cấp phát và giải phóng bộ nhớ thông qua bộ thu dọn rác.
* Đa luồng: Có thể viết chương trình có thể làm nhiều việc cùng một lúc. Cho phép các nhà phát triển xây dựng các ứng dụng tương tác chạy đồng thời và đồng bộ.
* Phân tán: Java có thể phát triển các ứng dụng Web, phân tán thông qua gói Java.net và công nghệ J2EE.
* Linh động: Java được coi là linh động hơn so với C hoặc C++ vì nó được thiết kế để thích ứng với môi trường phát triển mở.
* Các chương trình Java có thể xử lý số lượng lớn thông tin thực thi mà có thể được dùng để xác minh và giải quyết các truy cập đến các đối tượng trong thời gian thực thi.

1. Phạm vi nghiên cứu

Nghiên cứu môi trường phát triển, ngôn ngữ Java trong LTHDT. Tìm hiểu cách xây dựng các phương thức, thuộc tính đối tượng trong Java.

1. **Thiết kế hệ thống**



Hình 1.1: Sơ đồ hệ thống game

1. **Các công nghệ hướng đối tượng được sử dụng trong ứng dụng**
2. Kế thừa

Tính kế thừa trong Java là một kỹ thuật mà trong đó một đối tượng thu được tất cả các thuộc tính và hành vi của đối tượng cha. Ý tưởng đằng sau tính kế thừa trong Java là bạn có thể tạo các lớp mới mà được xây dựng dựa trên các lớp đang tồn tại. Khi bạn kế thừa từ một lớp đang tồn tại, bạn có thể tái sử dụng các phương thức và các trường của lớp cha, và bạn cũng có thể bổ sung thêm các phương thức và các trường khác. Tính kế thừa biểu diễn mối quan hệ IS-A, còn được gọi là mối quan hệ cha con.

Khi chúng ta nói về tính kế thừa, từ khóa thường xuyên nhất được sử dụng là extends và implements. Những từ khóa này có thể định nghĩa một kiểu là loại IS-A của loại khác. Bằng việc sử dụng các từ khóa, chúng ta có thể tạo một đối tượng sử dụng thuộc tính của đối tượng khác. Chúng ta sử dụng từ khóa extends của lớp con để có thể kế thừa các thuộc tính của lớp cha trừ các thuộc tính private của lớp cha.

Cú pháp của tính kế thừa trong Java

class ten\_lop\_con extends ten\_lop\_cha

{

//cac phuong thuc va cac truong

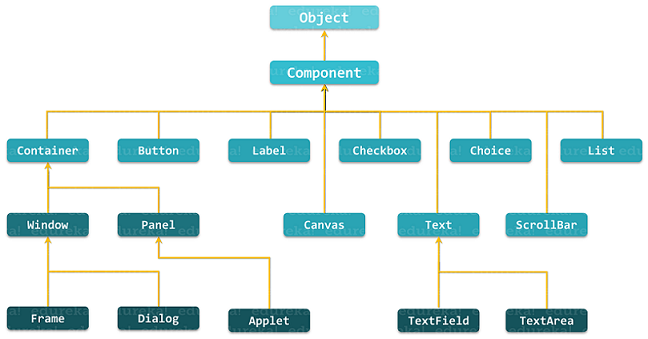
}

1. Bắt sự kiện trong Java AWT

Các hệ thống GUI xử lý các tương tác người dùng với sự trợ giúp của mô hình hướng sự kiện (event-driven).

Sự kiện trong Java:

* Sự kiện có thể được định nghĩa như là một kiểu tín hiệu tới chương trình thông báo một việc gì đó đã xảy ra.
* Sự kiện được sinh ra bởi các hành động của người dùng, ví dụ như di chuyển, click chuột, hay nhấn phím, ….



Hình 2.1: Sơ đồ Java AWT

Việc xử lý những sự kiện này phụ thuộc vào ứng dụng. Abstract Windowing Toolkit (AWT) xử lý một vài sự kiện. Môi trường mà các ứng dụng này được thi hành, ví dụ như trình duyệt, cũng có thể xử lý các sự kiện khác. Người lập trình cũng cần phải xử lý những sự kiện nhất định và cần phải viết hàm xử lý các sự kiện đó.

Ứng dụng cần đăng ký một hàm xử lý sự kiện với một đối tượng. Hàm xử lý sự kiện này sẽ được gọi bất cứ khi nào sự kiện tương ứng phát sinh.

Trong quy trình này, ứng dụng cho phép bạn đăng ký các phương thức (handler), hay còn gọi là listener với các đối tượng. Những handler này tự động được gọi khi một sự kiện thích hợp phát sinh.

Một Event Listener lắng nghe một sự kiện nào đó mà một đối tượng đã thiết lập. Mỗi Event Listener cung cấp các phương thức xử lý những sự kiện này. Lớp thực thi listener cần phải định nghĩa các phương thức. Để sử dụng mô hình, bạn có thể làm theo các bước sau:

* Cài đặt giao diện listener thích hợp:

**public** **class** Menu **extends** State **implements** Statemethods

* Xác định tất cả các thành phần tạo ra sự kiện. Các thành phần có thể là các button, label, menu item hay window:

jframe.addWindowFocusListener(**new** WindowFocusListener())

* Xác định tất cả các sự kiện được xử lý. Các sự kiện có thể là một ActionEvent nếu một button được click hay một MouseEvent nếu như chuột được kéo đi.
* Thi hành các phương thức của listener và viết hàm xử lý sự kiện tương ứng với các phương thức.

|  |  |
| --- | --- |
| Lớp sự kiện | Mô tả |
| ActionEvent | Phát sinh khi một button được nhấn, một item trong danh sách chọn lựa được nhấn đúp (double-click) hay một menu được chọn. |
| TextEvent | Phát sinh khi giá trị trong thành phần textfield hay textarea bị thay đổi. |
| MouseEvent | Phát sinh khi chuột di chuyển, được click, kéo hay thả ra. |
| KeyEvent | Phát sinh khi bàn phím ấn, nhả |

Bảng 1: Một số sự kiện được sử dụng trong ứng dụng và mô tả.

Các giao diện định nghĩa một số phương thức để xử lý mỗi sự kiện. Những phương thức này sẽ được nạp chồng trong lớp cài đặt các giao diện này.

1. Giao diện tương tác với người dùng

Các ứng dụng phần mềm hiện nay rất thân thiện vì được trình bày nhiều màn hình giao diện đồ họa đẹp mắt. Các ngôn ngữ lập trình hiện nay cung cấp các đối tượng đồ họa mà người lập trình hay người sử dụng có thể điều khiển được. Một trong số những kết quả quan trọng nhất chính là các ngôn ngữ hiện nay được dựa trên Giao diện đồ họa người dùng (Graphical User Interface - GUI).

GUI cung cấp chức năng nhập liệu theo cách thân thiện với người dùng. Nó đa dạng và có thể chứa nhiều điều khiển như hộp văn bản, nhãn, hộp danh sách hay các điều khiển khác. Một phần tử GUI được thiết lập bằng cách sử dụng thủ tục sau:

* Tạo đối tượng
* Xác định sự xuất hiện ban đầu của đối tượng
* Chỉ ra nó nằm ở đâu
* Thêm phần tử vào giao diện trên màn hình

Một thành phần (component) GUI là một đối tượng trực quan. Người dùng tương tác với đối tượng này thông qua con trỏ chuột hay bàn phím. Các thành phần như button, label có thể được nhìn thấy trên màn hình. Mọi thành phần chung của tất cả các thành phần GUI đều được tìm thấy trong lớp Component. Để tạo các đối tượng GUI chúng ta cần nhập gói java.awt hoặc gói java.swing.



Hình 2.2: Cấu trúc cây của gói Swing

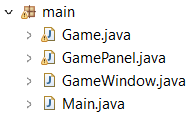
1. **Chi tiết hệ thống**
2. Hệ thống file

Sản phẩm của em gồm 7 package và 1 folder, mỗi phần mang một chức năng khác nhau. Sau đây, em xin trình bày chi tiết về sản phẩm của mình.

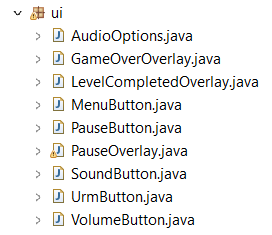
Về folder res, chúng được dùng để chứa dữ liệu ảnh và âm thanh. Trong folder res có 2 folder con là audio để chứa các tệp âm thanh và lvls để chứa các màn chơi trong game.

Về các package, 7 package bao gồm:

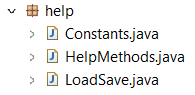
* Main: chứa các class khung chính cho chương trình



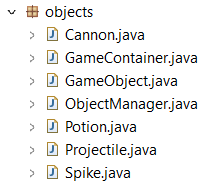
* Ui: chứa các class xử lý GUI



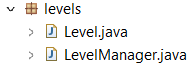
* Help: chứa các class có các hàm và các biến sẽ sử dụng trong chương trình.



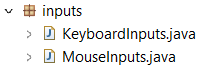
* Objects: chứa các class dùng cho các thực thể trong chương trình.



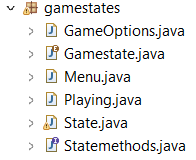
* Levels: chứa các class dùng cho việc lưu trữ các phần lưới ảnh.



* Inputs: chứa các class xử lý các tín hiệu từ bàn phím và chuột.



* Gamestates: chứa các class xử lý các hoạt động chính của game.



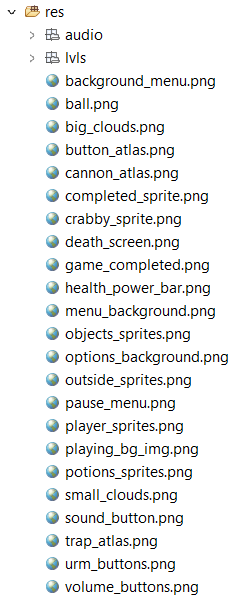
* Entities: chứa các class về quái và nhân vật trong game.



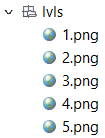
* Audio: chứa các class xử lý âm thanh.



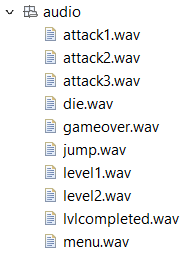
Về thư mục chứa đồ họa và âm thanh res:



* Thư mục level dùng để chứa các bản đồ của game:



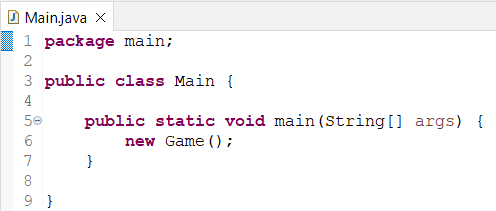
* Thư mục audio dùng để chứa các file âm thanh của game:



1. Chi tiết chương trình hoạt động

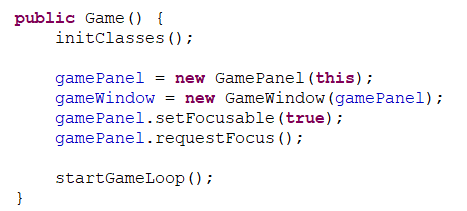
Sau đây, em xin đi vào chi tiết các hoạt động của chương trình này.

Chương trình của em sẽ được chạy từ file Main.java

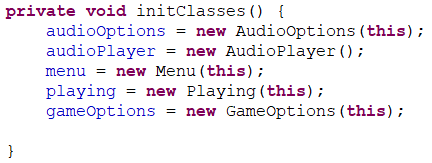


Như đã thấy ở trên, trong class Main có hàm main, và chương trình sẽ chỉ chạy nhưng hàm được gọi trong hàm main này.

Ở trong hàm main này, đối tượng kiểu Game được gọi. Do đó, chương trình sẽ chạy đến hàm tạo của đối tượng kiểu Game.



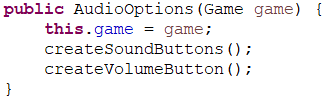
Trong hàm tạo Game() này, hàm đầu tiên được gọi là hàm initClasses() của class Game.



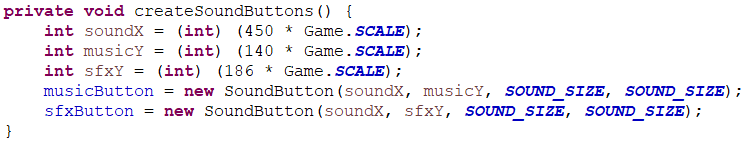
Trong hàm initClasses(), có 5 đối tượng được tạo ra bao gồm:

* audioOptions có kiểu AudioOptions
* audioPlayer có kiểu AudioPlayer
* menu có kiểu Menu
* playing có kiểu Playing
* gameOptions có kiểu GameOptions

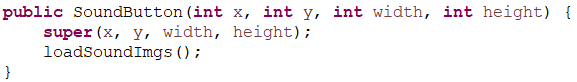
Như vậy, trong hàm initClasses() này có 5 hàm tạo của 5 kiểu đối tượng được tạo ra.



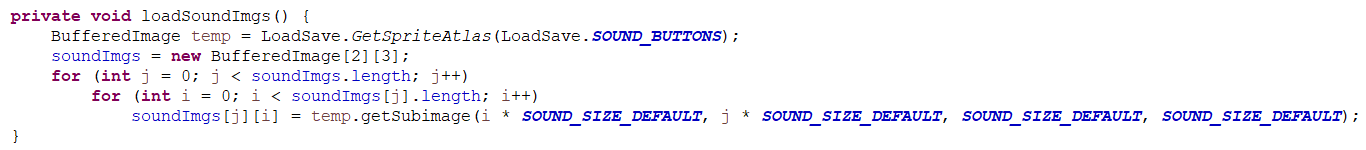
Trong hàm tạo AudioOptions(Game game), chúng ta gán đối tượng game của class Game cho biến game của class AudioOptions. Tiếp đến, chúng ta gọi hàm createSoundButtons() và createVolumeButton().



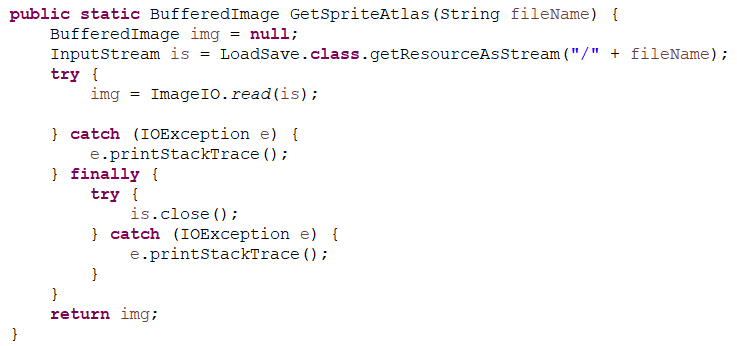
Trong hàm createSoundButtons(), biến soundX, musicY và sfxY được gán cho các giá trị cụ thể; và 2 đối tượng musicButton và sfxButton được tạo mới bằng hàm tạo kiểu SoundButton.



Trong hàm tạo SoundButton(int, int, int, int) này, các biến x, y, width, height được kế thừa từ class PauseButton; và hàm loadSoundImgs() được gọi đến.

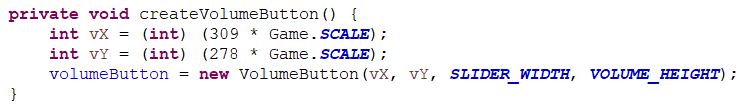


Ở hàm loadSoundImgs(), biến cục bộ temp có kiểu BufferdImage được tạo ra và được gán giá trị bằng ảnh sound\_button.png được lưu trữ trong folder res.

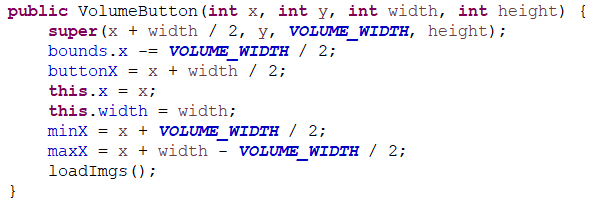


Sau khi đọc ảnh xong, mảng 2 chiều soundImgs được tạo ra để tách từng phần của bức ảnh vừa đọc.

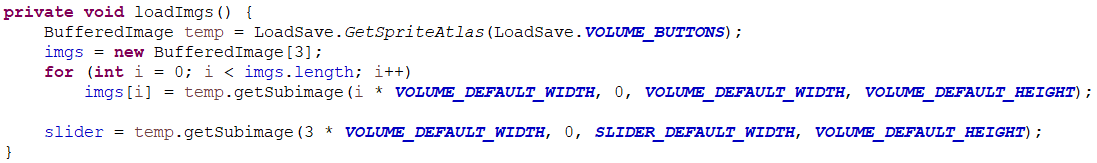
Trở lại với hàm createVolumeButton(), 2 biến cục bộ vX và vY được tạo ra nhằm mục đích làm thông số cho hàm tạo VolumeButton(int, int, int, int).



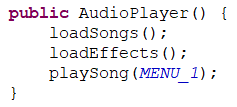
Hàm tạo VolumeButton(int, int, int, int) được gọi để tạo đối tượng kiểu VolumeButton cho biến volumeButton.



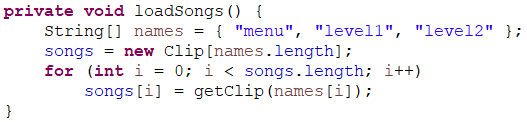
Trong hàm tạo VolumeButton(int, int, int, int), các biến x, y, width, height, được kế thừa từ class PauseButton. Ở cuối hàm tạo, hàm loadImgs() được gọi để đọc ảnh volume\_buttons.png.



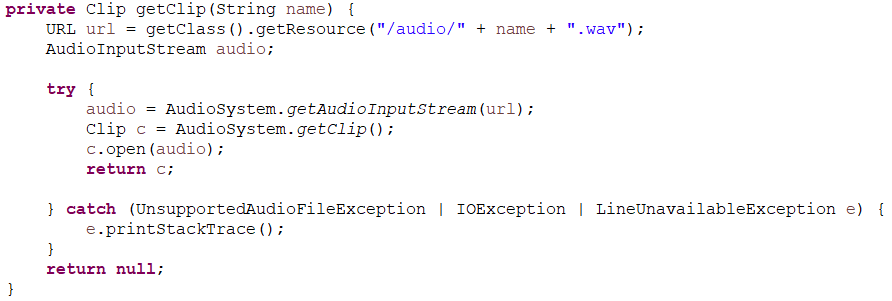
Sau khi tạo xong biến có kiểu AudioOptions, biến có kiểu AudioPlayer sẽ được tạo.



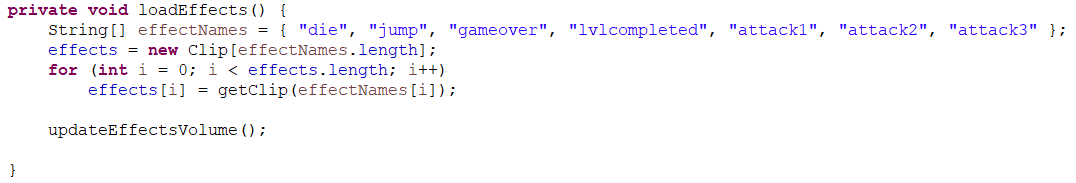
Trong hàm tạo AudioPlayer(), hàm loadSongs() được gọi để đọc cái file âm thanh của nhạc game.



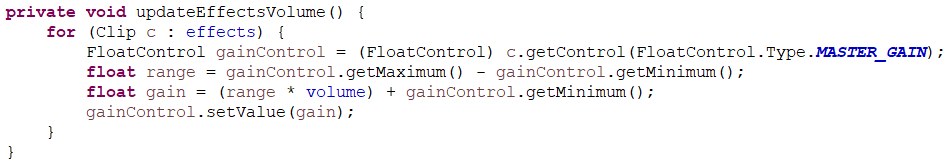
Mỗi biến trong mảng songs được lưu giá trị bằng file âm thanh, các file này được đọc bằng hàm getClip(String).



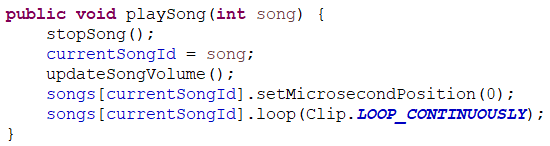
Sau đó, hàm loadEffects() được gọi để đọc các file âm thanh của hiệu ứng.



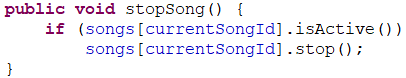
Trong hàm loadEffect() này, hàm updateEffectsVolume() được gọi để đặt giá trị âm lượng.



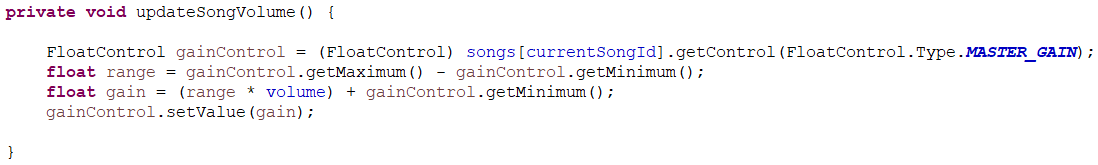
Sau khi lưu xong các file âm thanh, nhạc nền menu sẽ được chạy khi khởi động chương trình qua hàm playSong(MENU\_1).



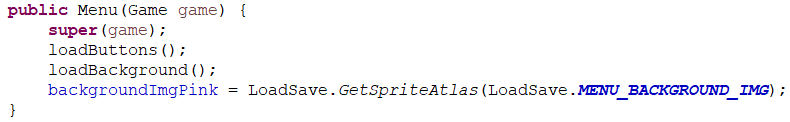
Trong hàm playSong(int) này, hàm stopSong() được gọi đầu tiên để dừng nhạc đang chạy dở.



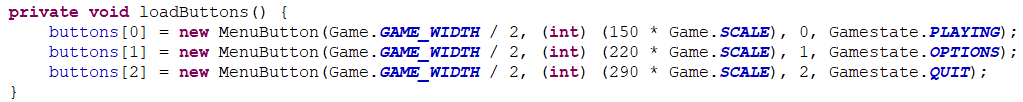
Sau đó, biến currentSongId được gán bằng âm thanh mới, sau đó, hàm updateSongVolume() được gọi để đặt lại các thông số âm thanh.



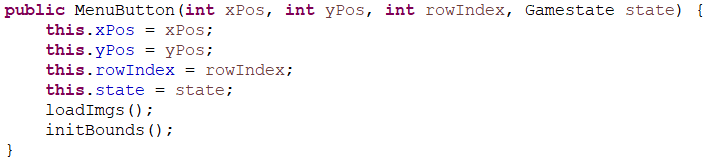
Sau khi tạo xong đối tượng kiểu AudioPlayer(), biến menu kiểu Menu được tạo ra bằng hàm tạo Menu(Game).



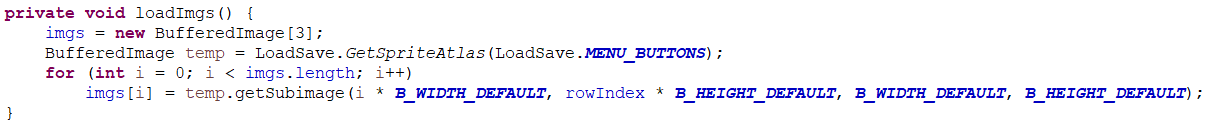
Trong hàm tạo Menu(Game), hàm loadButtons() được gọi để tạo các nút trong phần menu.



Trong hàm loadButtons(), có 3 nút được tạo ra với kiểu biến MenuButton bằng hàm tạo MenuButton(int, int, int, Gamestate) với các chức năng khác nhau.



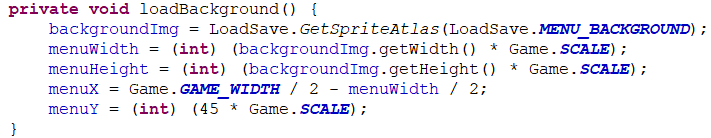
Trong hàm tạo MenuButton(int, int, int, Gamestate), hàm loadImgs() được gọi để đọc ảnh.



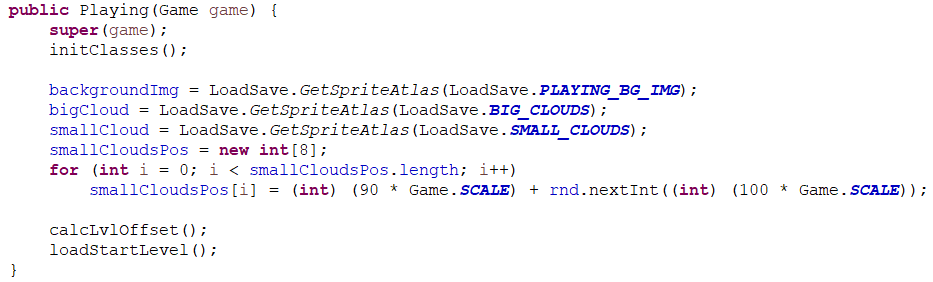
Sau đó, hàm initBounds() được gọi để tạo vùng tương tác với chuột.



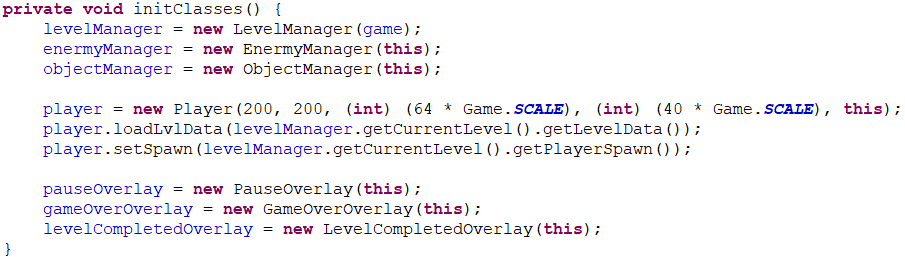
Sau khi tạo các nút xong, ảnh nền phần menu sẽ được đọc bằng hàm loadBackground().



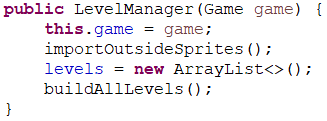
Sau khi đã tạo xong đối tượng kiểu Menu, biến playing sẽ được tạo bằng hàm tạo Playing(Game).



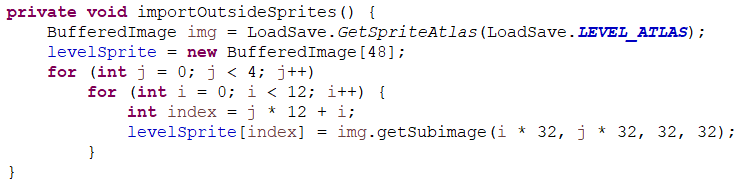
Đầu tiên, hàm initClasses() được gọi để khởi tạo các giá trị trong class Playing.



Ở trong hàm initClasses(), biến levelManager được khởi tạo bằng hàm tạo LevelManager(Game).

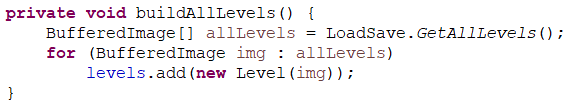


Hàm importOutsideSprites() được gọi để đọc bản đồ.

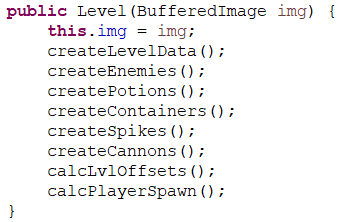


Sau đó, levels được khởi tạo bằng hàm tạo của collection ArrayList<>().

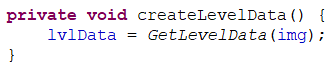
Hàm buildAllLevels() được gọi cuối cùng.



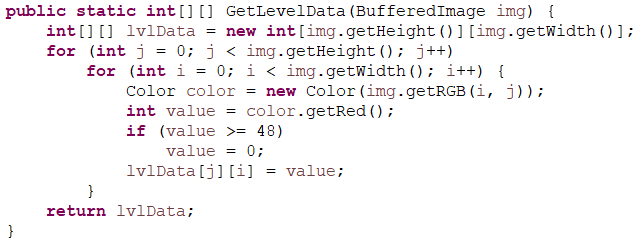
Trong hàm buildAllLevles() này, từng ảnh sau khi được đọc sẽ được dùng cho hàm tạo Level(BufferedImage) của từng phần tử của mảng levels.



Sau khi gán giá trị img của class Level bằng đối vào, chương trình gọi đến hàm createLevelData().



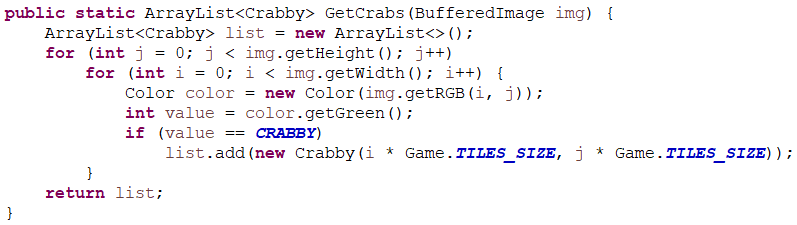
Trong hàm createLevelData(), biến lvlData được tạo bằng hàm GetLevelData(BuferedImage).



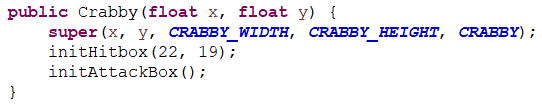
Sau khi tạo xong thanh máu, chương trình bắt đầu gọi đến hàm createEnemies().



Ở trong hàm createEnemies(), biến crabs được tạo bằng hàm GetCrabs(BufferedImage).

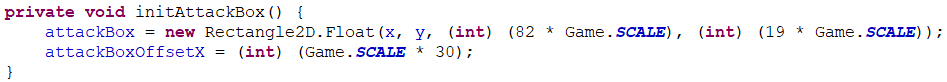


Do game của ta có nhiều quái vật nên chương trình sẽ lưu trữ các con quái đó bằng collection ArrayList, mỗi phần tử là một biến kiểu Crabby và tất nhiên, các phần tử đó được tạo mới bởi hàm tạo Crabby(float, float).



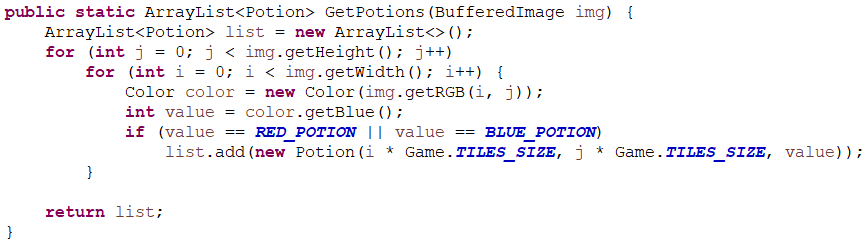
Trong hàm tạo Crabby(float, float), sau khi kế thừa các thông số ở class Enermy, hàm initHitbox(int, int) và initAttackBox() được gọi.



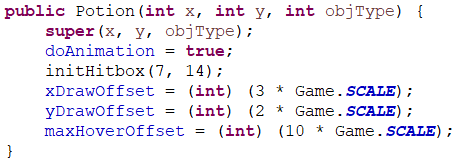


Sau khi tạo xong quái, chương trình sẽ gọi đến hàm createPotions().



Trong hàm createPotions(), biến potions sẽ được khởi tạo giá trị bằng hàm GetPotions(BufferedImage).  


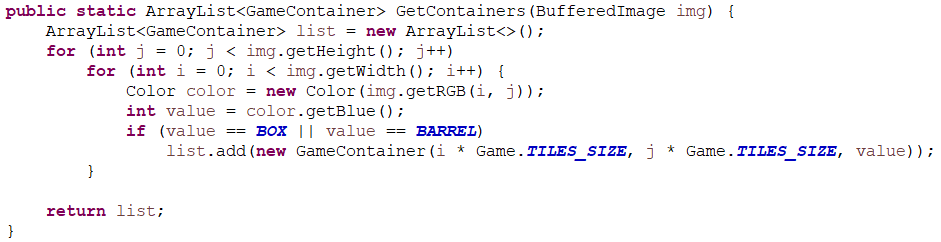
Do game có nhiều thuốc hồi máu và thuốc hồi năng lượng nên cũng giống như quái, chúng sẽ được lưu trữ trong collection ArrayList. Các phần tử của mảng sẽ được khởi tạo bằng hàm tạo Potion(int, int, int).



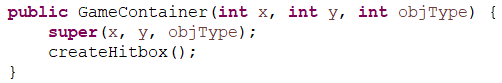
Sau khi khởi tạo xong thuốc, chương trình sẽ gọi đến hàm createContainers().

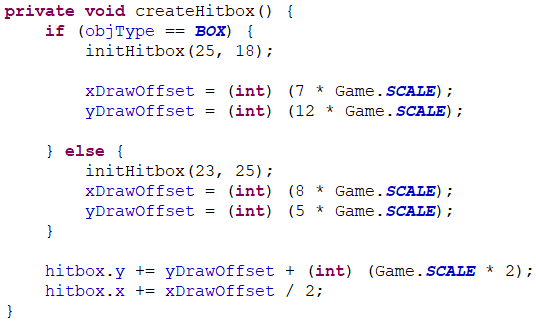


Trong hàm createContainers(), biến containers được khởi tạo bằng hàm GetContainers(BufferedImage).



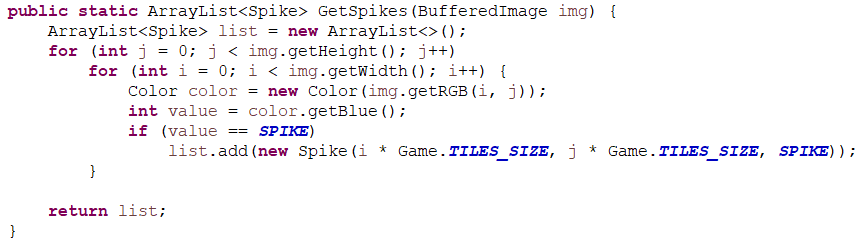
Do game chứa nhiều thùng nên chương trình sẽ lưu trữ chúng bằng collection ArrayList, các phần tử trong mảng được khởi tạo bằng hàm tạo GameContainer(int, int, int).

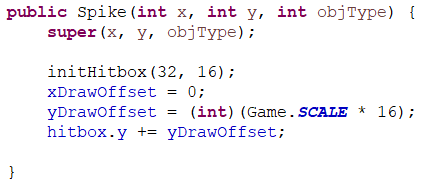




Sau khi tạo xong các thùng, ta đi



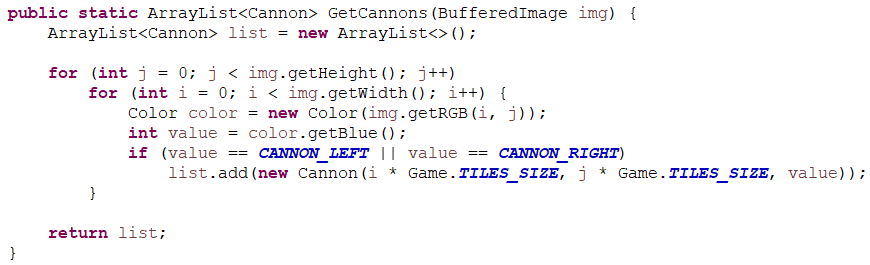




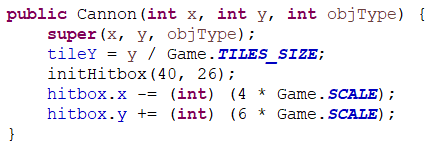
Tiếp theo, chương trình sẽ gọi đến hàm tạo đại bác trong game là hàm createCannons().



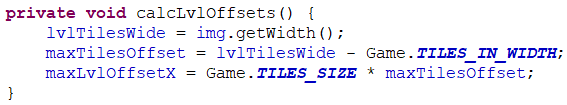
Trong hàm createCannons(), biến cannons được khởi tạo giá trị bằng hàm GetCannons(BufferedImage).



Do game có nhiều khẩu đại bác nên chương trình sẽ dùng collection ArrayList để chứa các khẩu đại bác này. Mỗi phần tử trong list sẽ được khởi tạo giá trị bằng hàm tạo Cannon(int, int, int).

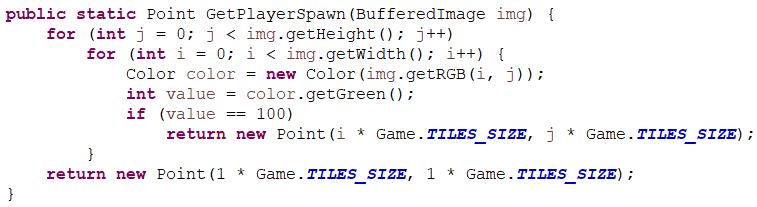


Sau khi khởi tạo xong biến cannons, chương trình sẽ gọi đến hàm calcLvlOffsets().

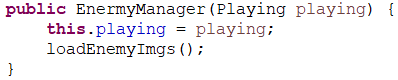


Sau khi hàm calcLvlOffsets() hoàn thành, chương trình sẽ gọi đến hàm calcPlayerSpawn().

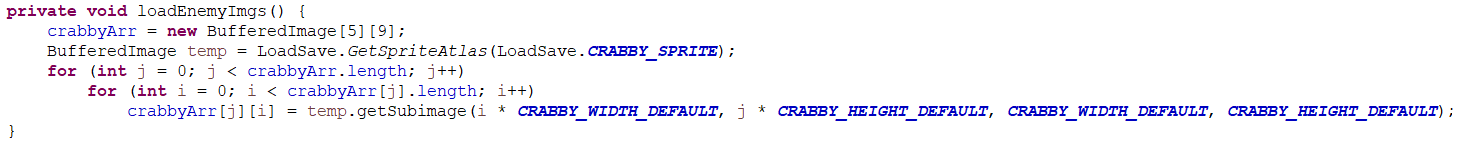




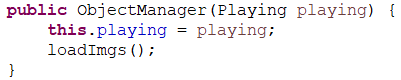
Biến enermyManager được khởi tạo bằng hàm tạo EnermyManager(Playing)



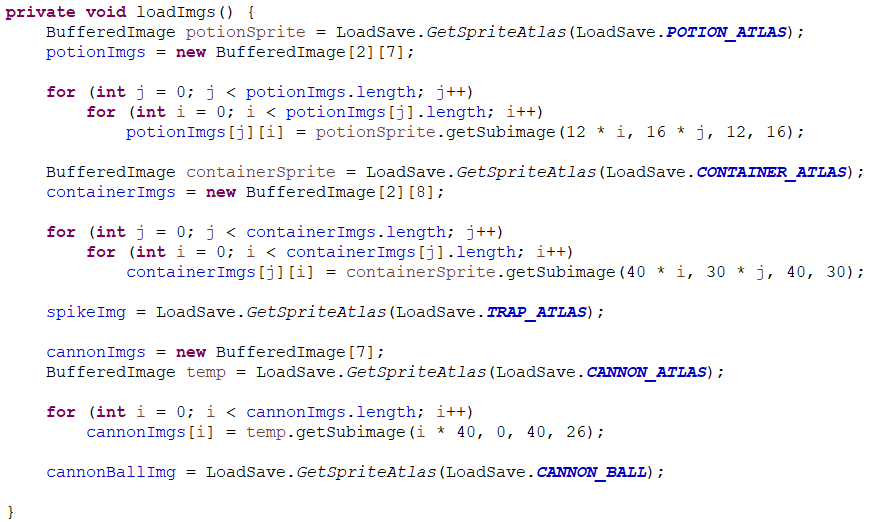
Trong hàm này, hàm loadEnemyImgs() được gọi đến để tải dữ liệu ảnh của quái vào chương trình.



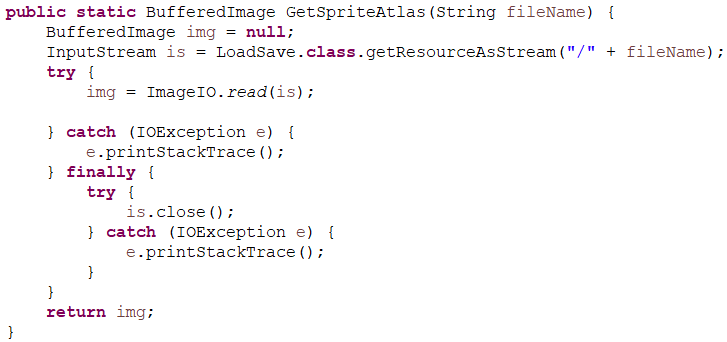
Biến objectManager được khởi tạo bằng hàm tạo ObjectManager(Playing).



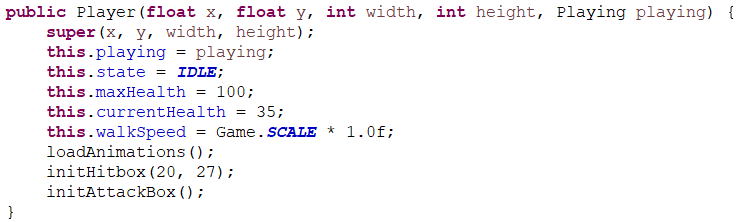
Trong hàm tạo ObjectManager(Playing), hàm loadImgs() được chương trình gọi đến để đọc dữ liệu ảnh.



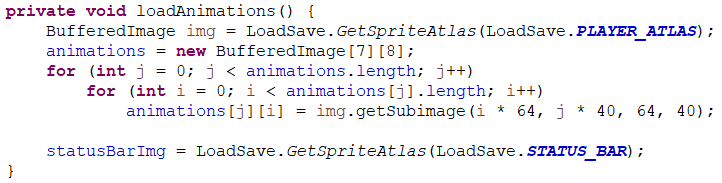
Để đọc dữ liệu ảnh, chương trình sẽ gọi đến hàm GetSpriteAtlas(String).



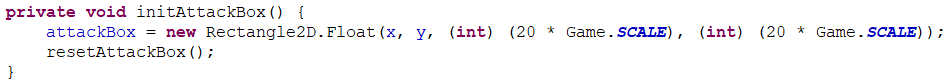
Sau đó, chương trình sẽ bắt đầu khởi tạo player bằng hàm tạo Player(float, float, int, int, Playing).



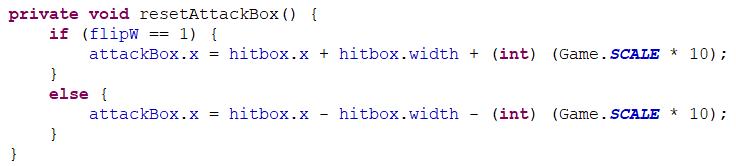
Trong hàm tạo Player(float, float, int, int, Playing), các biến x, y, width, height được kế thừa từ lớp Entity. Sau khi gán giá trị cho các biến playing, state, maxHealth, currentHealth và walkSpeed, chương trình sẽ gọi đến hàm loadAnimations() để tải đồ họa của người chơi bao gồm các trạng thái của nhân vật, thanh máu và thanh năng lượng.



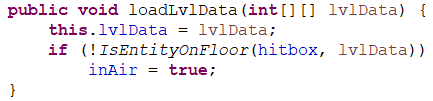
Sau khi tải xong đồ họa của nhân vật, chương trình sẽ gọi đến hàm initHitbox(). Sau đó, chương trình sẽ gọi đến hàm initAttackBox() để tạo vùng đánh.



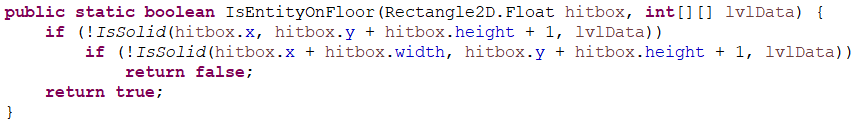
Trong hàm initAttackBox(), hàm resetAttackBox() được gọi để chỉnh vùng tấn công sau mỗi lần di chuyển.



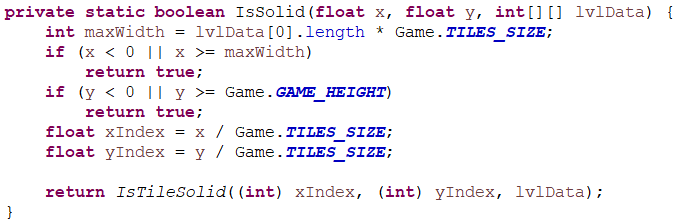
Sau đó, chương trình sẽ gọi đến hàm loadLvlData(int[][]) để đọc các grid của bản đồ các màn chơi.



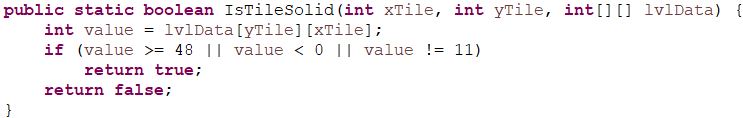
Trong hàm loadLvlData(int[][]), hàm IsEntityOnFloor(Rectangle2D.Float, int[][]) được dùng để kiểm tra xem phần grid đó có phải là nền hay không.



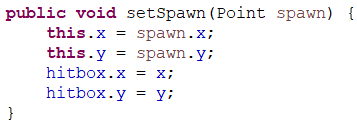
Trong hàm kiểm tra các grid IsEntityOnFloor(Rectangle2D.Float, int[][]), hàm IsSolid(float, float, int[][]) được sử dụng để kiểm tra xem phần grid trên bản đồ có đi qua được không.



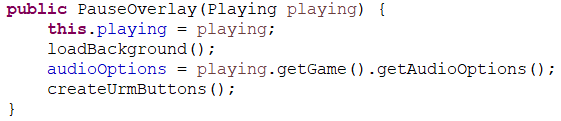
Ở



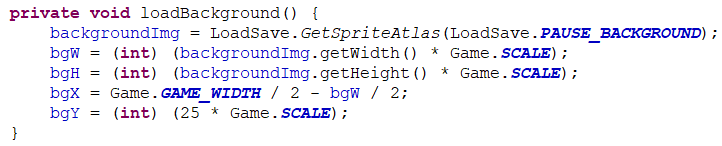
Sau đó,



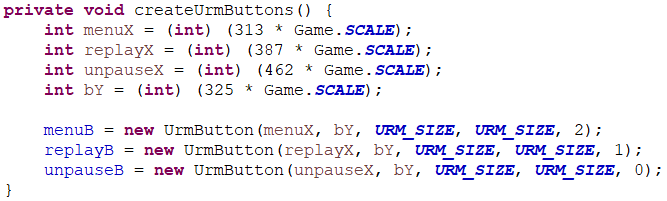
Sau đó, chương trình gọi đến hàm PauseOverlay(Playing)



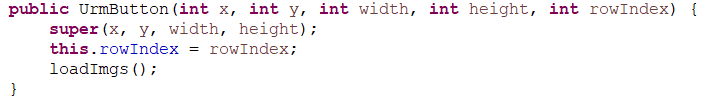
Trong hàm PauseOverlay(Playing), hàm loadBackground() được chương trình gọi đến để đọc ảnh nền khi chương trình tạm dừng.



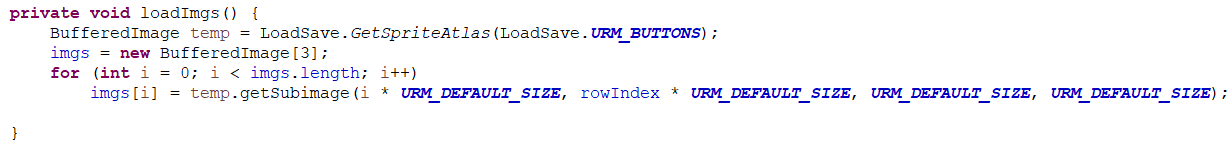
Sau đó hàm createUrmButtons() được gọi đến để tạo các nút trở về menu, khởi động lại game và trở lại game.



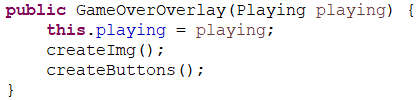
Các nút được tạo bằng hàm tạo UrmButton(int, int, int, int, int).



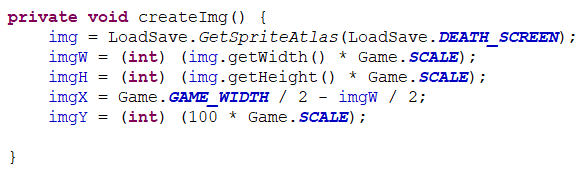
Trong hàm tạo UrmButton(int, int, int, int, int), hàm loadImgs() được gọi đến để đọc ảnh của các nút trên.

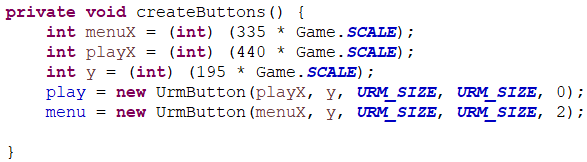


Sau đó, biến gameOverOverlay được khởi tạo bằng hàm tạo GameOverOverlay(Playing).

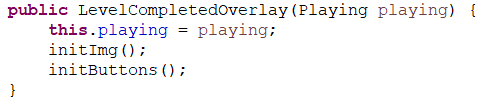


Trong hàm GameOverOverlay(Playing), hàm createImg() và hàm createButtons() lần lượt được gọi nhằm đọc dữ liệu ảnh, đồng thời tạo nút play và nút menu để tương tác với người chơi.

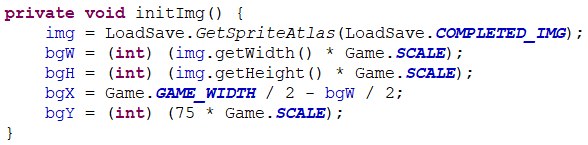


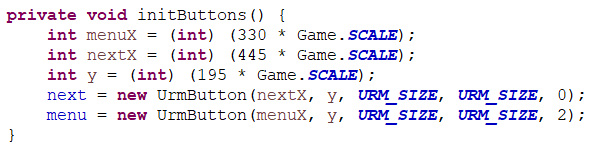


Sau đó, biến levelCompletedOverlay được khởi tạo bằng hàm LevelCompletedOverlay(Playing).

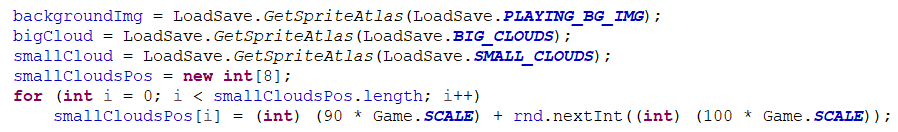


Trong hàm tạo LevelCompletedOverlay(Playing), hàm initImg() và hàm initButtons() lần lượt được gọi để đọc dữ liệu ảnh và tải đồ họa vào game.

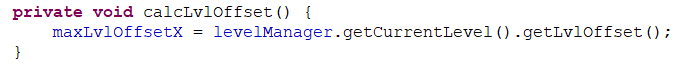




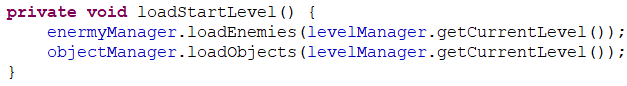
Sau khi hàm initClasses() được hoàn thành, chương trình sẽ gọi đến hàm GetSpriteAtlas() ở class LoadSave để đọc và tải đồ họa trong khi chương trình vào phần playing.



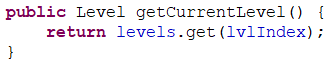
Sau khi tải xong dữ liệu đồ họa, chương trình gọi đến hàm calcLvlOffset().



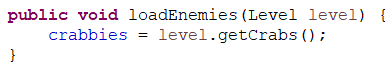
Sau đó, chương trình tiếp tục gọi đến hàm loadStartLevel() để tải các dữ liệu về các thực thể trong game.



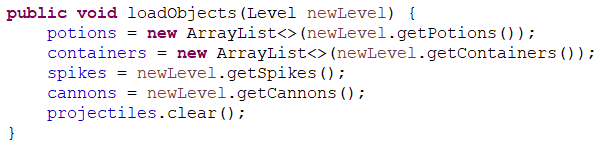
Trong hàm loadEnemies(Level), hàm getCurrentLevel() được dùng để lấy grid làm dữ liệu đầu vào.



Trong hàm loadEnemies(Level) này, biến crabbies được gán giá trị bằng hàm getCrabs() của biến đầu vào.



Sau khi khởi tạo xong các quái ở trong map, chương trình sẽ gọi đến hàm loadObjects(Level) để khởi tạo các giá trị cho các thực thể trong game.



Trong hàm loadObjects(Level), các biến potions, containers, spikes và cannons được khởi tạo lần lượt bằng các hàm getPotions(), getContainers(), getSpikes() và getCannons(). Do có nhiều hơn 1 potion và 1 container nên chúng được lưu trữ bằng collection ArrayList.

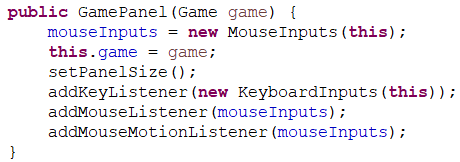








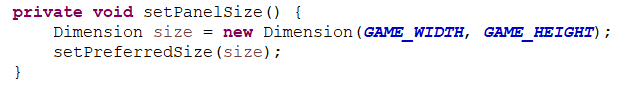
Sau khi hoàn thành hàm initClasses(), gamePanel được tạo bằng hàm tạo GamePanel(Game).



Biến mouseInputs nhận các tín hiệu từ bàn phím được tạo bằng hàm tạo của class MouseInputs.



Hàm setPanelSize() được gọi nhằm tạo kích thước của game.



Hàm setPreferredSize(Dimension) được dùng để thiết lập lại kích thước chương trình. Sau khi cài đặt được kích thước của chương trình, hàm addKeyListener() được gọi để đăng ký sử dụng lớp xử lý KeyEvent, và đối được sử dụng làm input của hàm này là đối tượng có kiểu KeyBroadInputs được khởi tạo mới bằng hàm tạo KeyBroadInputs(GamePanel).



Sau khi đăng ký xong lớp xử lý KeyEvent, chương trình tiếp tục gọi đến hàm addMouseListener() để đăng ký sử dụng lớp xử lý MouseEvent, và đối được sử dụng làm input của hàm này là biến mouseInputs.



Sau khi đăng ký xong lớp xử lý MouseEvent, chương trình gọi đến hàm addMouseMotionListener() để đăng ký sử dụng lớp xử lý MouseMotionEvent, và đối được sử dụng làm inout của hàm này là biến mouseInputs.



Sau khi khởi tạo xong biến gamePanel, chương trình sẽ khởi tạo biến gameWindow bằng hàm tạo GameWindow(GamePanel).



Trong hàm tạo GameWindow(GamePanel), đầu tiên, biến jframe được khởi tạo bằng hàm tạo JFrame(), sau khi khởi tạo, chúng ta cài đặt hàm tắt jframe mặc định ở trạng thái EXIT\_ON\_CLOSE, tức là sau khi JFrame bị đóng, chương trình cũng sẽ tắt theo.



Sau đó, chúng ta dùng hàm add() để thêm gamePanel vào Frame.



Sau đó, hàm setResizable(false) được gọi đến để cố định kích thước của game. Phương thức pack() sẽ tự động thay đổi kích thước của JFrame dựa trên kích thước của các component mà nó chứa kể cả các vùng được định nghĩa thêm.

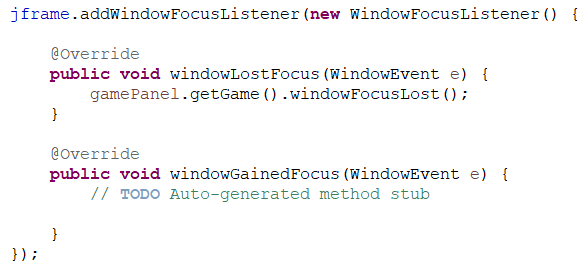
Mặc định thì JFrame sẽ hiển thị ở góc trên bên trái màn hình desktop, do vậy chương trình gọi đến hàm setLocationRelativeTo(null) để căn giữa màn hình.



Hàm setVisible(true) để hiển thị JFrame.



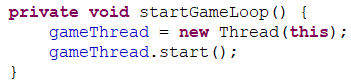
Sau đó, hàm addWindowFocusListener() được sử dụng để thay đổi trạng thái của Window.



Sau đó, hàm setFocusable() và requestFocus() được gọi đối với đối tượng gamePanel.



Cuối cùng, hàm startGameLoop() được gọi để bắt đầu game loop.

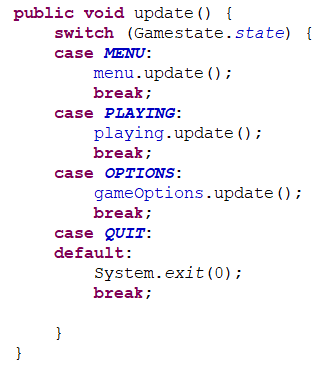


Những dòng code trên là cách chương trình bắt đầu, kết quả sẽ được màn hình menu như sau:



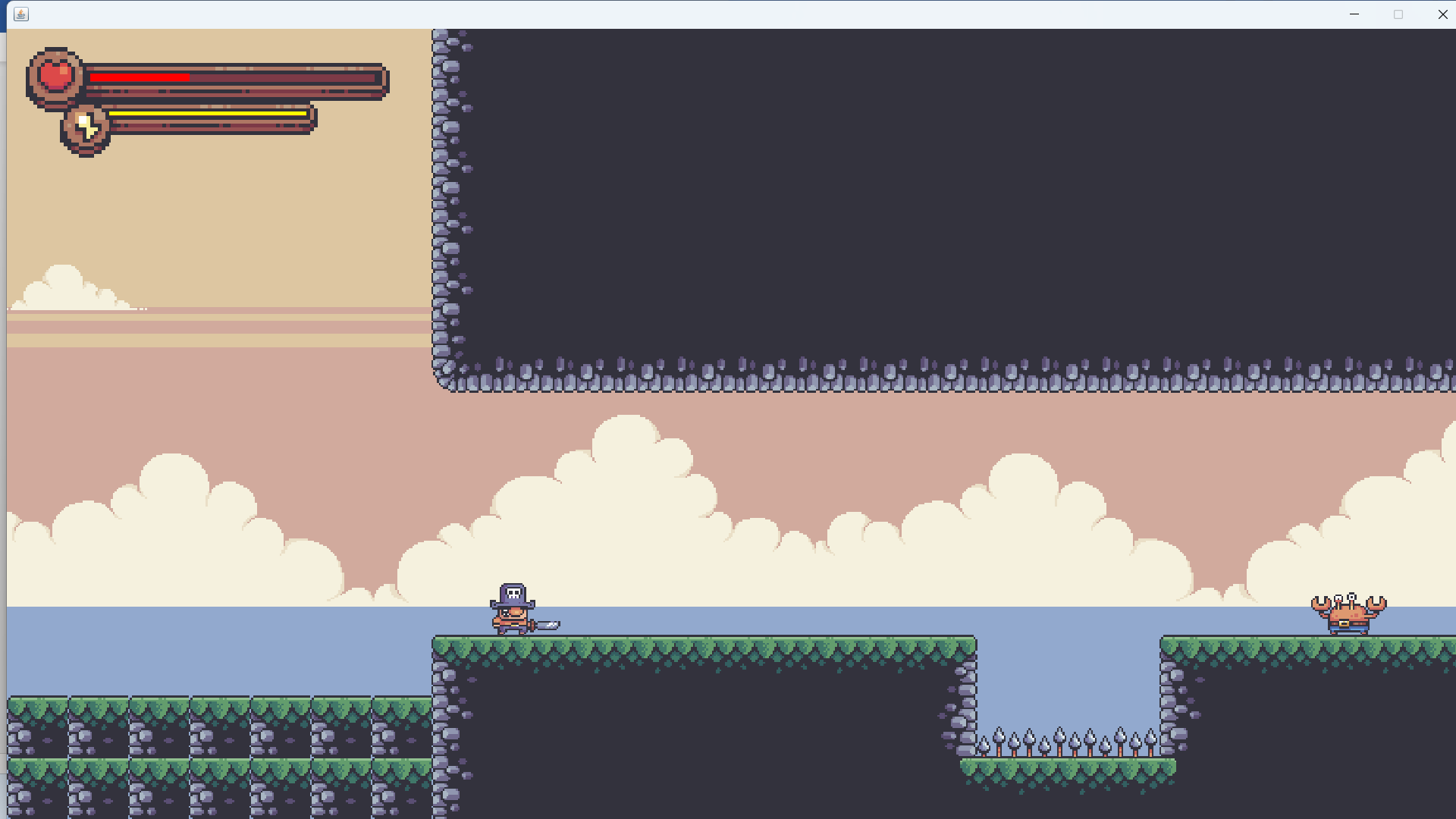
Hình 3.1: Hình nền menu

Ở Menu, chúng ta thấy có 3 lựa chọn đó là PLAY, OPTIONS và QUIT; các lựa chọn này sẽ được chạy ở trong hàm update().



Khi click chuột vào các nút thì các lựa chọn sẽ được thực thi.

* PLAYING



Hình 3.2: Hình nền trò chơi

* OPTIONS

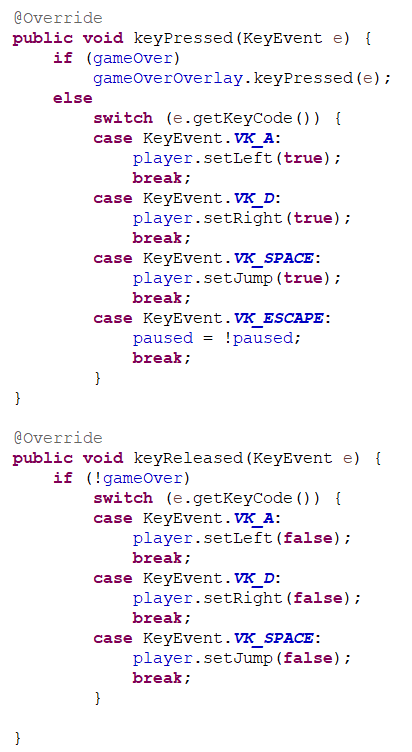


Hình 3.3: Hình nền phần cài đặt

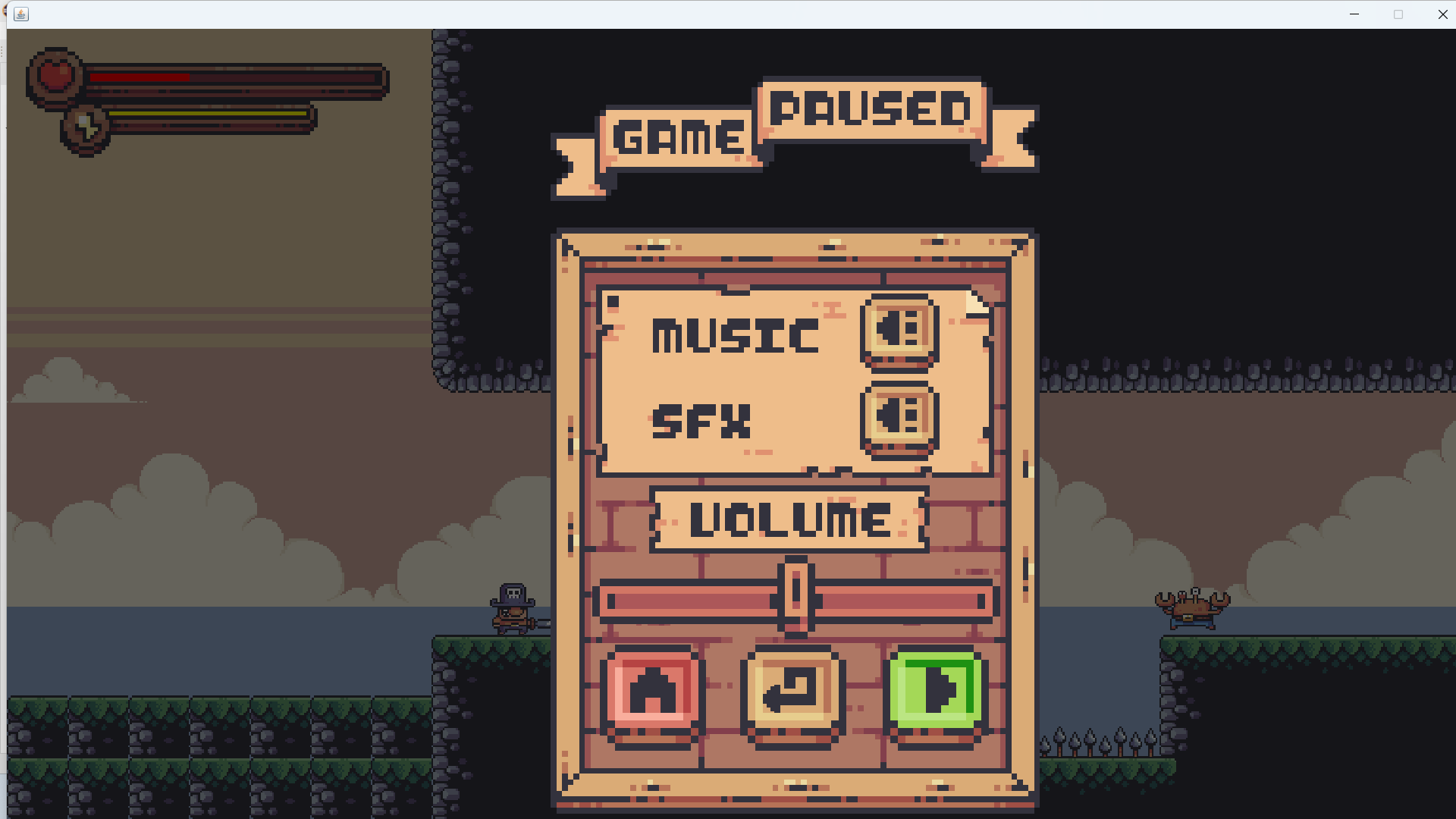
* QUIT

Khi ấn vào nút này thì chương trình sẽ kết thúc.

Khi vào trò chơi, nhân vật sẽ được điều kiển bằng cả bàn phím lẫn chuột.

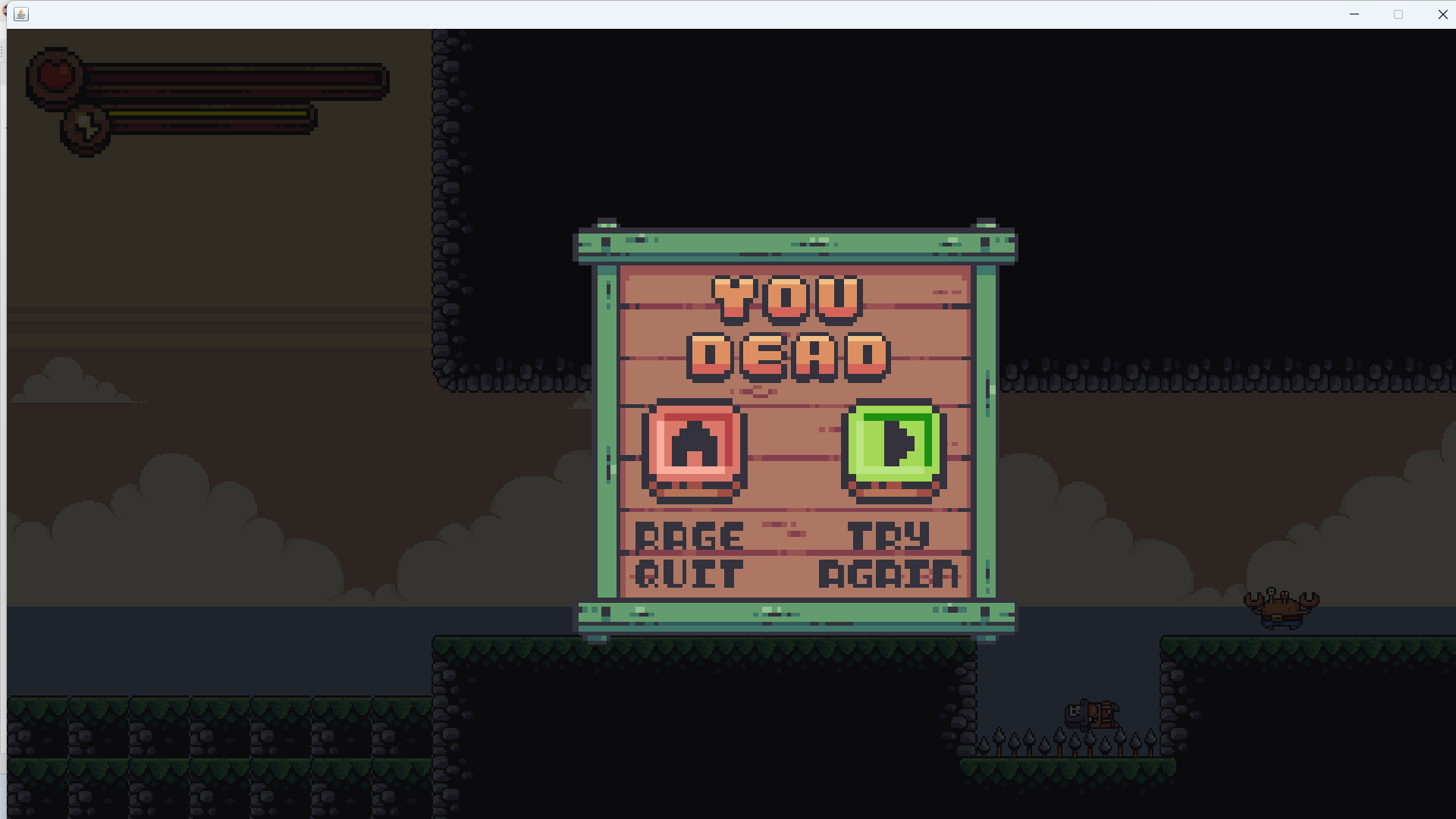


Nếu muốn tạm dừng trò chơi, người chơi ấn phím ESC và chương trình sẽ chuyển sang lớp PauseOverlay:



Hình 3.4: Hình nền khi tạm dừng trò chơi

Khi chết, chương trình sẽ tự động chuyển đến GameOverOverlay:



Hình 3.5: Hình nền khi kết thúc trò chơi

Nếu qua màn thì chương trình sẽ hiển thị lớp LevelCompletedOverlay:



Hình 3.6: Hình nền khi hoàn thành một level

1. **Kết luận và kiến nghị**

Hiện nay, lập trình game đang là một lĩnh vực hấp dẫn, thu hút nhiều lập trình viên. Việc xây dựng các ứng dụng vui chơi giải trí là rất cần thiết cho sự phát triển công nghệ.

Nhìn chung game tạo ra đáp ứng nhu cầu giải trí của người chơi. Giao diện thân thiện, đơn giản, dễ sử dụng và thao tác. Tuy nhiên, game vẫn còn có thể phát triển lên bằng cách thêm cơ sở dữ liệu để có thể lưu lại tiến trình chơi và có thể tạo nhiều tài khoản để chơi.

Tuy vậy, do những hạn chế về trình độ, thời gian nên em không thể tránh khỏi những vướng mắc, sai sót trong quá trình tìm hiểu và nghiên cứu. Em rất mong nhận được những đánh giá và chỉnh sửa chân thành từ cô.

Em xin chân thành cảm ơn!

1. **Danh mục tài liệu tham khảo**

6.1. Slide bài giảng trên lớp.

6.2. Series hướng dẫn làm game Java của youtuber Kaarin Gaming:

<https://www.kaaringaming.com/>

1. **Phụ lục**