TRƯỜNG ĐẠI HỌC GIAO THÔNG VÂN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

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



**BÁO CÁO MÔN**

**CÔNG NGHỆ JAVA**

Giảng viên hướng dẫn: **Thầy** **Huấn**

Sinh viên thực hiện:

**Hoàng Hồ Gia Khánh Mã sinh viên: 212634034**

**Phạm Quang Huy Mã sinh viên: 212613347**

Lớp: **Công Nghệ Thông Tin Việt Anh 1 – Khóa 62**

**Hà Nội, tháng 4 năm 2023**

# Lời nói đầu

Công nghệ Java là một môn học quan trọng và là nền tảng của ngành Công nghệ thông tin tại Trường Đại học Giao thông Vận tải. Nó giúp chúng ta hiểu rõ cách sử dụng một ngôn ngữ thuần lập trình hướng đối tượng. Trong môn học này, sinh viên sẽ được hướng dẫn lập trình hướng đối tượng để giải quyết các vấn đề trong đời sống và công nghiệp. Bằng cách sử dụng các kỹ thuật tính toán, cấu trúc chương trình, sinh viên sẽ có khả năng giải quyết các vấn đề với độ phức tạp khác nhau và đảm bảo tính hiệu quả và độ chính xác của phần mềm.

Sinh viên sẽ bắt đầu với những khái niệm cơ bản về công nghệ Java, sau đó tìm hiểu về các phương thức, hàm,... Hơn nữa, sinh viên sẽ được học cách áp dụng các kỹ thuật này để giải quyết các vấn đề thực tế. Môn học này giúp sinh viên có những kỹ năng quan trọng để phát triển và tối ưu hóa các chương trình máy tính, đồng thời cung cấp cho sinh viên những nền tảng và kiến thức cơ bản để phát triển các ứng dụng công nghệ thông tin trong tương lai. Chương trình này sẽ giúp sinh viên trở thành những chuyên gia trong lĩnh vực Công nghệ thông tin và đóng góp tích cực vào sự phát triển của ngành.

Em xin được tỏ lòng biết ơn sâu sắc tới thầy Huấn, người đã cống hiến thời gian và ân cần giảng dạy những kiến quan trọng này, bên cạnh đó cũng gửi lời cảm ơn đến các bạn trong lớp Công Nghệ Thông Tin Việt – Anh Khóa 62 đã chia sẻ kiến thức và hỗ trợ em trong quá trình thực hiện báo cáo.

Mục Lục

[Lời nói đầu 1](#_Toc133616201)

[I, Giới thiệu chung: 3](#_Toc133616202)

[1, Yêu cầu: 3](#_Toc133616203)

[Khi tạo ra một sản phẩm phần mềm trò chơi thì chúng ta phải có yêu cầu rõ ràng về cấu trúc cần có bao gồm một game phải được xây dựng theo lập trình hướng đối tượng và có các âm thanh, hình ảnh, nút di chuyển,.. 2, Mục đích: 3](#_Toc133616204)

[3, Ý tưởng: 3](#_Toc133616205)

[4, Code: 4](#_Toc133616206)

[A, Phần đăng nhập: 4](#_Toc133616207)

[B, Kiểm tra va trạm: 6](#_Toc133616208)

[C, Kết nối cơ sở dữ liệu: 6](#_Toc133616209)

[D, Lấy hình ảnh: 10](#_Toc133616210)

[E, Lấy nút: 11](#_Toc133616211)

[II, Tổng kết: 35](#_Toc133616212)

[1, Đánh giá chung về thuật toán: 35](#_Toc133616213)

[2, Tài liệu tham khảo: 36](#_Toc133616214)

# I, Giới thiệu chung:

Trong báo cáo này, chúng em đã làm game đua xe làm đề tài báo cáo lần này.

## 1, Yêu cầu:

## Khi tạo ra một sản phẩm phần mềm trò chơi thì chúng ta phải có yêu cầu rõ ràng về cấu trúc cần có bao gồm một game phải được xây dựng theo lập trình hướng đối tượng và có các âm thanh, hình ảnh, nút di chuyển,.. 2, Mục đích:

Tạo ra một sản phẩm thực tế bằng ngôn ngữ Java, thành thạo sử dụng phần mềm từ cơ bản đến nâng cao, hiểu rõ như thế nào là một sản phẩm hoàn thiện mà các doanh nghiệp đang phát triển.

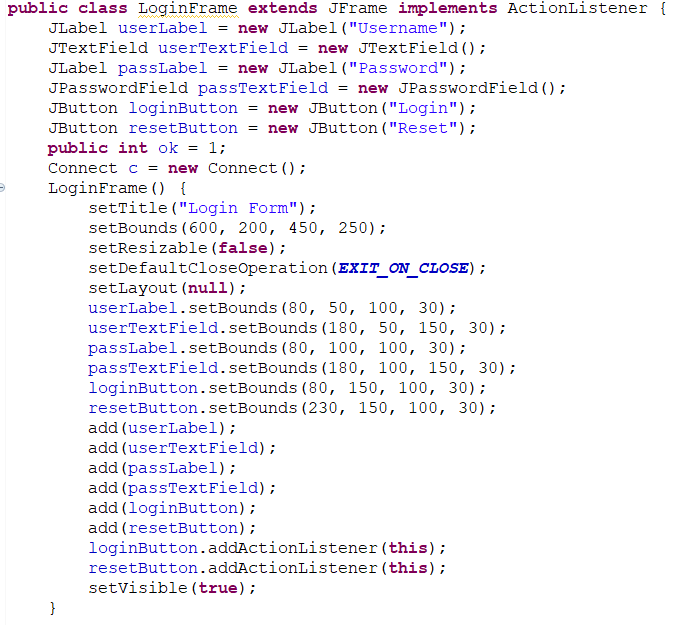
## 3, Ý tưởng:

Đây là tựa game 2D thuộc đua xe vượt chướng ngoại vật đem đến cho người chơi một trải nghiệm lái xe cực kỳ hấp dẫn và thử thách. Với các phím W A S D, người chơi có thể di chuyển chiếc xe chính của mình lên trên ,xuống, trái hoặc phải để né tránh các xe đi ngược chiều đang chạy trên 3 làn đường đầy náo nhiệt..

Cùng với sự tăng độ khó của game sau mỗi khoảng thời gian,những chướng ngại vật cũng sẽ xuất hiện ngày càng nhiều, đòi hỏi người chơi phải nhanh tay và sự tập trung cao độ,với nhiệm vụ kiếm càng nhiều điểm càng tốt. Tuy nhiên, đó chính là cơ hội để thử thách bản thân và thể hiện kỹ năng lái xe của mình.

## 4, Code:

### A, Phần đăng nhập:



Đoạn mã này định nghĩa một ứng dụng GUI (Giao diện người dùng đồ họa) trong Java, cụ thể là một biểu mẫu đăng nhập.

Lớp LoginFrame mở rộng lớp JFrame và triển khai giao diện ActionListener. Lớp JFrame đại diện cho cửa sổ chính của ứng dụng và giao diện ActionListener được sử dụng để xử lý hành động của người dùng trên các nút.

Trong lớp LoginFrame, có một số thành phần GUI được định nghĩa, bao gồm JLabel, JTextField, JPasswordField và JButton. Những thành phần này được sử dụng để hiển thị nhãn, trường văn bản, trường mật khẩu và nút trên biểu mẫu.

Hàm tạo của lớp LoginFrame đặt tiêu đề cho biểu mẫu là "Biểu mẫu đăng nhập", thiết lập kích thước cửa sổ, không cho phép thay đổi kích thước, đặt phương thức kết thúc ứng dụng khi đóng cửa sổ. Sau đó, các thành phần GUI được đặt vị trí và kích thước bằng phương thức setBounds(), và được thêm vào biểu mẫu bằng phương thức add(). Ngoài ra, hai nút loginButton và resetButton được thêm sự kiện ActionListener và được hiển thị trên biểu mẫu bằng phương thức setVisible(true).

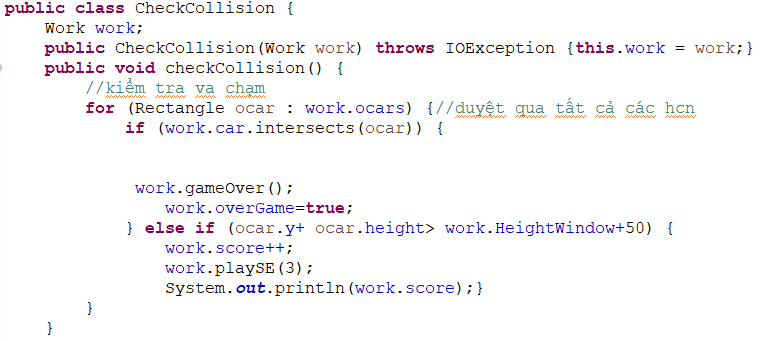


Phương thức actionPerformed được triển khai từ giao diện ActionListener để xử lý sự kiện người dùng nhấn nút trên biểu mẫu đăng nhập.

Nếu người dùng nhấn nút loginButton, phương thức sẽ lấy dữ liệu người dùng từ trường văn bản userTextField và trường mật khẩu passTextField. Sau đó, phương thức sẽ gọi phương thức returnPlayer trong lớp Connect để kiểm tra tên đăng nhập và mật khẩu. Nếu thông tin đăng nhập hợp lệ, một hộp thoại JOptionPane sẽ hiển thị thông báo "Đăng nhập thành công!" và mở một cửa sổ mới Work cho người dùng. Ngược lại, nếu thông tin đăng nhập không hợp lệ, một hộp thoại JOptionPane sẽ hiển thị thông báo "Tên đăng nhập hoặc mật khẩu không hợp lệ".

Nếu người dùng nhấn nút resetButton, phương thức sẽ xóa nội dung trong trường văn bản userTextField và trường mật khẩu passTextField.

### B, Kiểm tra va trạm:

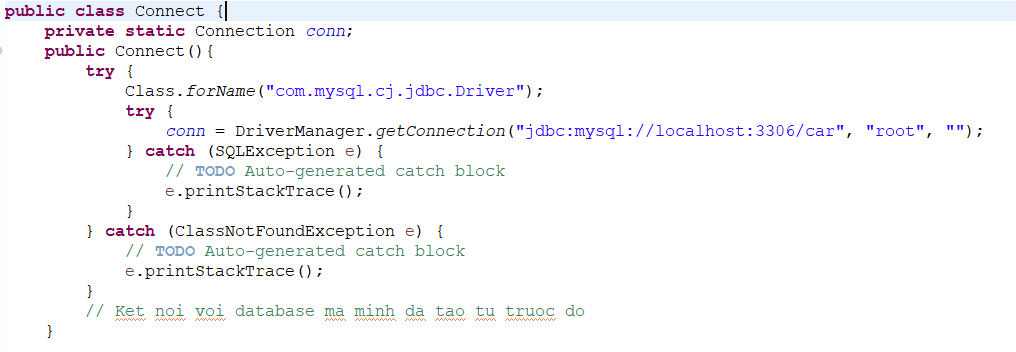


Lớp CheckCollision được sử dụng để kiểm tra va chạm giữa ô tô chơi và các ô tô khác trong trò chơi Work.

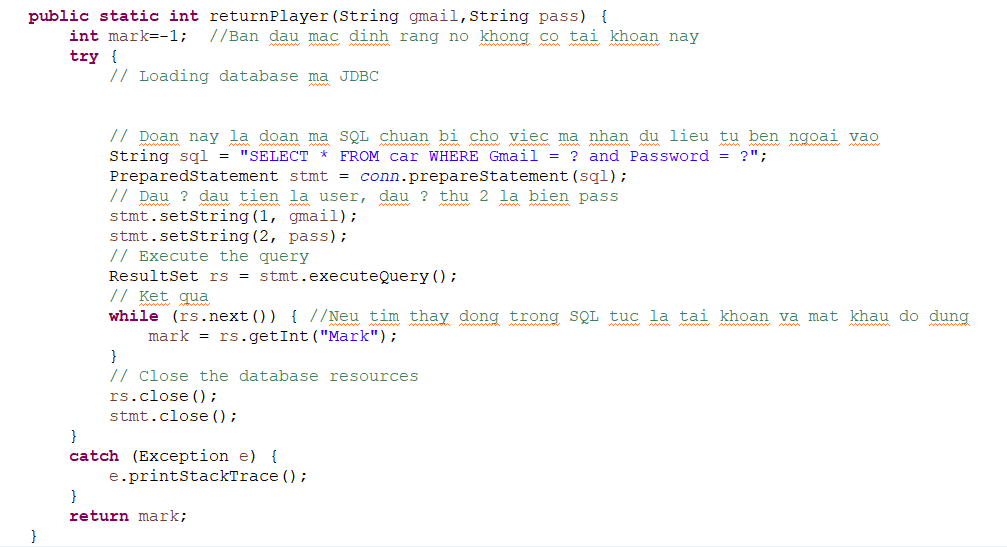
Phương thức khởi tạo CheckCollision nhận một đối tượng Work làm đối số, để truyền thông tin về ô tô và các ô tô khác trong trò chơi.

Phương thức checkCollision được sử dụng để kiểm tra va chạm giữa ô tô chơi và các ô tô khác. Vòng lặp for duyệt qua tất cả các hình chữ nhật ocar trong danh sách ocars của đối tượng work. Nếu ô tô chơi chạm vào bất kỳ hình chữ nhật nào trong danh sách, trò chơi sẽ kết thúc bằng cách gọi phương thức gameOver và đặt overGame là true. Nếu hình chữ nhật ocar vượt qua cửa sổ trò chơi, số điểm của người chơi sẽ tăng lên 1 và âm thanh được phát bằng phương thức playSE.

### C, Kết nối cơ sở dữ liệu:



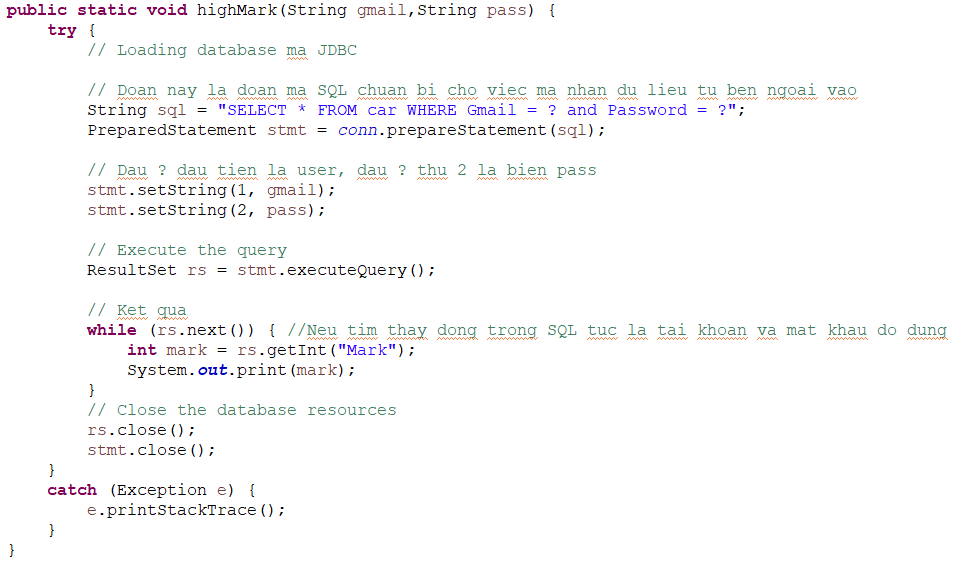
Đây là constructor của lớp Connect, nó được gọi khi một đối tượng của lớp Connect được tạo ra. Hàm này thiết lập kết nối tới cơ sở dữ liệu MySQL trên máy local sử dụng tên đăng nhập "root" và không có mật khẩu, và lưu trữ kết nối đó trong biến conn. Đầu tiên, lớp com.mysql.cj.jdbc.Driver được tải vào bộ nhớ bằng phương thức Class.forName(). Sau đó, phương thức DriverManager.getConnection() được sử dụng để tạo kết nối tới cơ sở dữ liệu. Khi một lỗi xảy ra, nó được in ra bằng phương thức printStackTrace().

****

Đây là một phương thức tĩnh (static method) trong lớp Connect. Phương thức này được sử dụng để kiểm tra xem một người dùng có tồn tại trong cơ sở dữ liệu không. Nó nhận vào hai tham số là địa chỉ email (gmail) và mật khẩu (pass) của người dùng cần kiểm tra.

Phương thức sử dụng JDBC (Java Database Connectivity) để kết nối và truy vấn cơ sở dữ liệu MySQL. Đầu tiên, nó sử dụng câu lệnh SQL để truy vấn cơ sở dữ liệu. Câu lệnh này sử dụng hai tham số ? để tránh lỗi SQL injection, và sau đó thực thi câu lệnh đó bằng cách sử dụng phương thức executeQuery().

Nếu tìm thấy một bản ghi phù hợp với địa chỉ email và mật khẩu cung cấp, phương thức sẽ trả về giá trị của trường "Mark" trong bản ghi đó (được lấy từ ResultSet). Nếu không tìm thấy bất kỳ bản ghi nào phù hợp, phương thức sẽ trả về giá trị mặc định của biến mark (-1).

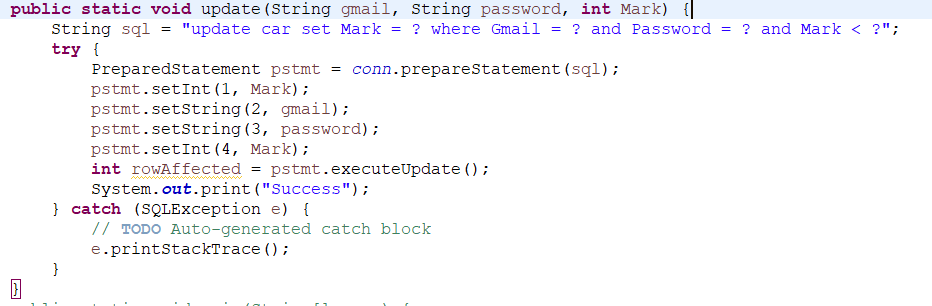


Đoạn mã trên là một phương thức Java có tên là "highMark" và có hai tham số đầu vào là "gmail" và "pass". Phương thức này được sử dụng để truy vấn cơ sở dữ liệu và trả về điểm số của người dùng có tên đăng nhập là "gmail" và mật khẩu là "pass".

Trong phương thức, đầu tiên chúng ta nạp trình điều khiển JDBC để kết nối và thực hiện các truy vấn đến cơ sở dữ liệu. Sau đó, chúng ta chuẩn bị một câu lệnh truy vấn SQL với điều kiện tìm kiếm tài khoản và mật khẩu. Các dấu "?" trong câu lệnh SQL đại diện cho các tham số đầu vào, và chúng ta sử dụng phương thức "setString" để thiết lập giá trị của các tham số này.

Sau đó, chúng ta thực thi truy vấn SQL và lấy kết quả trả về bằng cách sử dụng đối tượng "ResultSet". Nếu có kết quả trả về từ cơ sở dữ liệu, chúng ta sử dụng phương thức "getInt" để lấy giá trị của cột "Mark" và in ra điểm số tương ứng với người dùng tìm thấy.

Cuối cùng, chúng ta đóng các tài nguyên cơ sở dữ liệu và bắt các ngoại lệ (exceptions) nếu xảy ra lỗi trong quá trình truy vấn.



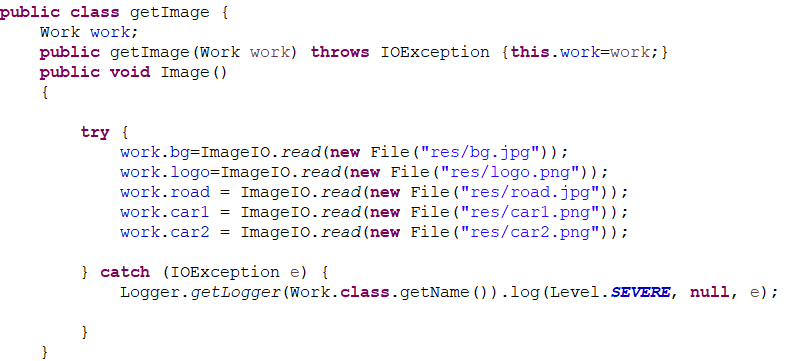
Đoạn mã trên là một phương thức Java có tên là "update" với ba tham số đầu vào: "gmail", "password" và "Mark". Phương thức này được sử dụng để cập nhật điểm số của người dùng có tên đăng nhập là "gmail" và mật khẩu là "password" trong cơ sở dữ liệu, chỉ khi điểm số mới "Mark" cao hơn điểm số cũ của người dùng.

Trong phương thức, chúng ta chuẩn bị một câu lệnh truy vấn SQL để cập nhật điểm số của người dùng trong bảng "car". Câu lệnh SQL này sử dụng toán tử "AND" để kết hợp nhiều điều kiện. Cụ thể, nó chỉ cho phép cập nhật điểm số mới nếu điểm số mới cao hơn điểm số cũ của người dùng, và nếu tên đăng nhập và mật khẩu của người dùng khớp với bản ghi trong cơ sở dữ liệu.

Sau đó, chúng ta sử dụng phương thức "setInt" và "setString" để thiết lập giá trị của các tham số trong câu lệnh SQL, tương ứng với "Mark", "gmail" và "password".

Cuối cùng, chúng ta thực thi câu lệnh SQL bằng cách sử dụng đối tượng "PreparedStatement" và phương thức "executeUpdate". Nếu thực hiện cập nhật thành công, phương thức sẽ in ra thông báo "Success". Nếu xảy ra lỗi trong quá trình thực thi truy vấn, chúng ta sẽ bắt ngoại lệ (SQLException) và in ra thông tin lỗi.

### D, Lấy hình ảnh:

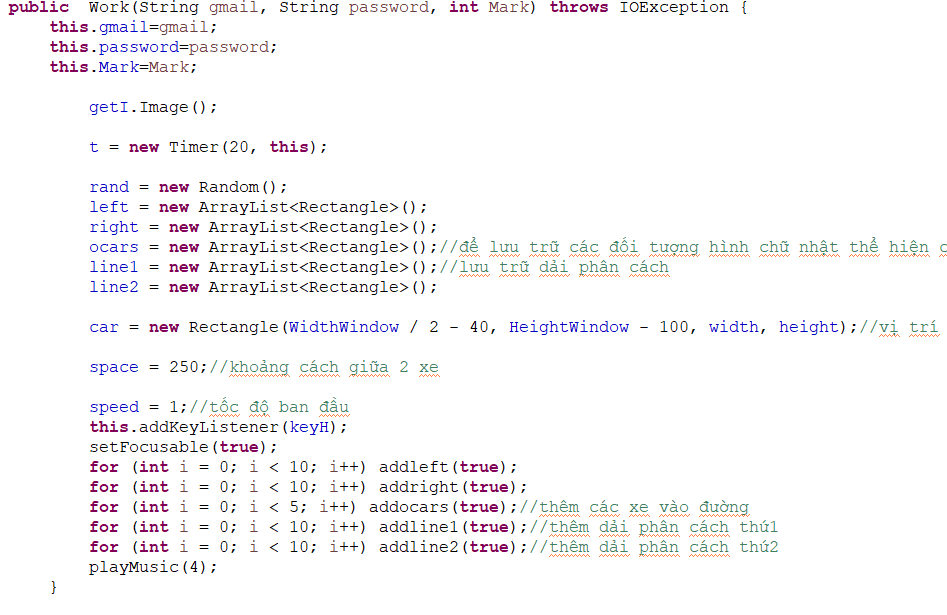


Đoạn mã trên là một lớp Java có tên là "getImage" và nó chứa một phương thức có tên là "Image". Phương thức này được sử dụng để đọc các tệp hình ảnh từ ổ đĩa và gán chúng cho các biến trong lớp "Work".

Đầu tiên, lớp "getImage" khởi tạo một đối tượng "Work" trong hàm tạo của nó và lưu trữ đối tượng "Work" này trong một thuộc tính. Sau đó, phương thức "Image" sử dụng phương thức "ImageIO.read" để đọc các tệp hình ảnh từ ổ đĩa và gán chúng cho các biến tương ứng trong đối tượng "Work".

Cụ thể, đối tượng "work" có các biến như "bg", "logo", "road", "car1" và "car2" để lưu trữ các hình ảnh của nền, logo, đường phố và hai loại ô tô khác nhau. Nếu không có lỗi xảy ra trong quá trình đọc các tệp hình ảnh, phương thức sẽ kết thúc mà không có bất kỳ thông báo nào. Nếu xảy ra lỗi trong quá trình đọc tệp hình ảnh, phương thức sẽ ghi lại lỗi đó bằng cách sử dụng Logger và Level.SEVERE.

### E, Lấy nút:



Đây là constructor của lớp Work, với các tham số là tài khoản gmail, mật khẩu, điểm số của người chơi. Constructor này được định nghĩa để khởi tạo đối tượng Work với các thuộc tính và giá trị ban đầu.

Các lệnh bên trong constructor này bao gồm:

Gọi phương thức getImage() để load ảnh lên.

Khởi tạo đối tượng Timer với thời gian delay là 20ms và đối tượng này sẽ gọi đến phương thức actionPerformed() trong lớp Work.

Khởi tạo một đối tượng Random để sinh số ngẫu nhiên.

Khởi tạo các ArrayList để lưu trữ các đối tượng hình chữ nhật thể hiện các xe di chuyển trên đường (left, right, ocars), dải phân cách (line1, line2).

Thiết lập kích thước và vị trí mặc định của xe.

Thiết lập khoảng cách giữa các xe và tốc độ ban đầu.

Thêm các xe, dải phân cách và xử lý âm thanh.



Đây là một đoạn mã trong chương trình Java để vẽ một trò chơi đua xe đơn giản. Cụ thể, các phương thức này được sử dụng để thêm các ô vuông (hoặc các "xe") vào hai bên của màn hình, cũng như thêm các "xe" mới vào đường chạy.

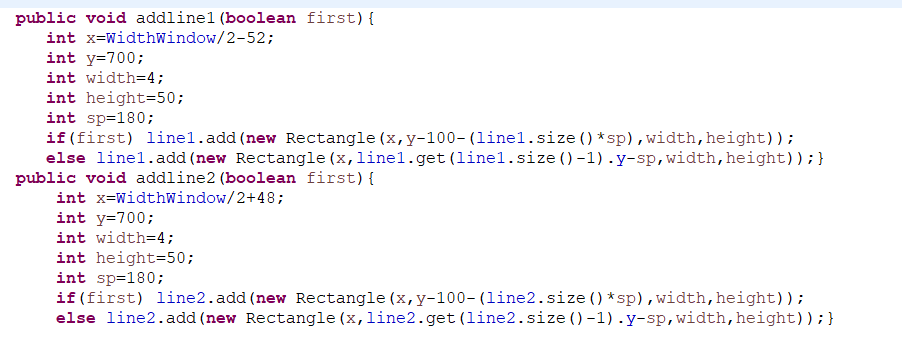
Phương thức "addleft" được sử dụng để thêm các ô vuông vào bên trái của màn hình. Nếu tham số "first" bằng true, ô vuông đầu tiên sẽ được thêm vào vị trí (50, 650), còn nếu không, ô vuông sẽ được thêm vào vị trí cuối cùng của danh sách "left" với khoảng cách "sp" (100 đơn vị) giữa chúng.

Phương thức "addright" cũng hoạt động tương tự, nhưng thay vì thêm các ô vuông vào bên trái, nó sẽ thêm chúng vào bên phải của màn hình.

Phương thức "addocars" được sử dụng để thêm các "xe" vào đường chạy. Biến "positionx" sẽ lưu trữ một số nguyên ngẫu nhiên trong khoảng từ -2 đến 1 (giá trị phần dư khi chia ngẫu nhiên cho 3), đại diện cho vị trí mà các "xe" mới sẽ được thêm vào. Nếu "positionx" bằng 0, xe mới sẽ được thêm vào trên làn đường giữa của màn hình, nếu bằng 1, xe mới sẽ được thêm vào trên làn đường phía bên trái của màn hình, còn nếu bằng 2, xe mới sẽ được thêm vào trên làn đường phía bên phải của màn hình.

Nếu tham số "first" bằng true, "xe" đầu tiên sẽ được thêm vào vị trí (x, -100) trên màn hình, còn nếu không, "xe" sẽ được thêm vào vị trí cuối cùng của danh sách "ocars" với khoảng cách "space" (100 đơn vị) giữa chúng. Vị trí của "xe" mới được tính dựa trên vị trí của "xe" cuối cùng trong danh sách "ocars".

Đây là một đoạn mã đơn giản trong trò chơi đua xe, được sử dụng để tạo ra các "xe" mới và thêm chúng vào màn hình.



Các phương thức addline1 và addline2 được sử dụng để thêm các đường line trên đường đua. Các tham số đầu vào bao gồm:

first: đánh dấu xem đây có phải là lần đầu tiên thêm đối tượng vào danh sách hay không.

Các biến và giá trị được sử dụng trong phương thức:

x: tọa độ x của đường line.

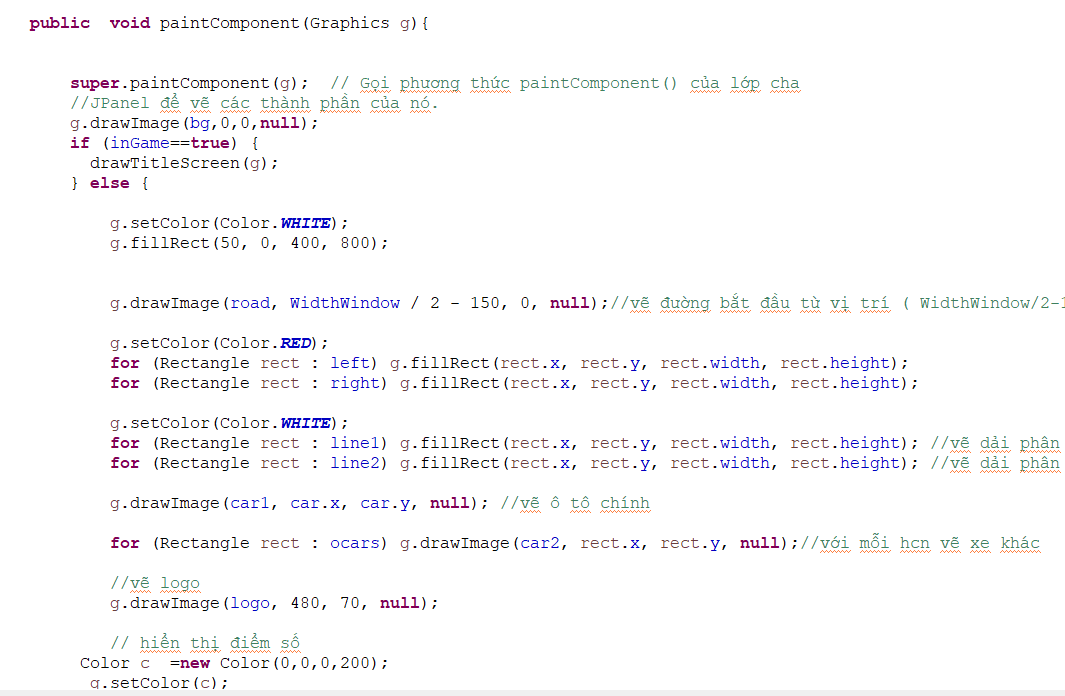
y: tọa độ y của đường line.

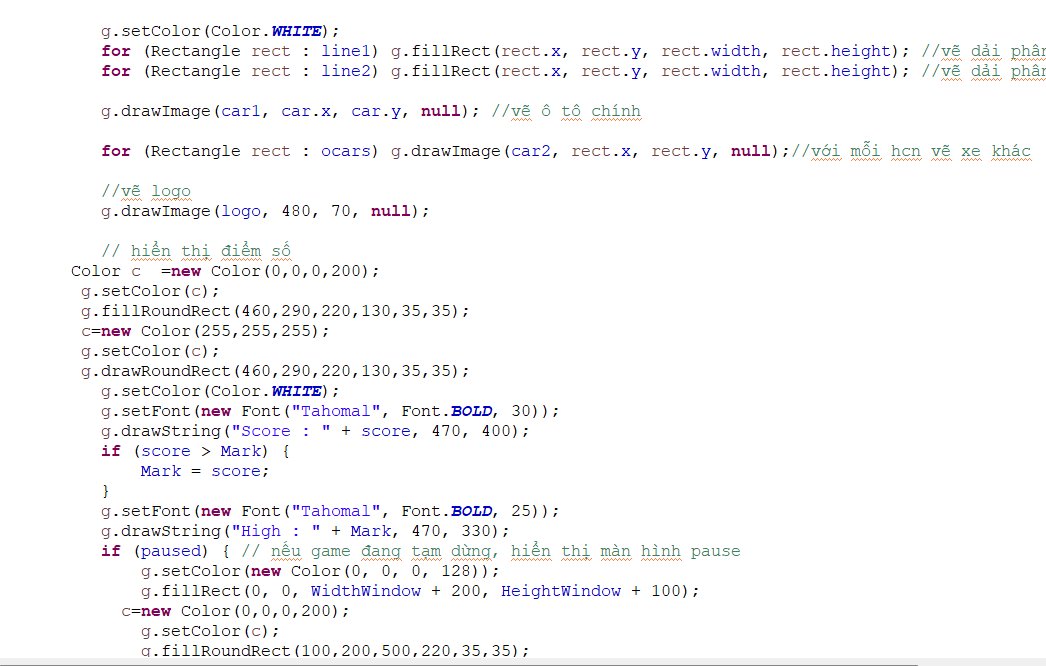
width: độ rộng của đường line.

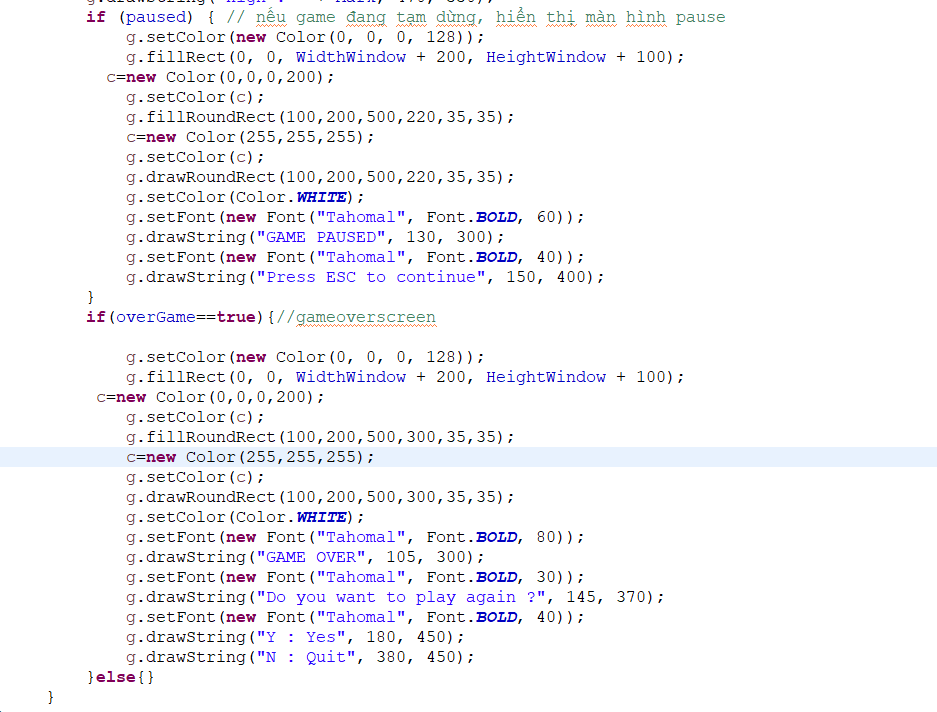
height: chiều cao của đường line.

sp: khoảng cách giữa các đường line trên đường đua.

Nếu đây là lần đầu tiên thêm đường line vào danh sách, thì phương thức sẽ tạo một đối tượng Rectangle mới với các tham số tương ứng và thêm vào danh sách tương ứng (line1 hoặc line2). Nếu không phải là lần đầu tiên, phương thức sẽ lấy thông tin về đối tượng cuối cùng trong danh sách tương ứng, tính toán tọa độ của đối tượng tiếp theo, tạo một đối tượng Rectangle mới và thêm vào danh sách.



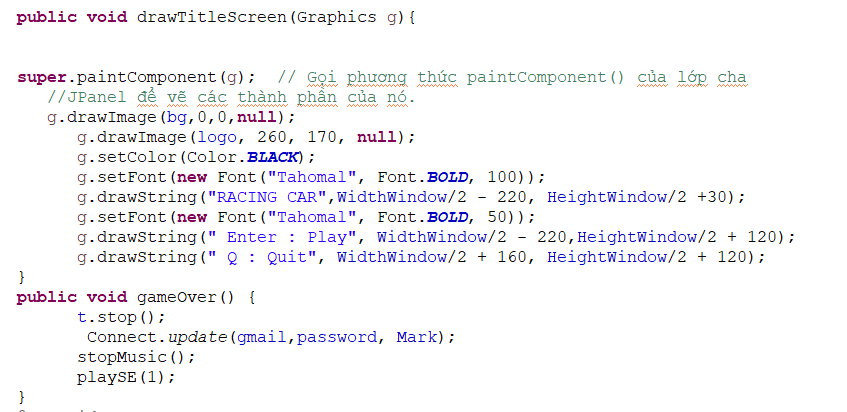




Phương thức paintComponent(Graphics g) được sử dụng để vẽ các thành phần trên màn hình của trò chơi. Trước tiên, phương thức này gọi đến phương thức paintComponent() của lớp cha để vẽ các thành phần của nó. Sau đó, nếu biến inGame bằng true, phương thức sẽ vẽ màn hình bắt đầu của trò chơi bằng cách gọi phương thức drawTitleScreen(g). Nếu không, phương thức sẽ vẽ các thành phần của trò chơi như đường, ô tô chính, ô tô khác, các dải phân cách và hiển thị điểm số trên màn hình.

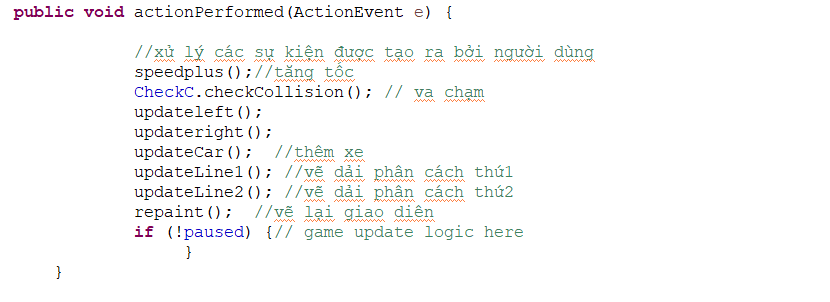
Nếu biến paused bằng true, phương thức sẽ hiển thị màn hình tạm dừng trò chơi. Nếu biến overGame bằng true, phương thức sẽ hiển thị màn hình kết thúc trò chơi và cho phép người chơi chơi lại hoặc thoát khỏi trò chơi.

Trong phương thức này, được sử dụng các phương thức của đối tượng Graphics để vẽ các hình ảnh và đối tượng trên màn hình. Các biến đại diện cho các hình ảnh và các đối tượng trong trò chơi được sử dụng để vẽ các thành phần trên màn hình.



Phương thức drawTitleScreen được sử dụng để vẽ màn hình chờ của trò chơi. Nó vẽ ảnh nền và logo, sau đó vẽ tiêu đề "RACING CAR" và hai lựa chọn "Enter : Play" và "Q : Quit" trên màn hình.

Phương thức gameOver được gọi khi người chơi thua cuộc trong trò chơi. Nó dừng chuyển động của các đối tượng trong trò chơi, cập nhật điểm số cao nhất bằng phương thức Connect.update, dừng nhạc nền và phát ra âm thanh đánh dấu kết thúc trò chơi.





Đây là một phương thức xử lý sự kiện trong Java, được thực hiện khi người dùng tương tác với giao diện đồ họa của chương trình. Phương thức này bao gồm một loạt các hoạt động được thực hiện trong một vòng lặp.

Các hoạt động bao gồm:

speedplus(): Phương thức này tăng tốc độ của các xe và các đường phân cách, sau đó di chuyển chúng theo tốc độ mới. Nó cũng cập nhật biến count để đếm số lần phương thức đã được gọi.

CheckC.checkCollision(): Kiểm tra xem có va chạm giữa các xe không.

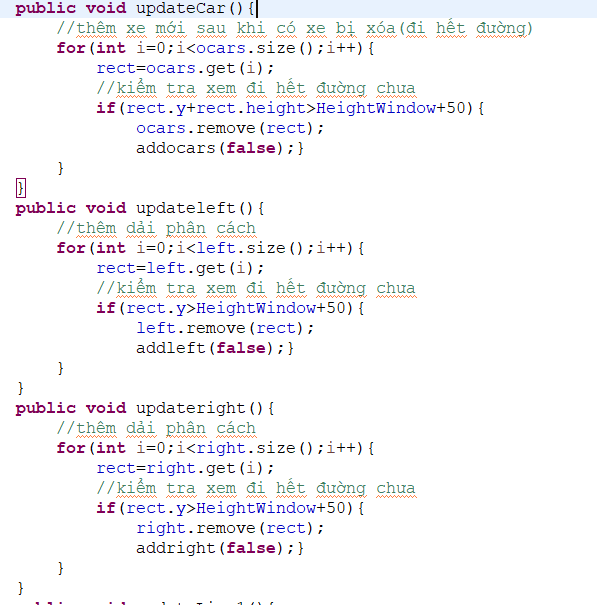
updateleft(), updateright(): Cập nhật vị trí của các xe bên trái và bên phải.

updateCar(): Thêm một số xe mới vào game.

updateLine1(), updateLine2(): Cập nhật vị trí của các đường phân cách.

repaint(): Vẽ lại giao diện.

Nếu biến paused bằng false, các hoạt động được thực hiện trong một vòng lặp khác để cập nhật trạng thái của game.

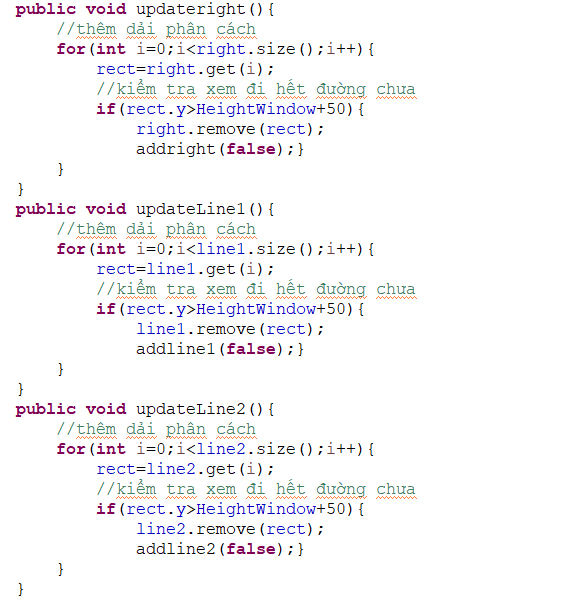


Đây là một số phương thức để cập nhật các đối tượng trong game, bao gồm các phương thức updateCar(), updateleft(), và updateright(). Mỗi phương thức này thực hiện một nhiệm vụ riêng biệt để cập nhật trạng thái của các đối tượng.

Phương thức updateCar() được sử dụng để kiểm tra xem các xe trong game đã đi hết đường hay chưa. Nếu một xe đã đi hết đường, phương thức này sẽ loại bỏ nó khỏi danh sách các xe và thêm một xe mới vào game bằng phương thức addocars(false).

Phương thức updateleft() và updateright() được sử dụng để kiểm tra xem các đường phân cách trong game đã đi hết đường hay chưa. Nếu một dải phân cách đã đi hết đường, phương thức này sẽ loại bỏ nó khỏi danh sách và thêm một dải phân cách mới vào game bằng phương thức addleft(false) hoặc addright(false) tương ứng.

Các phương thức addocars(false), addleft(false), và addright(false) được sử dụng để thêm các đối tượng mới vào game. Tham số false được truyền vào để chỉ định rằng các đối tượng này sẽ không xuất hiện ngay lập tức mà sẽ được di chuyển từ phía trên cùng của màn hình xuống vị trí ban đầu của chúng.



Các phương thức updateRight(), updateLine1(), và updateLine2() đều có mục đích kiểm tra xem các đối tượng đã di chuyển qua màn hình chưa, nếu đã di chuyển hết thì loại bỏ đối tượng đó và thêm một đối tượng mới vào.

Trong phương thức updateRight(), vòng lặp sẽ duyệt qua danh sách các đối tượng dải phân cách bên phải và kiểm tra xem đối tượng đó đã di chuyển hết khỏi màn hình chưa (tọa độ y của đối tượng lớn hơn chiều cao của màn hình cộng thêm 50). Nếu đã di chuyển hết thì đối tượng đó sẽ được loại bỏ và một đối tượng mới sẽ được thêm vào danh sách.

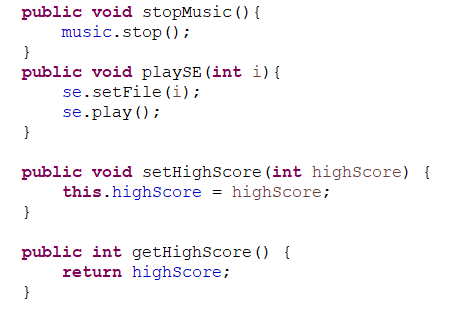
Tương tự như vậy, phương thức updateLine1() và updateLine2() cũng thực hiện kiểm tra và xử lý các đối tượng dải phân cách thứ nhất và thứ hai tương ứng.



Phương thức togglePause() sẽ tạm dừng hoặc tiếp tục trò chơi. Nếu trò chơi đang tạm dừng, nó sẽ bật nhạc và khởi động lại bộ đếm thời gian. Nếu trò chơi đang chạy, nó sẽ dừng nhạc và tạm dừng bộ đếm thời gian.

Phương thức restart() sẽ khởi động lại trò chơi bằng cách đặt lại tất cả các biến và các đối tượng trong trò chơi về giá trị mặc định ban đầu. Nó cũng khởi động lại nhạc, tạo ra các đối tượng mới và bắt đầu bộ đếm thời gian.

Phương thức playMusic(int i) sẽ chơi nhạc trong trò chơi bằng cách truyền vào một số nguyên i để chỉ định loại nhạc cụ thích hợp. Sau đó, nó sẽ bật nhạc và lặp lại chơi lại từ đầu cho đến khi trò chơi kết thúc.



Phương thức stopMusic() sẽ dừng nhạc trong trò chơi.

Phương thức playSE(int i) sẽ phát ra âm thanh hiệu ứng (sound effect - SE) trong trò chơi bằng cách truyền vào một số nguyên i để chỉ định loại hiệu ứng âm thanh cụ thể.

Phương thức setHighScore(int highScore) sẽ đặt giá trị điểm số cao nhất (high score) của trò chơi bằng cách truyền vào một số nguyên highScore.

Phương thức getHighScore() sẽ trả về giá trị điểm số cao nhất (high score) của trò chơi.



Đây là một đoạn mã Java để phát các tệp âm thanh trong ứng dụng.

Lớp "Sound" có ba thuộc tính là "clip" để đại diện cho âm thanh được phát, "soundURL" để lưu trữ đường dẫn tới các tệp âm thanh, và "fc" để điều chỉnh âm lượng.

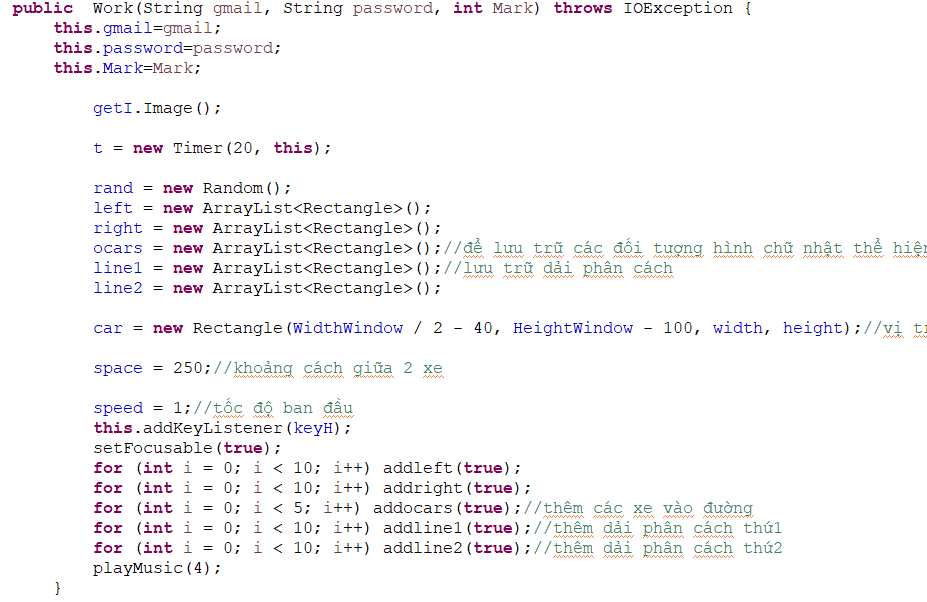
Hàm khởi tạo của lớp Sound gán các đường dẫn tới các tệp âm thanh vào mảng "soundURL".

Phương thức "setFile" được sử dụng để chọn tệp âm thanh được phát. Nó tải tệp âm thanh tương ứng với chỉ số được truyền vào, tạo ra một đối tượng AudioInputStream từ tệp âm thanh và tạo một đối tượng Clip từ đối tượng AudioSystem. Sau đó, nó mở Clip với AudioInputStream và gán đối tượng FloatControl cho "fc" để điều chỉnh âm lượng.

Phương thức "play" được sử dụng để phát âm thanh.

Phương thức "loop" được sử dụng để lặp lại phát lại âm thanh liên tục.

Phương thức "stop" được sử dụng để dừng phát lại âm thanh.



Đây là constructor của class Work với các tham số đầu vào là gmail, password và Mark. Constructor này được thiết kế để khởi tạo đối tượng Work và các thuộc tính liên quan đến trò chơi đua xe.

Constructor bao gồm các bước sau:

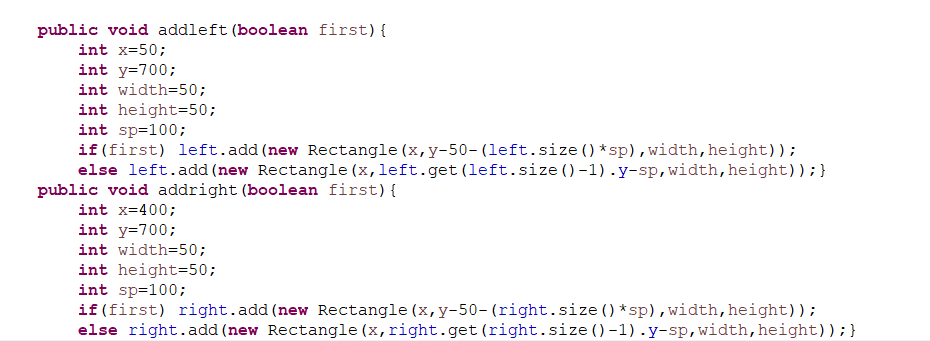
Gán giá trị cho các thuộc tính của đối tượng như gmail, password, Mark, space, speed và kích thước và vị trí của xe.

Tạo mới một đối tượng Timer với tần số 20ms, đây là đối tượng dùng để cập nhật trạng thái của trò chơi.

Khởi tạo các danh sách ArrayList để lưu trữ các xe di chuyển trên đường, các dải phân cách trên đường.

Thêm các xe và các dải phân cách vào đường.

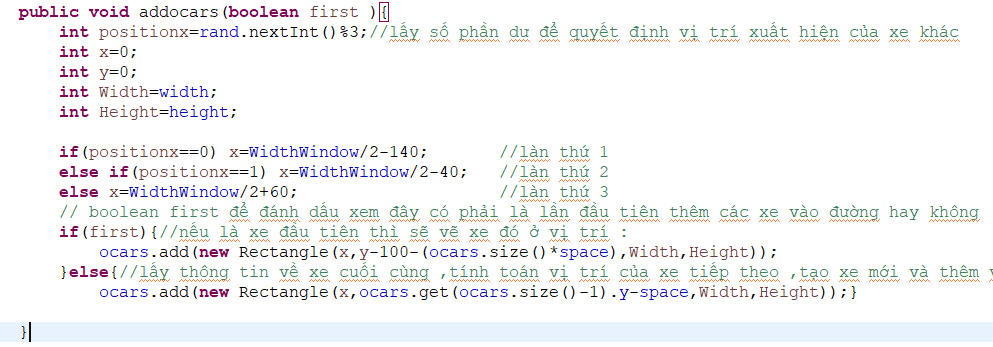
Phát nhạc nền của trò chơi.



Các phương thức addleft() và addright() được sử dụng để thêm các đối tượng hình chữ nhật thể hiện các xe di chuyển trên đường.

Phương thức addleft() thêm một đối tượng hình chữ nhật vào ArrayList left, ở vị trí cố định (x=50, y=700) và có kích thước (width=50, height=50). Các đối tượng hình chữ nhật được thêm theo khoảng cách sp=100 giữa chúng, bắt đầu từ vị trí y=650 và tăng dần khi thêm các đối tượng tiếp theo.

Tương tự, phương thức addright() thêm các đối tượng vào ArrayList right, ở vị trí cố định khác (x=400, y=700) và cũng có kích thước và khoảng cách tương tự.



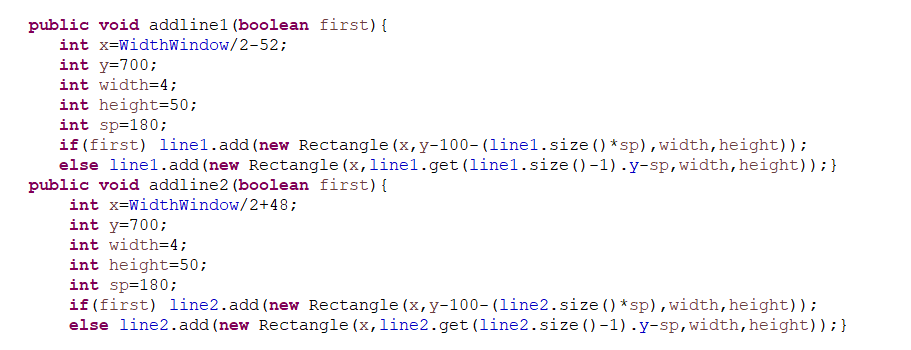
Phương thức addocars(boolean first) được viết bằng ngôn ngữ Java và được sử dụng để thêm các đối tượng xe vào trên một đường.

Đầu tiên, phương thức này sẽ lấy một số ngẫu nhiên để xác định vị trí xuất hiện của các xe trên đường. Giá trị này sẽ được lấy phần dư của số ngẫu nhiên chia cho 3 để quyết định vị trí xuất hiện của xe trên 3 làn đường khác nhau.

Tiếp theo, phương thức sẽ khởi tạo giá trị cho các biến x, y, Width và Height. Biến x sẽ được tính toán dựa trên vị trí của các làn đường và kích thước của cửa sổ hiển thị.

Nếu đây là lần đầu tiên thêm xe vào đường (biến first được truyền vào với giá trị true), phương thức sẽ thêm một hình chữ nhật mới vào danh sách ocars với kích thước và vị trí được tính toán dựa trên giá trị của biến x, y, Width và Height.

Nếu không phải là lần đầu tiên thêm xe vào đường, phương thức sẽ tính toán vị trí của xe tiếp theo dựa trên vị trí của xe cuối cùng đã được thêm vào trong danh sách ocars. Sau đó, phương thức sẽ tạo một hình chữ nhật mới và thêm vào danh sách ocars với kích thước và vị trí được tính toán dựa trên giá trị của biến x, y, Width và Height.



Đây là hai phương thức addline1(boolean first) và addline2(boolean first) được viết bằng ngôn ngữ Java và được sử dụng để thêm các dòng kẻ vạch trên đường.

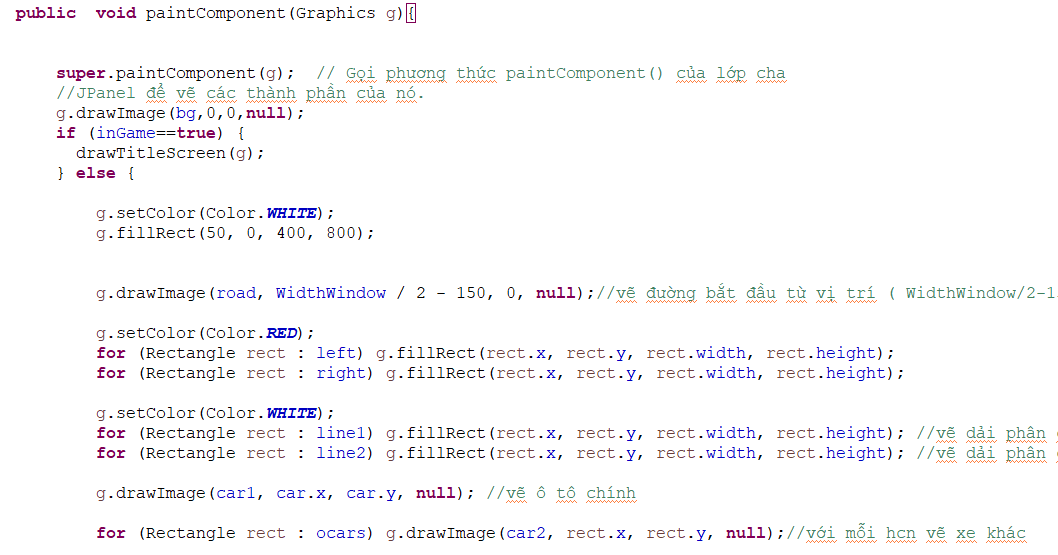
Phương thức addline1(boolean first) sẽ thêm các đối tượng hình chữ nhật vào danh sách line1, mô tả các dòng kẻ vạch trên làn đường bên trái.

Đầu tiên, phương thức này sẽ khởi tạo giá trị cho biến x, y, width, height và sp (khoảng cách giữa các đối tượng hình chữ nhật). Biến x được tính toán để đặt các đối tượng hình chữ nhật ở giữa màn hình, còn biến y sẽ được tính toán dựa trên độ dài của danh sách line1 (nếu là lần đầu tiên thêm đối tượng vào danh sách thì sẽ có giá trị ban đầu của biến y, còn nếu không phải thì sẽ lấy vị trí của đối tượng cuối cùng đã được thêm vào danh sách và tính toán vị trí của đối tượng tiếp theo dựa trên khoảng cách giữa chúng).

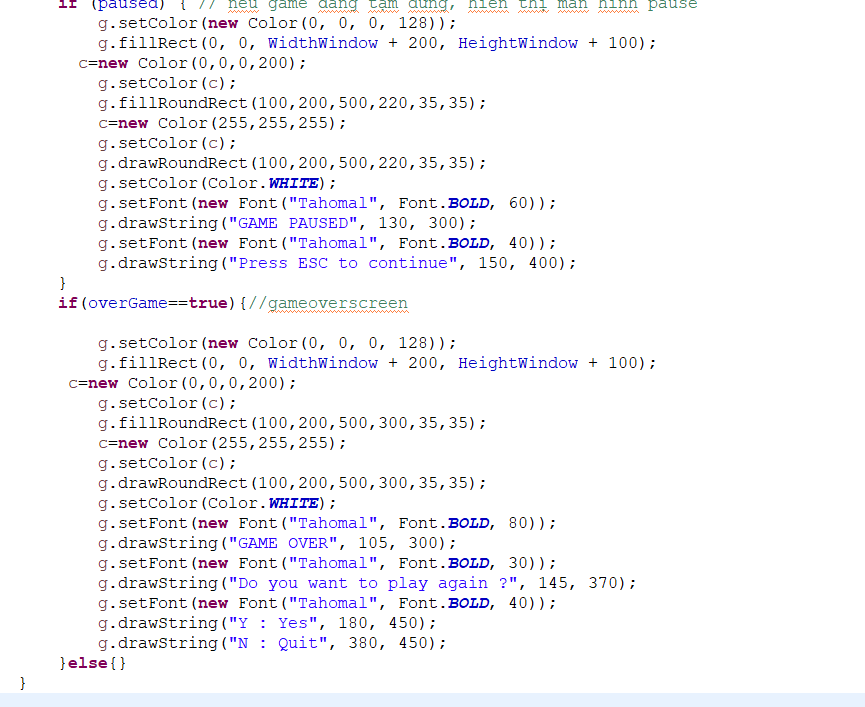
Nếu đây là lần đầu tiên thêm đối tượng vào danh sách line1 (biến first được truyền vào với giá trị true), phương thức sẽ thêm một hình chữ nhật mới vào danh sách line1 với kích thước và vị trí được tính toán dựa trên giá trị của biến x, y, width và height.

Nếu không phải lần đầu tiên thêm đối tượng vào danh sách, phương thức sẽ tính toán vị trí của đối tượng tiếp theo dựa trên vị trí của đối tượng cuối cùng đã được thêm vào danh sách line1. Sau đó, phương thức sẽ tạo một hình chữ nhật mới và thêm vào danh sách line1 với kích thước và vị trí được tính toán dựa trên giá trị của biến x, y, width và height.

Phương thức addline2(boolean first) tương tự như phương thức addline1(boolean first), nhưng được sử dụng để thêm các đối tượng hình chữ nhật vào danh sách line2, mô tả các dòng kẻ vạch trên làn đường bên phải.





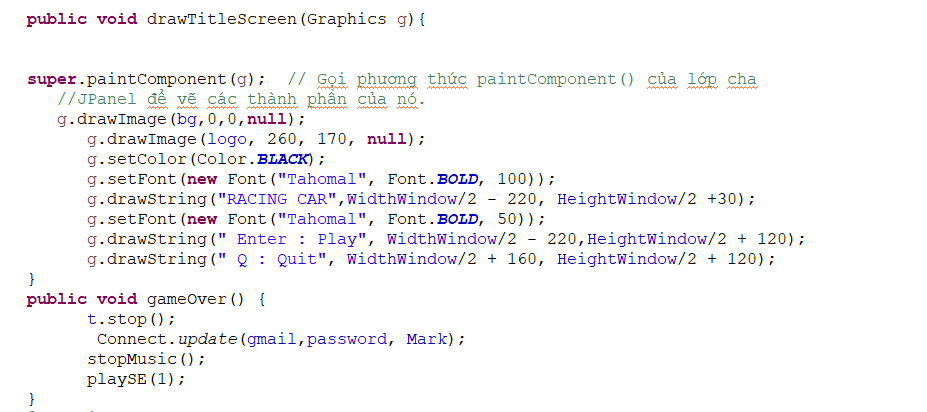


Phương thức paintComponent() được sử dụng để vẽ các thành phần của một JPanel. Trong phương thức này, đầu tiên nó gọi phương thức paintComponent() của lớp cha để vẽ các thành phần của nó. Sau đó, nó vẽ hình ảnh nền và kiểm tra nếu trò chơi đang trong trạng thái chơi thì vẽ màn hình chơi, nếu không thì vẽ màn hình kết thúc trò chơi.

Nếu trò chơi đang chơi, phương thức vẽ đường, các ô tô, dải phân cách, logo và điểm số. Điểm số được hiển thị và được so sánh với điểm cao nhất (Mark) và nếu điểm số vượt qua Mark, thì Mark được cập nhật.

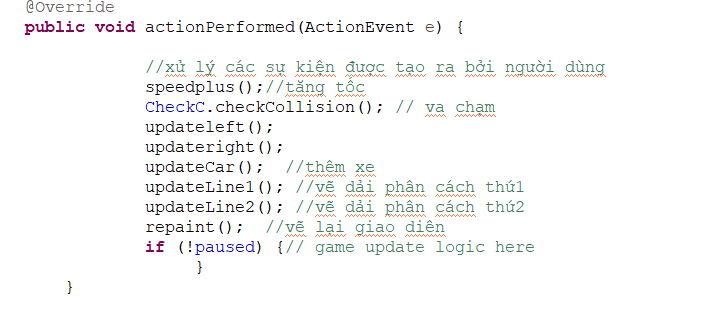
Nếu trò chơi bị tạm dừng, màn hình tạm dừng sẽ được hiển thị với nút tiếp tục được yêu cầu nhấn.

Nếu trò chơi kết thúc, màn hình kết thúc sẽ được hiển thị với tùy chọn chơi lại hoặc thoát game.



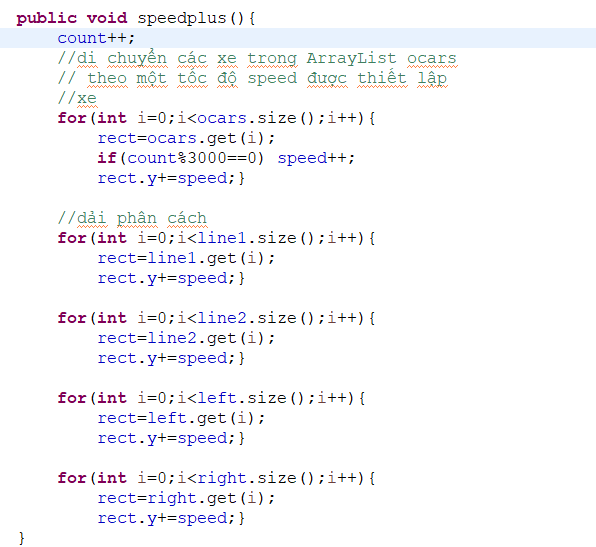
Phương thức "drawTitleScreen(Graphics g)" được sử dụng để vẽ màn hình tiêu đề cho trò chơi. Trong đó, phương thức "super.paintComponent(g)" được gọi để vẽ các thành phần của JPanel. Sau đó, hình ảnh nền và hình ảnh logo được vẽ lên màn hình bằng cách sử dụng phương thức "g.drawImage()". Tiếp theo, văn bản "RACING CAR" được vẽ lên màn hình với kích thước chữ là 100 và kiểu chữ đậm bằng phương thức "g.drawString()". Cuối cùng, văn bản "Enter: Play" và "Q: Quit" được vẽ lên màn hình với kích thước chữ là 50 và kiểu chữ đậm bằng phương thức "g.drawString()".

Phương thức "gameOver()" được sử dụng để dừng trò chơi khi người chơi đã thua. Trong đó, đối tượng "t" đang chạy sẽ được dừng lại bằng phương thức "t.stop()". Sau đó, điểm số sẽ được cập nhật và lưu trữ thông qua phương thức "Connect.update(gmail,password, Mark)" để đưa lên máy chủ. Tiếp theo, âm thanh đang chạy sẽ được dừng lại bằng phương thức "stopMusic()" và âm thanh hiệu ứng 1 sẽ được phát ra bằng phương thức "playSE(1)".



Phương thức "actionPerformed(ActionEvent e)" được sử dụng để xử lý các sự kiện được tạo ra bởi người dùng, bao gồm các phương thức để tăng tốc độ, kiểm tra va chạm, cập nhật vị trí của xe và các dải phân cách, và vẽ lại giao diện với phương thức "repaint()".

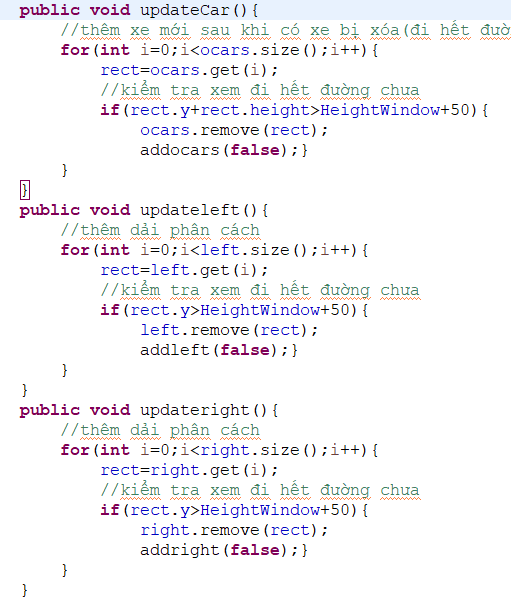
Trong phần điều kiện cuối cùng của phương thức, nếu trò chơi không bị tạm dừng, thì các phương thức cập nhật trò chơi sẽ được gọi để thực hiện logic của trò chơi. Tuy nhiên, phần này bị thiếu nội dung cụ thể về các phương thức được gọi để cập nhật trò chơi.



Phương thức "speedplus()" được sử dụng để tăng tốc độ của trò chơi và di chuyển các thành phần của trò chơi, bao gồm xe, dải phân cách, và các đối tượng bên trái và phải màn hình.

Trong phương thức này, biến "count" được tăng lên 1 đơn vị. Sau đó, với mỗi xe trong danh sách các xe "ocars", tốc độ của xe được cập nhật bằng cách thêm giá trị của biến "speed" vào vị trí y của xe. Các dải phân cách, đối tượng bên trái và bên phải màn hình cũng được di chuyển bằng cách thêm giá trị của biến "speed" vào vị trí y của chúng.

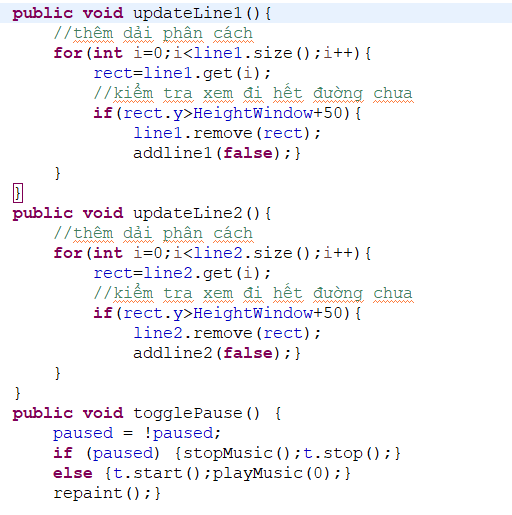
Nếu biến "count" chia hết cho 3000, tốc độ của trò chơi sẽ được tăng lên 1 đơn vị.



Các phương thức "updateCar()", "updateleft()" và "updateright()" được sử dụng để thêm xe và các dải phân cách mới vào trò chơi.

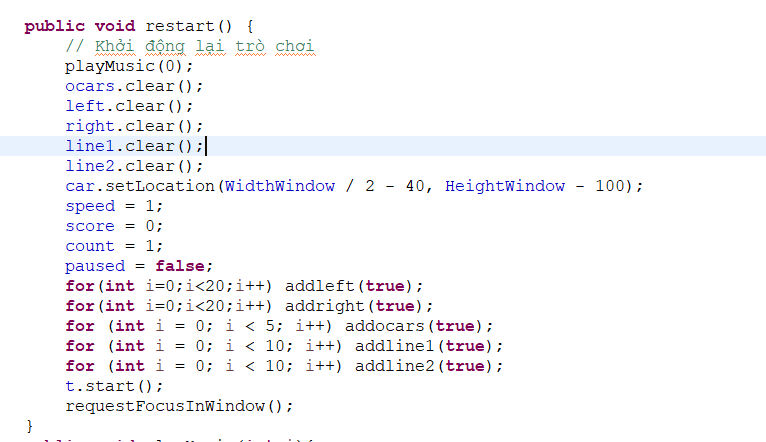
Trong phương thức "updateCar()", với mỗi xe trong danh sách các xe "ocars", phương thức kiểm tra xem xe đã đi hết đường hay chưa. Nếu vị trí y của xe cộ cộng với chiều cao của nó lớn hơn chiều cao của cửa sổ trò chơi cộng thêm 50, xe sẽ được xóa khỏi danh sách "ocars" và một xe mới sẽ được thêm vào danh sách "ocars" bằng phương thức "addocars()".

Tương tự, trong phương thức "updateleft()" và "updateright()", với mỗi dải phân cách trong danh sách tương ứng, phương thức kiểm tra xem dải phân cách đã đi hết đường hay chưa. Nếu vị trí y của dải phân cách lớn hơn chiều cao của cửa sổ trò chơi cộng thêm 50, dải phân cách sẽ được xóa khỏi danh sách và một dải phân cách mới sẽ được thêm vào danh sách bằng phương thức tương ứng "addleft()" hoặc "addright()".



Phương thức updateLine1() và updateLine2() được sử dụng để thêm dải phân cách vào màn hình và di chuyển chúng lên trên. Nếu dải phân cách đi qua đỉnh màn hình thì phương thức sẽ gỡ bỏ chúng khỏi danh sách dải phân cách và thêm một dải mới vào màn hình.

Phương thức togglePause() được sử dụng để tạm dừng hoặc tiếp tục trò chơi. Nếu trò chơi đang được chạy, thì phương thức sẽ dừng âm nhạc và đồng hồ đếm thời gian và ngược lại. Khi phương thức được gọi, nó cũng gọi phương thức repaint() để vẽ lại giao diện người dùng.



Phương thức "restart" trong đoạn mã này được sử dụng để khởi động lại trò chơi. Khi được gọi, nó sẽ đặt lại tất cả các giá trị và thuộc tính cho trò chơi và bắt đầu lại vòng lặp chính để chơi trò chơi. Cụ thể, phương thức này sẽ thực hiện các công việc sau:

Phát nhạc lại từ đầu.

Xóa các xe (ocars), các đường dây trái (left), các đường dây phải (right), dòng 1 (line1) và dòng 2 (line2) đã có.

Đặt lại vị trí của xe người chơi (car) ở giữa màn hình.

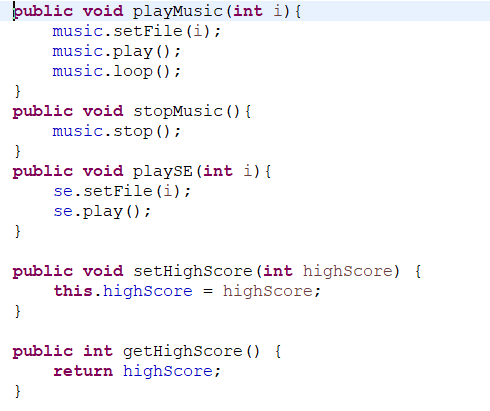
Thiết lập tốc độ và điểm số ban đầu là 1 và 0.

Thiết lập trạng thái "paused" (tạm dừng) là "false".

Thêm 20 xe, 20 đường dây trái, 20 đường dây phải, 5 xe khác và 10 dòng 1, 10 dòng 2 mới vào trò chơi.

Bắt đầu bộ đếm thời gian (t).

Yêu cầu tập trung vào cửa sổ trò chơi.



Phương thức "playMusic(int i)" là phương thức để phát nhạc. Đầu vào của phương thức là một số nguyên "i", đại diện cho tệp âm thanh muốn phát. Phương thức này sẽ đặt tệp âm thanh cho đối tượng "music" và sau đó phát và lặp lại nó.

Phương thức "stopMusic()" là phương thức để dừng phát nhạc. Phương thức này dừng lại đối tượng "music" đang phát.

Phương thức "playSE(int i)" là phương thức để phát hiệu ứng âm thanh. Đầu vào của phương thức là một số nguyên "i", đại diện cho tệp âm thanh muốn phát. Phương thức này sẽ đặt tệp âm thanh cho đối tượng "se" và sau đó phát nó.

Phương thức "setHighScore(int highScore)" là phương thức để đặt điểm số cao nhất. Đầu vào của phương thức là một số nguyên "highScore", đại diện cho điểm số muốn đặt. Phương thức này sẽ đặt giá trị của biến "highScore" cho giá trị đầu vào.

Phương thức "getHighScore()" là phương thức để lấy điểm số cao nhất. Phương thức này trả về giá trị của biến "highScore".

# II, Tổng kết:

## 1, Đánh giá chung về thuật toán:

Java là một ngôn ngữ lập trình phổ biến được sử dụng để phát triển game. Với các tính năng bảo mật và khả năng tái sử dụng mã lập trình, Java được ưa chuộng trong việc phát triển các trò chơi trực tuyến và game di động.

Một trong những lợi ích của việc sử dụng Java để phát triển game là tính di động. Java có thể chạy trên nhiều nền tảng khác nhau, cho phép game được phát triển một lần và chạy trên nhiều hệ thống khác nhau. Điều này giúp tiết kiệm thời gian và chi phí phát triển, đặc biệt là đối với các game di động.

Một ưu điểm khác của Java là tính linh hoạt. Java có thể được sử dụng để phát triển nhiều loại game khác nhau, từ các trò chơi đơn giản như Tetris đến các game đồ họa 3D phức tạp.

Tuy nhiên, một số nhược điểm của Java là hiệu suất và tốc độ xử lý có thể không bằng các ngôn ngữ khác như C++ và C#. Điều này có thể làm cho việc phát triển các game đòi hỏi sự tinh tế cao hơn gặp khó khăn.

Tóm lại, Java là một ngôn ngữ lập trình tuyệt vời để phát triển game, đặc biệt là cho các trò chơi di động và game trực tuyến. Tuy nhiên, việc lựa chọn ngôn ngữ phù hợp cũng phụ thuộc vào loại game bạn muốn phát triển và các yêu cầu kỹ thuật cụ thể của dự án.

## 2, Tài liệu tham khảo:

Tài liệu chính thức của Oracle về Java Game Development: https://docs.oracle.com/javase/8/docs/technotes/guides/games/

Sách "Beginning Java Game Development with LibGDX" của Lee Stemkoski, tập trung vào sử dụng thư viện LibGDX để phát triển game Java: https://www.amazon.com/Beginning-Java-Development-LibGDX-Stemkoski/dp/148421501X

Các khóa học trực tuyến về Java Game Development trên Udemy: https://www.udemy.com/topic/java-game-development/

Các hướng dẫn và ví dụ trên trang chủ của LibGDX: https://libgdx.com/dev/

Các dự án mẫu và tài liệu trên GitHub: https://github.com/libgdx/libgdx/wiki/Community-Projects

Ngoài ra, bạn cũng có thể tham gia các cộng đồng Java Game Development như Java Gaming, Java-Gaming.org, hay JavaGameDev subreddit để được hỗ trợ và trao đổi kinh nghiệm với các nhà phát triển game khác.