|  |  |  |
| --- | --- | --- |
| TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI | | |
| **VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG** | | |
|  | | |
| **BÁO CÁO ĐỒ ÁN II** | | |
| ***ĐỀ TÀI:*** | | |
| **CẤU TRÚC DỮ LIỆU NGĂN XẾP VÀ ỨNG DỤNG** | | |
|  | | |
| ***GVHD*: PGS.TS. Cao Tuấn Dũng** | | |
| ***DANH SÁCH SINH VIÊN:*** | | |
| **Nguyễn Trung Kiên** | **CNTT-10** | **20173209** |
| **Nguyễn Đăng Trung Kiên** | **CNTT-10** | **20173203** |
| **Hà Nội 12-2019** | | |

**NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN**

1. **Mục đích và nội dung của đồ án:**
2. **Kết quả đạt được:**
3. **Ý thức làm việc của sinh viên:**

Hà Nội, ngày tháng năm

Giảng viên hướng dẫn

*(Ký và ghi rõ họ tên)*

Lời giới thiệu

Trong thời đại công nghệ số, lượng dữ liệu và thông tin được sinh ra và trao đổi ngày càng tăng cao với tốc độ nhanh. Chính vì vậy việc lưu trữ thông tin một cách khoa học đã và đang là một trong những khía cạnh cực kỳ quan trọng trong công nghệ thông tin nói riêng và tất cả các lĩnh vực khác nói chung.

Hiện này, trong khoa học máy tính có rất nhiều cách tổ chức dữ liệu như: stack, queue, tree… và được ứng dụng giải quyết các bài toán khác nhau về mặt lý thuyết và cả trong thực tế. Tuy nhiên, các cấu trúc dữ liệu và ứng dụng của chúng trong thực tế lại khá khó để có thể nhìn ra nên đôi khi trở nên khó tiếp cận. Chính vì vậy, việc mô phỏng và trực quan hóa các cấu trúc dữ liệu cũng là một việc nên làm, để giúp sinh viên hiểu hơn về cấu trúc dữ liệu cũng như cách ứng dụng chúng trong thực tế.

Môn Project 1 có mục đích chính là để sinh viên được làm việc thực tế để tạo ra sản phẩm, giúp sinh viên không chỉ được học lý thuyết mà còn được áp dụng nó vào thực tế. Việc này giúp ích rất nhiều trong việc hiểu các kiến thức đã học cũng như định hướng cho bản thân sau này.

Trong phạm vi của một project, nhóm em xin thực hiện đề tài “Mô phỏng cấu trúc dữ liệu Ngăn xếp và Ứng dụng”. Đề tài này giúp sinh viên hiểu hơn về cấu trúc dữ liệu và cụ thể là cấu trúc dữ liệu ngăn xếp, bên cạnh đó cũng đưa ra một ứng dụng của ngăn xếp và triển khai thành một chương trình đáp ứng thực tế.

Mục lục

[I. Mô tả đề tài 6](#_Toc28554089)

[A. Chương trình mô phỏng Ngăn xếp (Demo Stack) 6](#_Toc28554090)

[1. Các chức năng chính: 6](#_Toc28554091)

[2. Chức năng bổ sung: 6](#_Toc28554092)

[3. Yêu cầu: 6](#_Toc28554093)

[B. Chương trình mô phỏng bãi đỗ xe (Car Park) 6](#_Toc28554094)

[1. Các chức năng chính: 6](#_Toc28554095)

[2. Yêu cầu: 7](#_Toc28554096)

[II. Kiến thức nền tảng 7](#_Toc28554097)

[A. Cấu trúc dữ liệu, thuật toán: 7](#_Toc28554098)

[1. Stack 7](#_Toc28554099)

[2. Thiết kế stack: 7](#_Toc28554100)

[3. Thuật toán mô phỏng trực quan: 8](#_Toc28554101)

[B. Ngôn ngữ lập trình, các kỹ thuật lập trình liên quan: 9](#_Toc28554102)

[1. Giới thiệu về ngôn ngữ Java: 9](#_Toc28554103)

[2. Giới thiệu về JavaFX: 9](#_Toc28554104)

[III. Giải quyết vấn đề 14](#_Toc28554105)

[A. Chương trình mô phỏng ngăn xếp đơn giản: 14](#_Toc28554106)

[1. Ý tưởng thực hiện: 14](#_Toc28554107)

[2. Tiến hành thực hiện: 15](#_Toc28554108)

[B. Chương trình mô phỏng bãi đỗ xe 23](#_Toc28554109)

[1. Ý tưởng thực hiện: 23](#_Toc28554110)

[2. Tiến hành thực hiện: 24](#_Toc28554111)

[IV. Kết quả - Đánh giá 33](#_Toc28554112)

[A. Chương trình mô phỏng ngăn xếp đơn giản 33](#_Toc28554113)

[1. Kết quả: 33](#_Toc28554114)

[2. Đánh giá: 34](#_Toc28554115)

[B. Chương trình mô phỏng bãi đỗ xe 34](#_Toc28554116)

[1. Kết quả: 34](#_Toc28554117)

[2. Đánh giá: 36](#_Toc28554118)

[V. Mô tả kế hoạch thực hiện - Đóng góp của các thành viên 36](#_Toc28554119)

[A. Kế hoạch thực hiện: 36](#_Toc28554120)

[B. Đóng góp của các thành viên: 37](#_Toc28554121)

[1. Nguyễn Trung Kiên - 20173209: 37](#_Toc28554122)

[2. Nguyễn Đăng Trung Kiên – 20173203: 37](#_Toc28554123)

# Mô tả đề tài

## Chương trình mô phỏng Ngăn xếp (Demo Stack):

### Các chức năng chính:

* + - * Các chức năng Push và Pop cho phép thêm phần tử vào ngăn xếp cũng như lấy phần tử của ngăn xếp ra.
      * Các chức năng thông báo trạng thái ngăn xếp như Top, isEmpty, isTrue, Size.
      * Hai lựa chọn chạy thủ công hoặc chạy tự động chương trình.

### Chức năng bổ sung:

Để chương trình thêm tính thân thiện với người dùng, yêu cầu có các chức năng bổ sung:

* + - * Thay đổi tốc độ của chương trình.
      * Điều chỉnh kích thước giao điện
      * Điều chỉnh màu sắc, kích thước của các đối tượng trong giao diện.
      * Thay đổi vị trí và kích thước của ngăn xếp

### Yêu cầu:

* + - * Các chức năng phải hoạt động đúng theo nguyên lý của ngăn xếp.
      * Các chức năng đều được mô phỏng trực quan qua đồ họa và chuyển động thực tế để có thể dễ dàng theo dõi.
      * Chương trình phải có tính thân thiện với người dùng, dễ sử dụng và đạt được hiệu quả sử dụng.

## Chương trình mô phỏng bãi đỗ xe (Car Park):

### Các chức năng chính:

* Thêm xe vào và lấy xe ra khỏi bãi đỗ xe.
* Thêm làn, bớt làn xe.
* Chức năng tự động để người dùng có thể tiện trong quá trình kiểm tra sản phẩm cũng như tiện trong việc quảng bá sản phẩm.
* Hiển thị trạng thái hệ thống trên màn hình để người dùng tiện theo dõi hoạt động trong bãi đỗ xe.
* Thay đổi tốc độ chương trình để phù hợp với nhu cầu thực tế.

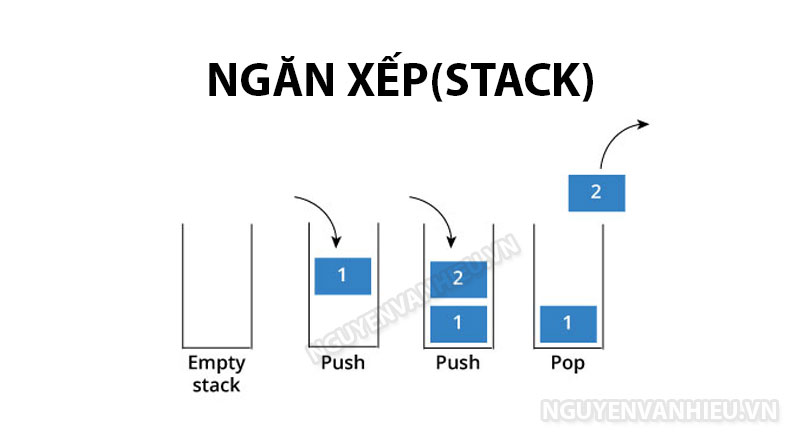
### Yêu cầu:

* Các chức năng hoạt động dựa trên nguyên lý hoạt động của ngăn xếp
* Chương trình được thực hiện chuyển động đồ họa để người dùng dễ dàng theo dõi.
* Chương trình phải có tính thân thiện với người dùng, dễ sử dụng và đạt được hiệu quả sử dụng.

# Kiến thức nền tảng

## Cấu trúc dữ liệu, thuật toán:

### Stack:

Stack là một cấu trúc dữ liệu trừu tượng hoạt động theo nguyên lý “vào sau ra trước” (**L**ast **I**n **F**irst **O**ut (LIFO)). Tức là, phần tử cuối cùng được chèn vào ngăn xếp sẽ là phần tử đầu tiên được truy cập trong ngăn xếp. Việc thao tác với ngăn xếp của chúng ta bao gồm các hành động sau:

* **Push**: Thêm một phần tử vào đỉnh của ngăn xếp, số phần tử của ngăn xếp tăng lên 1.
* **Pop**: Xóa bỏ phần tử đầu tiên ở đỉnh của ngăn xếp, số phần tử của ngăn xếp giảm đi 1.
* **Top**: Lấy giá trị của phần tử đầu tiên ở đỉnh của ngăn xếp, số phần tử của ngăn xếp không thay đổi.
* **IsEmpty**: Kiểm tra ngăn xếp trống hay không. Ngăn xếp trống là ngăn xếp không có phần tử nào.
* **IsFull**: Kiểm tra ngăn xếp đã đầy hay chưa. Thao tác này không phải lúc nào cũng có.
* **Size**: Lấy số lượng phần tử stack đang có.

### Thiết kế stack:

Với chương trình này, nhóm em không sử dụng thư viện hỗ trợ stack sẵn của java mà tự thiết kế 1 lớp mô phỏng stack và các chức năng đặc trưng của stack. Stack được lưu trữ dưới dạng mảng 1 chiều, các hành động của stack được thiết kế xoay quanh mảng này.

Mảng bắt đầu từ phần từ đầu tiên chỉ số là 0 và đây cũng là đáy stack. Ta sử dụng biến top để đánh dấu đỉnh stack, là chỉ số phần từ cuối cùng nhất trong mảng, khi stack rỗng top =-1.

Các hành động stack được thiết kế dưới dạng các phương thức như sau

* **Push**: thêm phần tử vào stack hay chính là thêm phần tử vào cuối của mảng. Tham số vào là một phần tử cùng loại của stack. Phương thức thực hiện bằng cách tăng giá trị top thêm 1, gán input vào phần tử top của mảng.
* **Pop**: đẩy phần tử cuối của stack ra. Trả về phần tử có chỉ số top và giảm top 1 đơn vị.
* **IsEmpty**: nếu top = -1 tức là stack trống.
* **IsFull**: kiểm tra giá trị top và số lượng phần tử của stack, top = kích thước stack -1 tức là stack đầy, phương thức trả về giá trị true.
* **FindStack**: phương thức này nhóm em tạo thêm cho stack, mục đích để tìm vị trí 1 phần tử và trả về vị trí của phần tử đó. Phương thức này phục vụ cho quá trình tìm kiếm phần tử để thực hiện push pop khi thực hiện trực quan. Phương thức thiết kế bằng cách dùng vòng lặp for duyệt hết mảng để tìm phần tử tham số vào.

### Thuật toán mô phỏng trực quan:

* Cơ sở để trực quan là chạy các thuật toán tương ứng. Các thuật toán chạy bên dưới sẽ thể hiện lên trực quan. Cụ thể trong chương trình, khi push pop, code chạy dưới là push pop thì sẽ thêm phương thức tạo trực quan tương ứng.
* Chương trình thiết kế gốm nhiều stack chính và 1 stack phụ. Các thuật toán để thực hiện trong chương trình như sau
* Tạo 1 phần tử và đẩy vào stack:
* Thuật toán:Tạo yêu cầu push phần tử. Kiểm tra stack chính còn chỗ không, nếu còn chỗ thực hiện push phần tử vào, nếu hết chỗ hiển thị thông báo hết chỗ, không push được.
* Lấy 1 phần từ trong stack ra:
* Thuật toán: tìm kiếm vị trí phần tử trong stack. Thực hiện push và pop các phần tử trước nó sang stack phụ. Lấy phần tử đó ra. Push và pop các phần tử từ stack 2 về lại stack 1.

## Ngôn ngữ lập trình, các kỹ thuật lập trình liên quan:

Trong Project này chúng em sử dụng ngôn ngữ lập trình Java ,thư viện Javafx và sử dụng fxml để thiết kế giao diện chương trình.

### Giới thiệu về ngôn ngữ Java:

* Java là một trong những ngôn ngữ lập trình hướng đối tượng. Nó được sử dụng trong phát triển phần mềm, trang web, game hay ứng dụng trên các thiết bị di động.
* Java được khởi đầu bởi James Gosling và bạn đồng nghiệp ở Sun MicroSystem năm 1991. Ban đầu Java được tạo ra nhằm mục đích viết phần mềm cho các sản phẩm gia dụng, và có tên là Oak.
* Java được phát hành năm 1994, đến năm 2010 được Oracle mua lại từ Sun MicroSystem.
* Java được tạo ra với tiêu chí “Viết (code) một lần, thực thi khắp nơi” (Write Once, Run Anywhere  – WORA). Chương trình phần mềm viết bằng Java có thể chạy trên mọi nền tảng (platform) khác nhau thông qua một môi trường thực thi với điều kiện có môi trường thực thi thích hợp hỗ trợ nền tảng đó .

### Giới thiệu về JavaFX:

JavaFX là một giải pháp công nghệ cho GUI trên nền tảng Java nhằm tạo giao diện đồ họa người dùng dựa trên Swing và Java2D nhưng phong phú, mới mẻ hơn rất nhiều. FXML là một ngôn ngữ dựa trên XML cung cấp cấu trúc cho việc xây dựng một giao diện người dùng riêng biệt, nó cũng tương tự dạng xây dựng giao diện HTML.

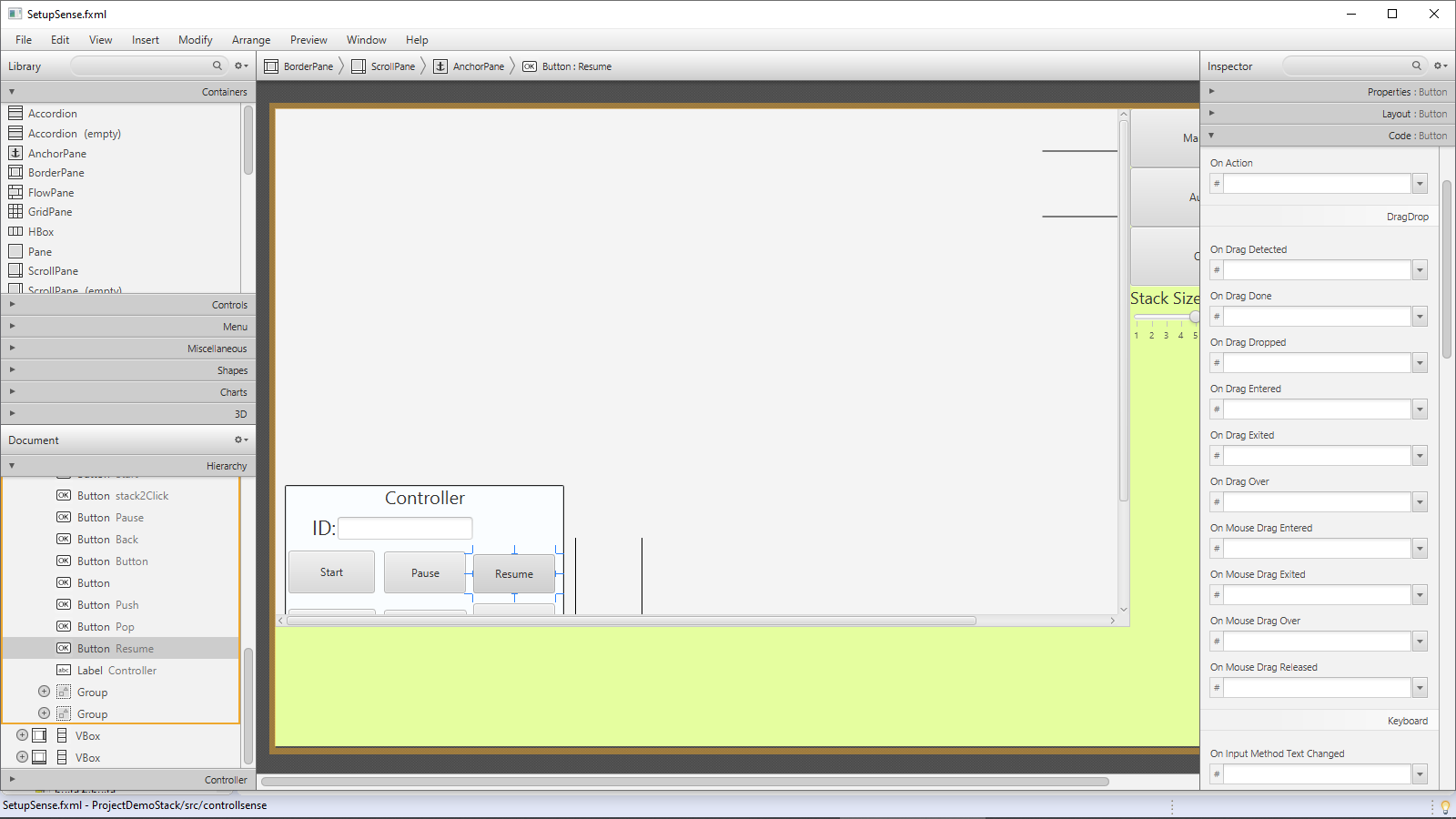
Bên cạnh đó, để viết giao diện FXML dễ dàng hơn nên cài đặt thêm JavaFX Scene Builder. JavaFX Scene Builder là một công cụ thiết kế trực quan, cho phép tạo ra giao diện ứng dụng một cách nhanh chóng bằng cách kéo thả. Và code được tự động tạo ra tương ứng dưới dạng XML.

Đề tài do nhóm thực hiện mục đích là mô phỏng lại ứng dụng stack, trực quan quá trình push pop stack, mô phỏng gửi lấy xe từ bãi đỗ xe. Đề tài yêu cầu chính là làm về trực quan trên giao diện, các hiệu ứng hoạt họa di chuyển. JavaFX, FXML có thể đáp ứng những yêu cầu trên, thêm vào đó, công nghệ còn cho phép kết hợp CSS, chèn hình ảnh dễ dàng. Vì vậy nhóm em chọn công cụ này để thực hiện đề tài.

Trong các chương trình của mình, nhóm em sử dụng nhiều công cụ của javafx và để hiểu được chương trình, nhóm xin trình bày các công cụ quan trọng dưới đây:

* Gán sự kiện cho button, event button click

Button hiểu đơn giản là một nút bấm trong chương trình. Tuy nhiên trong chương trình của nhóm, để tiện nhất thì nhóm cũng đồng thời sử dụng button như các object (các xe, các phần tử của stack).

Với button được tạo bằng phương pháp kéo thả trực tiếp với Scene Builder, công cụ này hỗ trợ bắt sự kiện cho button, sự kiện là những phương thức được tạo trong class controller. Ngoài Action khi click, công cụ hỗ trợ nhiều cách thức kích hoạt khác như trong hình

Với button được tạo bằng java code, thiết lập ActionEvent cho button thực hiện qua phương thức setOnAction, cú pháp cụ thể như sau:

button.setOnAction(**new** EventHandler<ActionEvent>() {

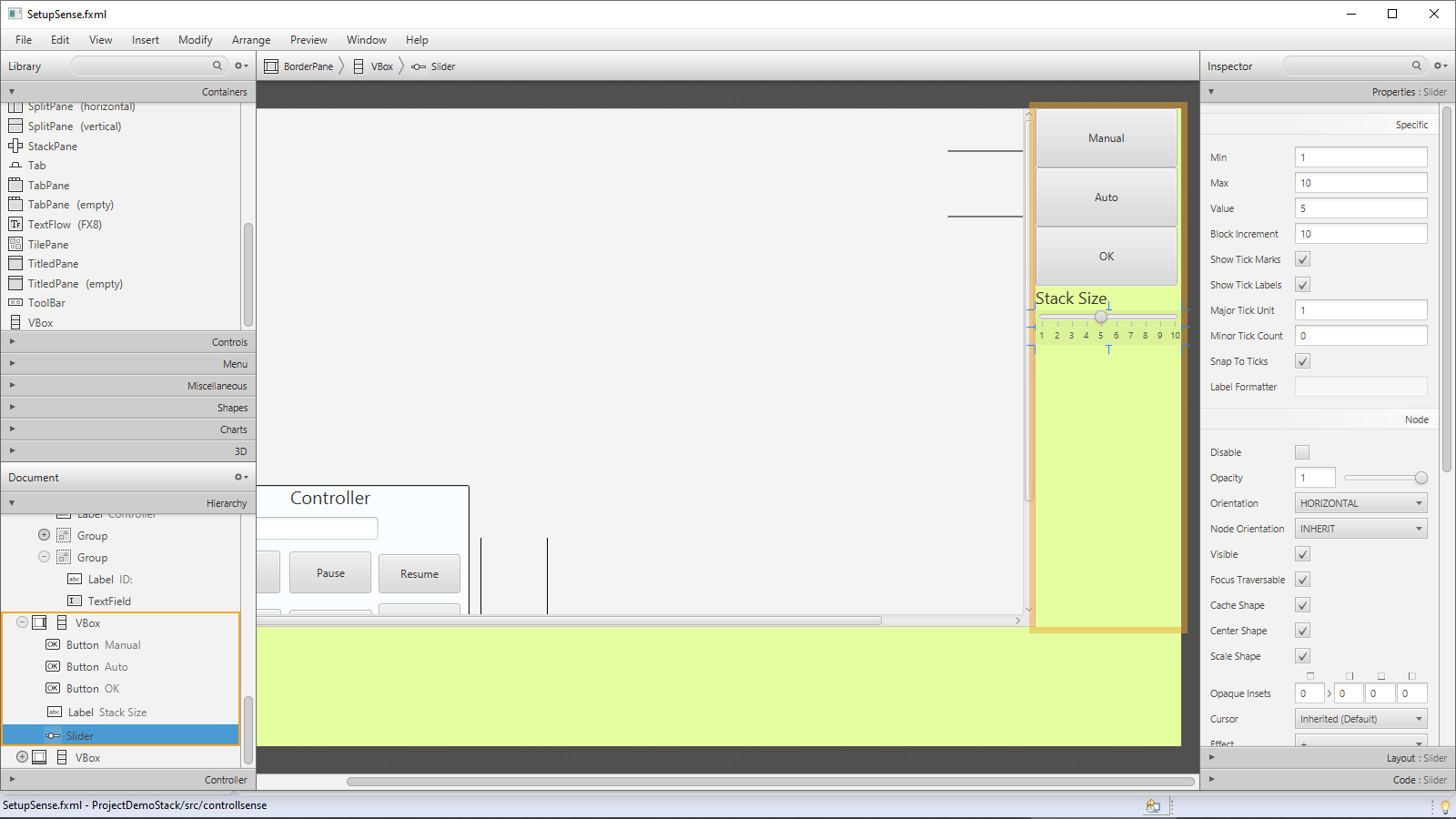
@Override

**public** **void** handle(ActionEvent event) {

}

});

* Slider (thanh trượt):

Trong JavaFX thành phần này có thể tạo trực tiếp bằng Scene Builder vào file fxml rồi gắn ID hoặc tạo bằng java code qua thư viện javafx.scene.control.Slider; của javafx.

Dùng công cụ có sẵn của Scene Builder, ta có thể thêm 1 slider tùy ý, chỉnh chỉ số min max, khoảng nhảy. Và đặt ID để sử dụng trong java code

Trong class controller khai báo:

@FXML

**private** Slider stackSizeSlider;

với stackSizeSlider là ID của slider. Chúng ta có thể sử dụng tên biến này để viết java code. Dữ liệu trên slider được lấy qua phương thức gán một Listener để theo dõi sự thay đổi của slider, và giá trị sau khi thay đổi sẽ được gán vào giá trị cần. Cụ thể như sau:

slider.valueProperty().addListener(**new** ChangeListener<Number>() {

**public** **void** changed(ObservableValue<? **extends** Number> ov, Number old\_val, Number new\_val) {

}}});

Trong phương thức trên, old\_val là số liệu cũ của slider, new\_val là số liệu mới, số mà con trỏ trên slider đang chỉ. Ta có thể tùy ý sử dụng các giá trị này để thực hiện yêu cầu đề ra.

* ColorPicker (công cụ chọn màu sắc):

Công cụ này này được javafx hỗ trợ và nằm trong package javafx.scene.control. Màu được lấy về để sử dụng qua phương thức getValue của ColorPicker. Để khi chọn màu thì có hiển thị tương ứng sử dụng thêm phương thức setOnAction với cú pháp như sau:

colorPicker.setOnAction(**new** EventHandler<ActionEvent>() {

@Override

**public** **void** handle(ActionEvent event) {

}

});

Bên trong phương thức handle có gọi đến phương thức getValue để chỉnh màu sắc của các phần tử.

* ComboBox

Combobox trong javafx giống 1 arraylist. Ở đây sử dụng combobox dạng string ghi các thông số độ dày thêm kí tự x ở phía trước. Cú pháp khởi tạo giống với cú pháp khởi tạo arraylist: ComboBox<String> comboBox = **new** ComboBox<String>();

Thêm các thành phần cho combobox bằng lệnh getItem().addAll(…); Trong chương trình các thành phần combobox là "x1.0","x2.0","x4.0","x6.0"

Thông số người dùng chọn lấy về bằng phương thức valueProperty() của combobox

comboBox.valueProperty().addListener(**new** ChangeListener<String>() {

@Override

**public** **void** changed(ObservableValue<? **extends** String> observable, String old\_val, String new\_val) {

}

});

new\_val là giá trị mới của combobox, là giá trị người dùng chọn. Từ đó ta có thể thao tác với các giá trị này theo nhu cầu.

* Transition (các chuyển động trong javafx):

Trong chương trình, các hiệu ứng di chuyển button thực hiện trên cơ sở dịch chuyển tịnh tiến từ vị trí này (có layoutX, layoutY) đến vị trí khác. Tuy nhiên, không phải là tịnh tiến bình thường, biến mất ở đây và xuất hiện ở kia, mà hiệu ứng này thực hiện di chuyển từ từ trên đường thẳng nối điểm giữa 2 vị trí, là một animation với thời gian được thiết lập. JavaFX cung cấp package javafx.animation cho phép tạo các hiệu ứng animation.

* Để thực hiện hiệu ứng di chuyển, nhóm sử dụng class TranslateTransiton trong package. Class giúp tạo hiệu ứng di chuyển chuyển động từ vị trí thứ nhất đến vị trí thứ 2. Tạo một đối tượng của class này, thiết lập các thông số: đối tượng cần di chuyển, đường đi (hiệu tọa độ điểm đến và điểm đi), thời gian di chuyển … Hiệu ứng sau khi được thiết lập được thêm vào SequentialTransition (tương tự như một cơ chế lập lịch) để sau này thực hiện. Ví dụ cài đặt di chuyển cho 1 button từ vị trí 1 đến vị trí 2:

**public** **void** moveButton(Button a, XY pos1, XY pos2, SequentialTransition b) {

TranslateTransition transition = **new** TranslateTransition();

transition.setDuration(Duration.*seconds*(0.5));

transition.setNode(a);

transition.setToX(pos2.getX()-pos1.getX());

transition.setToY(pos2.getY()-pos1.getY());

transition.setAutoReverse(**false**);

b.getChildren().addAll( **new** PauseTransition(Duration.*seconds*(0.25)),transition);

}

Với pos1 là tọa độ điểm đi (x,y), pos2 là tọa độ điểm đến, b là đối tượng của class SequentialTransition

* Để làm biến mất hay xuất hiện 1 node, đều dùng class FadeTransition của javafx. Khác thông số mà sẽ là xuất hiện hay biến mất.

ft.setFromValue(1.0);

ft.setToValue(0.0);

Ở đây là hiệu ứng xuất hiện, nếu đổi thông số cho nhau thì sẽ thành biến mất. Sau khi thiết lập các thông số cho đối tượng giúp biến mất xuất hiện. Đối tượng này được thêm vào SequentialTransition.

* Tương tự như vậy, để thực hiện thao tác xoay vật thể, ta sử dụng Rotate Transition và điều chỉnh số góc muốn xoay cho vật thể.
* Chuyển scene và truyền dữ liệu qua các scene:
* Chuyển scene:

Mục đích giống class main, cần stage và scene từ file FXML tương ứng.

Trước hết, tạo 1 stage mới hoặc lấy stage hiện tại bằng cách: từ 1 đối tượng trọng stage hiện tại thực hiện phương thức getScene().getWindow, ép kiểu về Stage.

Để có được scene từ file FXML:

* Tạo đối tượng giúp đọc file

FXMLLoader loader = **new** FXMLLoader();

* Thiết lập vị trí đọc qua phương thức setLocation với đầu vào là URL đến file.
* URL được lấy bằng phương thức getClass().getResourse(“…”) tham số là đường dẫn đến file FXML.
* Tạo scene từ việc load loader.
* Thêm scene vào stage.
* Truyền dữ liệu qua các scene:

Sau khi load được file FXML theo cách trên, ta có thể truy nhập vào class điều khiển FXML bằng cách sử dụng phương thức getController(); Từ đó có thể truyền dữ liệu.

Trong chương trình do nhóm thực hiện, có truyền các thuộc tính của các thành phần từ setupscene sang 2 chế độ demo. Thực hiện bằng cách trong 2 controller của 2 chế độ demo có phương thức lấy thuộc tính được truyền đến. Các thuộc tính được thiết lập trung gian qua đổi tượng truyen thuộc class ObjectInScene. Sau mỗi bước thiết lập, cập nhật lại đối tượng thuộc class này.

Khi button chọn chế độ demo được click, phương thức thiết lập trong 2 controller được gọi với tham số truyền vào là đối tượng truyen. Phương thức thiết lập lấy dữ liệu và tùy chỉnh cho scene của mình.

# Giải quyết vấn đề

## Chương trình mô phỏng ngăn xếp đơn giản:

Chương trình mục đích mô phỏng push pop qua lại giữa 2 stack, giúp nhóm nắm được cách thực hiện trực quan, tìm hiểu và thực hiện các câu lệnh trong lập trình giao diện. Trong phần này nhóm sẽ nói chi tiết về các phương thức quan trọng cũng như đi sâu vào các phương pháp sử dụng javafx để giải quyết bài toán, các chức năng cơ bản trong javafx như gán sự kiện cho button và nhiều tính năng khác. Các chức năng này cũng được sử dụng trong chương trình mô phỏng bãi đỗ xe. Vì vậy nhóm sẽ trình bày chi tiết để từ đó áp dụng vào phần chương trình mô phỏng bãi đỗ xe giúp người đọc dễ hiểu vấn đề hơn. Chương trình có cấu trúc tổng thể như sau:

Khi chạy chương trình, sẽ vào giao diện cài đặt, thiết lập các thông số.

* Tùy chỉnh kích thước stack, số phần tử stack có thể chứa (1->10)
* Chỉnh màu sắc các thành phần trong giao diện
* Thay đổi kích thước khung hình (thay đổi kích thước pane)
* Thay đổi kích thước stack, chiều dài, chiều rộng, thay đổi độ dày stack
* Sau khi thiết lập chọn chức năng manual hoặc auto để chuyển hướng sang kiểu mô phỏng tương ứng

2 kiểu mô phỏng đều có chức năng chung:

* Thể hiện quá trình push pop
* Chỉnh tốc độ quá trình trực quan
* Có nút quay lại giao diện cài đặt

Mô phỏng kiểu manual có thêm các chức năng:

Nhập ID phần tử, nếu không nhập ID sinh ngẫu nhiên

* Chức năng push, pop từng bước
* Mô phỏng kiểu auto tự động có thêm chức năng tạm dừng pause và tiếp tục continue

### Ý tưởng thực hiện:

Đối với yêu cầu bài toán, nhóm sử dụng 2 mảng loại Button đại diện cho 2 stack 1 và stack 2, 2 mảng loại string để lưu id tương ứng. Các stack được cài đặt giống với các thao tác trên stack. Việc kiểm tra còn chỗ hay không thực hiện trên các stack này.

Khi thực hiện push, chương trình sẽ đồng thời push trên cả 2 loại stack và tương tự đối với pop. Khi pop có ID, ID được tìm trong mảng string. Từ đó lấy ra thứ tự phần tử trong stack và thực hiện các thao tác push pop phù hợp. Các quá trình đều có hiệu ứng giao diện.

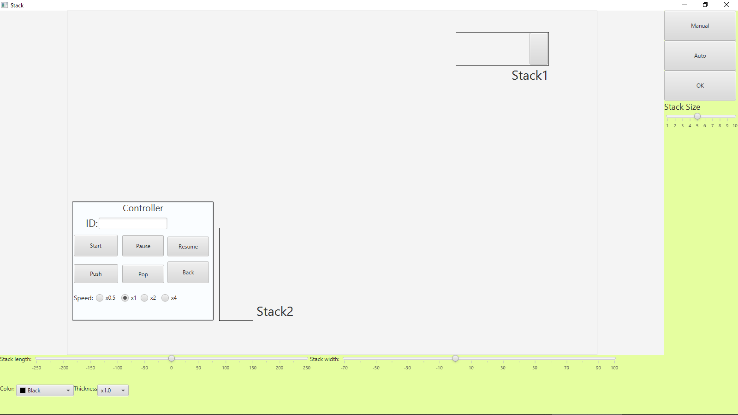
Hiệu ứng di chuyển trên anchorpane trong chương trình được thực hiện bằng phép tịnh tiến button (node)

Chương trình sử dụng thêm 1 biến để lưu lại tọa độ đỉnh hiện tại của stack trên pane, phục vụ cho quá trình trực quan push, pop trên giao diện. Biến này có giá trị ban đầu là tọa độ đáy stack lùi lại 1 button (stack 1 lùi lại tọa độ x, stack 2 lùi lại tọa độ y). Khi push, pop tọa độ biến này thay đổi tương ứng. Biến gồm 2 tọa độ X, Y tương ứng với layout X và layout Y của node.

Ngoài ra, nhóm còn sử dụng điểm trung gian M cố định, làm điểm trung gian di chuyển giữa 2 stack.

### Tiến hành thực hiện:

#### Thiết kế giao diện cài đặt, giao diện khi khởi động chương trình

Để đáp ứng các yêu cầu đề ra, giao diện của chương trình được thiết kế một cách phù hợp nhất như sau:

Nhóm dùng FXML giao diện kéo thả để thiết kế bố cục tổng thể, các thiết lập thêm sẽ dùng java code để tạo vì khi dùng FXML để thiết kế, các thành phần là cố định, không thể loại khỏi pane. Khi click vào thành phần nào thì sẽ hiện các cài đặt cho thành phần đó bằng cách tạo button ấn dưới. Việc tạo giao diện được nhóm thực hiện như sau:

* Trong javafx fxml và thiết kế giao diện nói chung, các thành phần component được chia thành 2 loại là container và children, trong đó container là loại component dùng để chứa các component khác, có thể là 1 container khác hoặc children: button, label…. Khi tạo giao diện, chúng ta cần tạo 1 container là pane (trong chương trình là kiểu border pane) để chứa đựng tất cả các thành phần khác.
* Để thực hiện chương trình, trước hết tạo file fxml, với pane là border pane. Border pane chia sẵn thành 5 vùng top bot left right middle. Vùng right đặt vbox pane chứa các tùy chọn cố định: button chọn chế độ demo, và thanh slider chọn kích thước stack. Vùng bot để hiển thị các cài đặt tùy biến cho từng thành phần. Còn middle là giao diện sẽ dùng để demo. Các tùy biến sẽ hiển thị ở middle. Middle chọn anchor pane, hiệu ứng di chuyển chỉ có thể thực hiện trên anchor pane
* Trên vùng middle, dùng công cụ line để vẽ 2 stack, label để đặt tên rồi neo vào 2 cạnh của pane. Tạo vùng controller người dùng, thêm các button, textfield để người dùng nhập id. Với mỗi thành phần sẽ tạo 1 button, làm mờ thành không màu không hiển thị đệm bên dưới thành phần đó. Button dùng để kích hoạt hiện các cài đặt cho thành phần đó. Cài đặt hiển thị ở phần bot của border pane.
* Vì ở đây mô phỏng push pop các phần tử, có thao tác click trực tiếp trên phần tử để pop nên các phần tử được định dạng là button. Sau này khi mô phỏng bãi xe, chỉ cần chỉnh style của các button này thành hình ảnh ô tô.

#### Các chức năng bổ sung của chương trình:

* Tạo các setting cho các phần của chương trình:

Mỗi thành phần đều tạo một button, chỉnh button không màu và gán sự kiện. Khi thành phần được click (button ẩn dưới được click) thì các chức năng cho phép cài đặt hiển thị ở bot của border pane.

Các chức năng cho phép tùy chỉnh ứng với mỗi thành phần:

* Pane: chỉnh kích thước pane (kích thước khung nhìn), chỉnh màu sắc
* Stack: chỉnh chiều dài chiều rộng stack, chỉnh màu, chỉnh độ dày line
* Label: chỉnh màu

Button được kích hoạt, sẽ xóa hết các thành phần đã có trong pane trước rồi mới thêm các tùy chỉnh hiển thị. Việc này đảm bảo các thành phần thiết lập không bị chồng chéo lên nhau. Xóa các thành phần trên vpane ở border pane bằng phương thức clear của pane. Thêm các thành phần bằng phương thức getchildren().add(…);. Các tùy chỉnh sẽ hiển thị kết quả ngay phía trên.

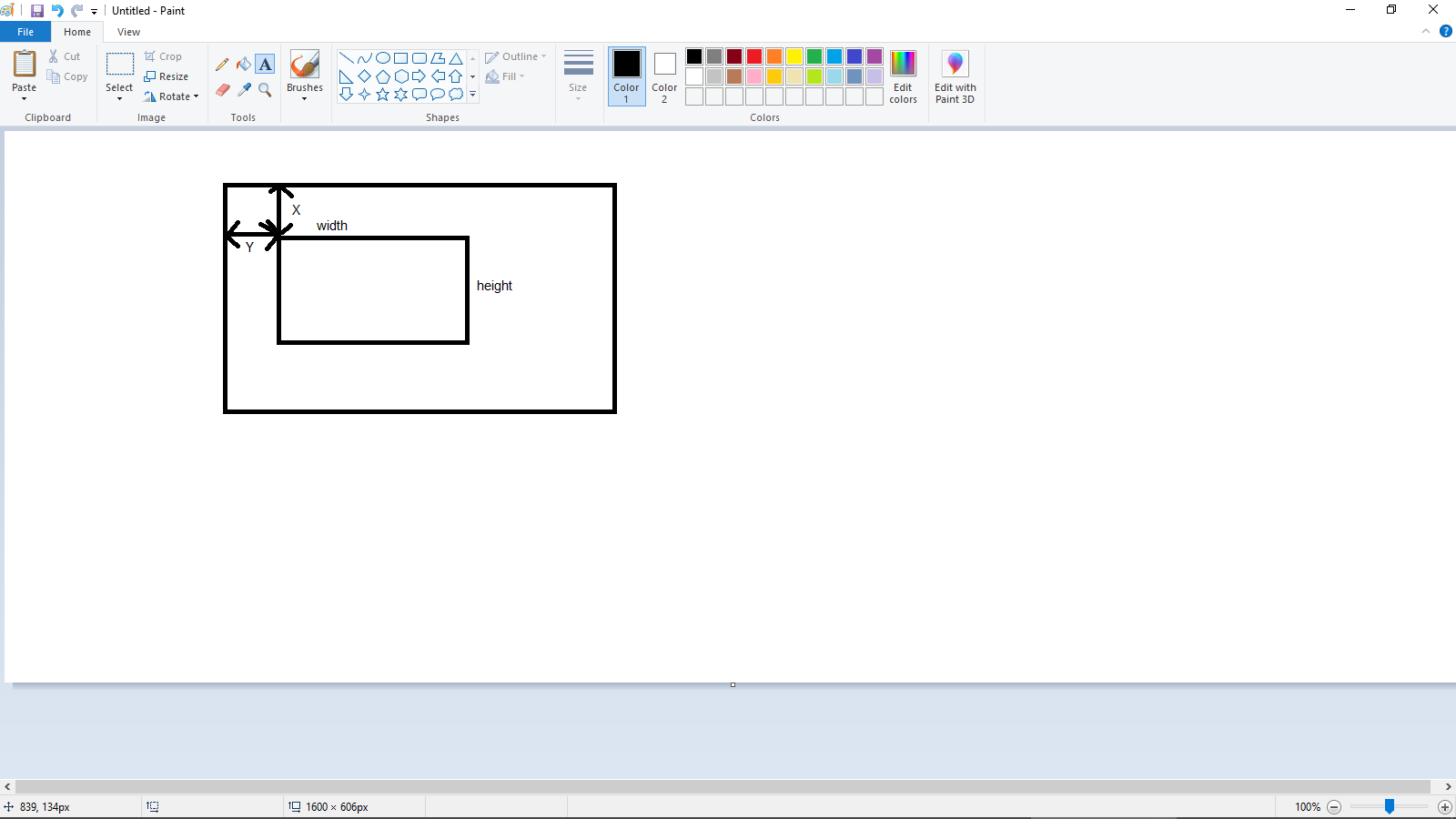
* Chức năng chỉnh kích thước ngăn xếp:

Với chức năng này, ta sử dụng slider (thanh trượt trả về các giá trị trong một khoảng nhất định) là phù hợp nhất.

Từ số liệu mới này được sử dụng để chỉnh kích thước button mô tả kích thước stack và được truyền sang scene khác để phục vụ việc mô phỏng.

Button mô tả đặt trong stack như trong hình. Khi kích thước stack thay đổi, kích thước button này cũng thay đổi theo.

Kích thước của button và layout so với pane trong javafx



Button mô tả có height bằng chiều dài line làm đáy stack, width bằng chiều dài line làm thành bên stack chia kích thước stack. Kích thước được thiết lập qua phương thức setprefsize(width, height); của button. Phần thiết lập này được thêm vào trong phương thức addListener(); của slider.

* Thiết kế chức năng thiết lập cho mỗi thành phần
* Chức năng chỉnh màu sắc

Chương trình sử dụng class ColorPicker của JavaFx để tạo hộp chọn màu.

Mỗi thành phần trong chương trình sẽ có cách đặt màu khác nhau.

Với stack được cấu thành bởi 3 line, chỉnh màu stack chính là chỉnh màu cho 3 line này. Line được cài đặt màu qua phương thức setStroke(Paint paint);

Với label, màu được đặt trực tiếp bằng phương thức setTextFill(Paint paint);

Còn đối với button và pane, việc chỉnh màu phức tạp hơn vì phải cài đặt qua câu lênh CSS dưới dạng String. Phương thức getValue của ColorPicker trả về mã màu dạng 0x+ mã màu. Vì vậy phải tách phần mã màu riêng và kết hợp với String khác tạo câu lệnh CSS chuẩn rồi từ đó dùng phương thức setStyle để đặt màu.

* Chức năng chỉnh chiều dài chiều rộng của stack, chỉnh độ dày line stack
* Chức năng chỉnh chiều dài chiều rộng của stack:

Tạo 2 slider bằng java code cho chiều dài và chiều rộng, cài đặt các thông số qua phương thức của slider. Đặt trong phương thức listener của slider lệnh thay đổi chiều dài line, thay đổi start x, y end x,y để chỉnh chiều dài line.

* Chỉnh độ dày line stack:

Nhóm sử dụng combobox thiết kế chức năng này.

Để chỉnh độ dày cho các line của stack, ta sử dụng giá trị new\_val và dùng phương thức setStrokeWidth(Double ….) để thay đổi độ dày.

* Chỉnh kích thước pane, kích thước khung nhìn:

Ở đây, tạo 2 TextField ứng với width và height của pane để người dùng có thể thiết lập. Nếu thông số sai chương trình báo lỗi.

* B1: Đọc dữ liệu String từ TextField bằng phương thức getText.
* B2: Chuyển dữ liệu sang double, nếu không chuyển được báo lỗi
* B3: Đặt kích thước của pane bằng phương thức setPrefSize(…);

Ngoài ra, chương tình có thêm chức năng lấy kích thước màn hình máy tính đang chạy chương trình để thiết lập cho pane bằng Dimension screenSize = Toolkit.*getDefaultToolkit*().getScreenSize();

Width và height ứng với khi gọi screenSize.getWidth(); screenSize.getHeight();

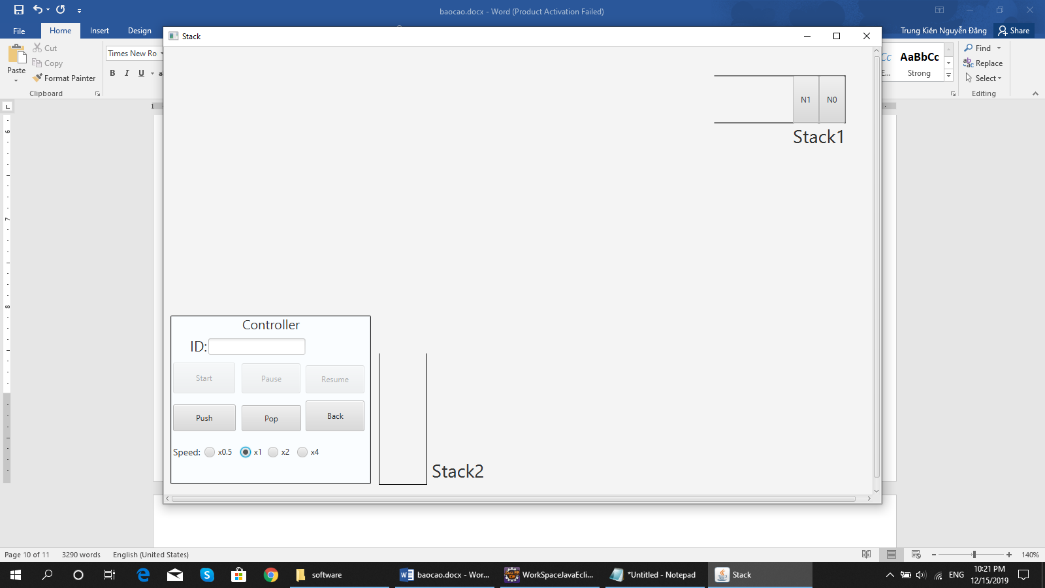
Trong chương trình này nhóm em muốn dùng cách này để thiết lập trạng thái màn hình, sang chương trình demo bãi xe sẽ dùng scale để thiết lập.

* Chức năng chỉnh speed.

Speed trong khi trực quan có thể được thay đổi bằng cách thay đổi tốc độ thực hiện của SequentialTransition. Sử dụng hàm rateProperty().bind(…); để thay đổi. Tham số vào cho bind là ObservableValue. Để đơn giản, trong chương trình nhóm em sử dụng 1 slider ngầm làm ObservableValue. ObservableValue chính là giá trị hiện tại của slider.

Chương trình sử dụng group radio button để giúp người dùng chọn speed. Khi người dùng chọn, nhãn của button được trả về dưới dạng String. Ta tách String lấy dự liệu dạng số và đặt giá trị mặc định của slider ngầm là giá trị này. Khi đó speed được thay đổi theo đúng yêu cầu của người dùng

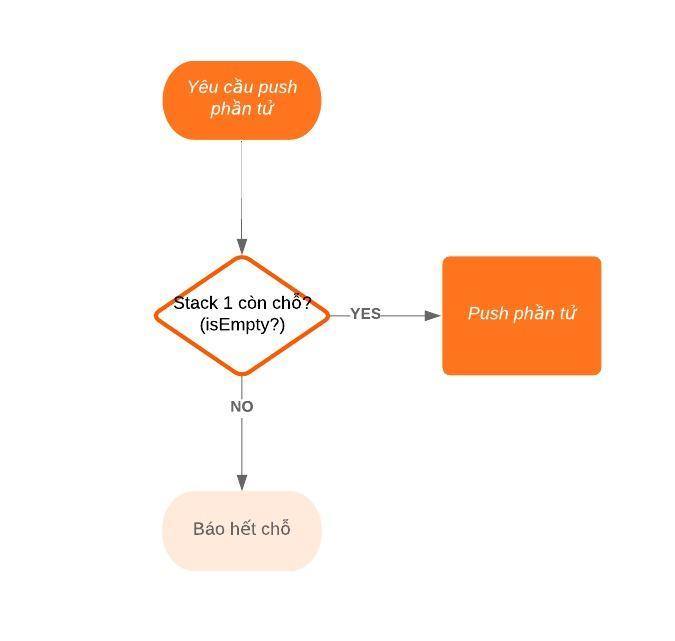
#### Các chức năng chính của chương trình:

Chế độ demo manual có cấu trúc chương trình: chức năng push, pop và nút back để quay lại giao diện. Cho phép click trực tiếp vào phần tử để pop. Có vùng TextField để người dùng nhập ID phần tử. Các vùng text hiển thị thông báo.

Trong chương trình này, các thao tác mô phỏng sẽ thực hiện trên stack lý thuyết trước. Ứng với mỗi quá trình push pop… trên stack, chương trình sẽ thực hiện mô phỏng tương ứng trên giao diện và hiện thông báo phù hợp.

* Chức năng push

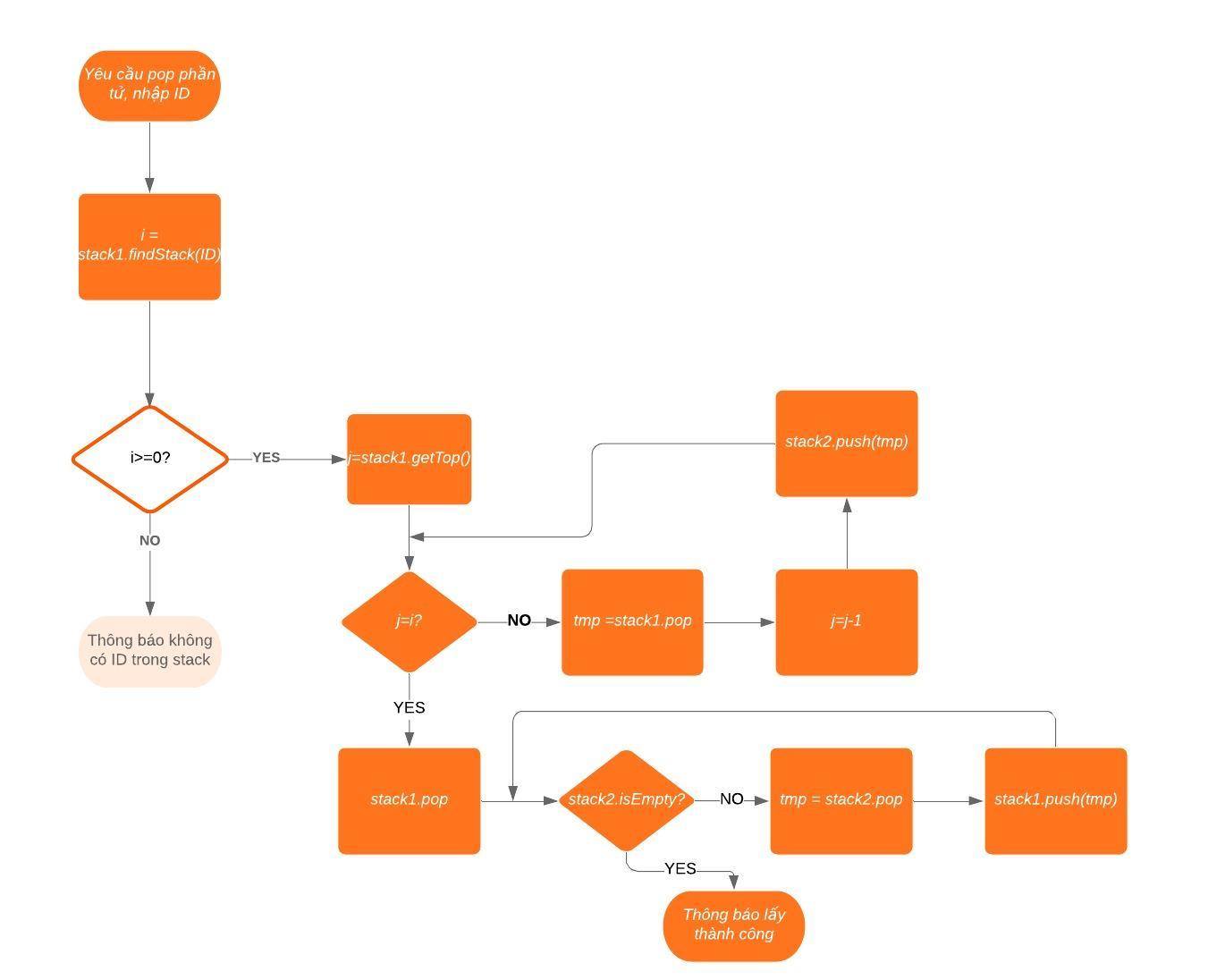
Nhóm thực hiện chức năng push với thuật toán như sau:



Ở chức năng này, người dùng có thể nhập ID cụ thể cho phần tử hoặc chương trình sẽ tự động điền ID theo mặc định. ID này sẽ được lưu trữ và sử dụng cho việc truy cập phần tử sau này.

* Chức năng pop với ID được nhập vào

Chức năng được thực hiện như sau:

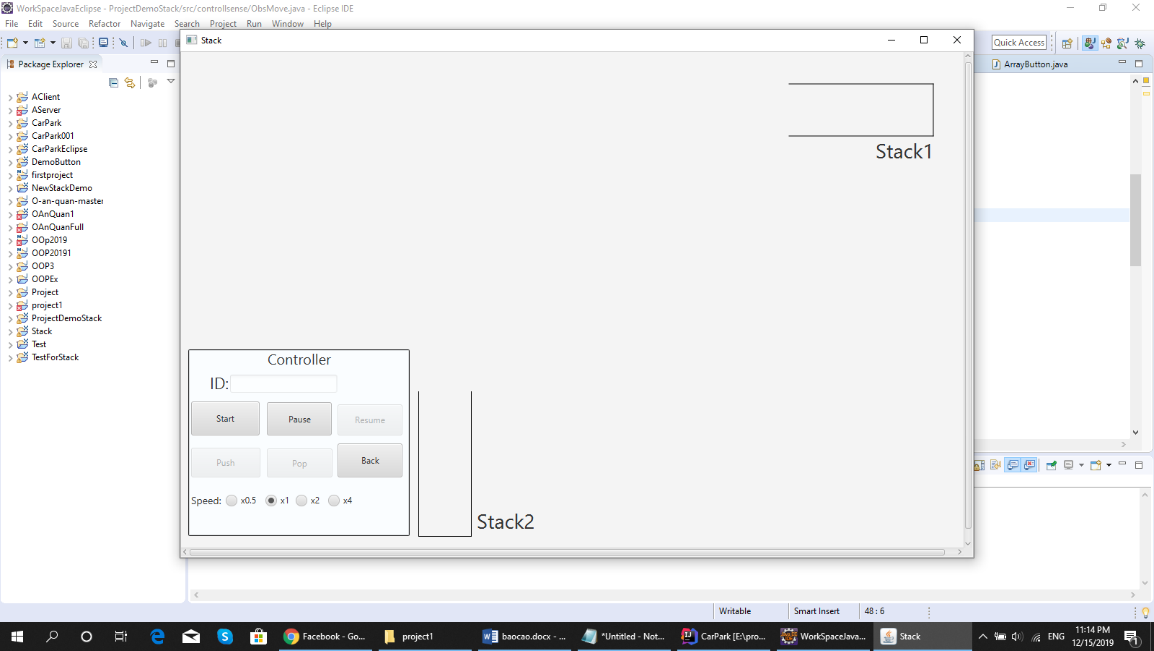


Trong đó findStack là một phương thức được nhóm thêm vào trong đối tượng stack được tạo. findStack sẽ tìm kiếm trong stack và trả về vị trí của phần tử được chọn trong stack. Từ đó chương trình thực hiện lấy phần tử ra.

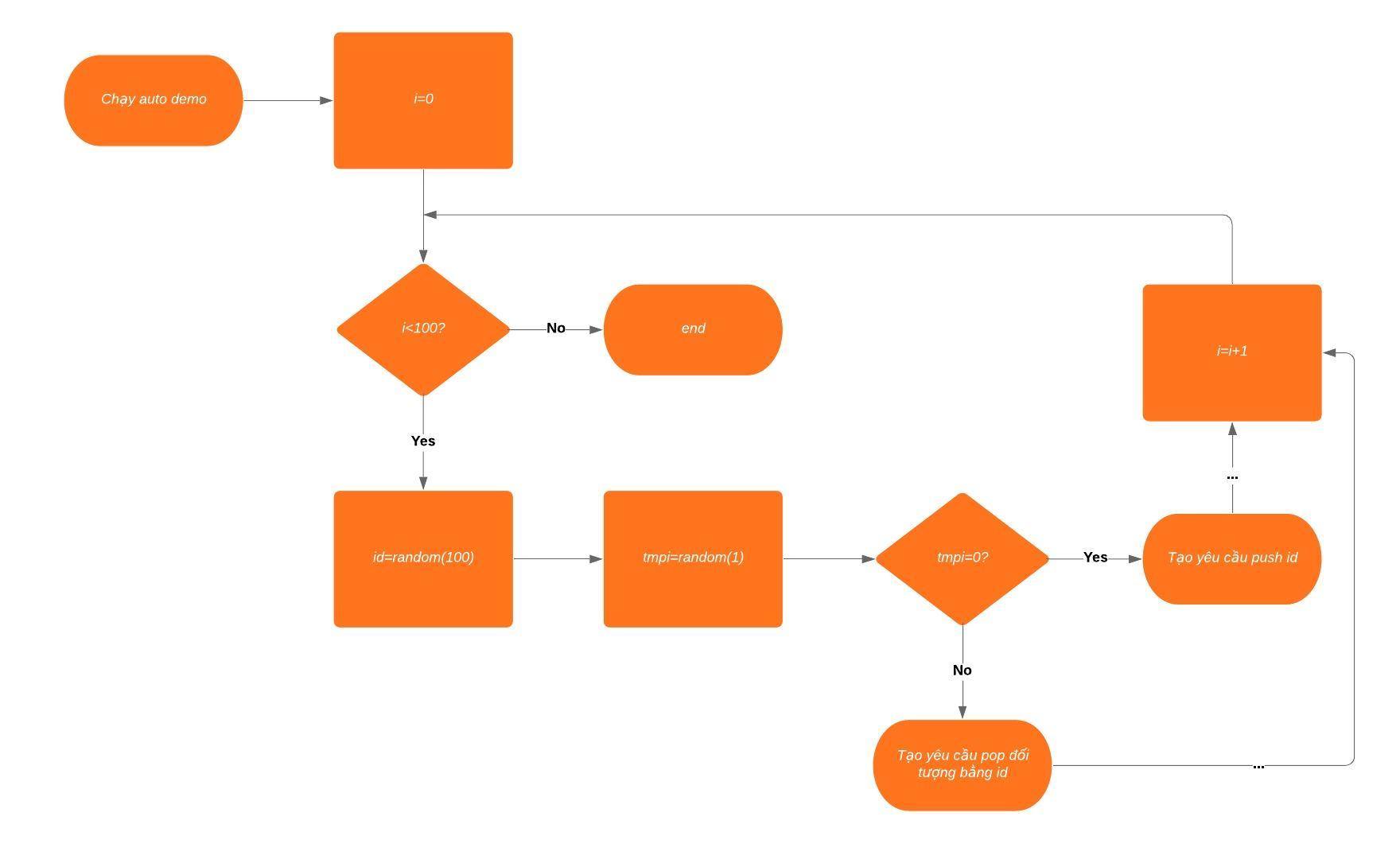
Quá trình chuyển động này được dựa trên cơ chế chuyển động tuần tự, vì vậy người dùng có thể theo dõi dễ dàng quá trình di chuyển cũng như hướng đi của các phần tử.

Sau khi thực hiện xong, chương trình thực hiện làm mới lịch để tránh bị lặp lại khi thực hiện các chức năng khác của chương trình.

* Chức năng auto

Về cơ bản cách thiết kế giống với chức năng manual ở push pop.

Để chạy auto, nhóm sử dụng vòng lặp while với số vòng lặp 30 để sinh dữ liệu mô phỏng. Quá trình push pop và id đều được chọn ngẫu nhiên với lệnh random.

Vòng lặp này được biểu diễn như sau:

Trong đó các thao tác push và pop được thực hiện như khi người dùng làm thủ công.

* Chức năng pause và resume:

Bắt sự kiện, pause gọi phương thức pause() của SequentialTransition, còn resume thì gọi tiếp tục phương thức play();

#### Tạo class Main thực hiện chuyển scene:

Trong chương trình class main đóng vai trò tạo giao diện, tao khung hình nơi đặt FXML. Main xác định scene nào được chạy đầu tiên. Định dạng khung hình, style giao diện

Khi chương trình chạy cần 1 cửa sổ Stage và nội dung bên trong (file FXML). Class main có nhiệm vụ tạo môi trường để chạy chương trình bằng cách kết nối 2 thành phần này.

Cú pháp chung của class main như sau:

**public** **class** MainAppClass **extends** Application{

@Override

**public** **void** start(Stage primaryStage) {

**try** {

Parent root = FXMLLoader.*load*(getClass()

.getResource("/controllsense/SetupSense.fxml"));

primaryStage.setTitle("Stack");

primaryStage.setScene(**new** Scene(root));

primaryStage.show();

} **catch**(Exception e) {

e.printStackTrace();

}

}

**public** **static** **void** main(String[] args) {

*launch*(args);

}

}

/controllsense/SetupSense.fxml là tên đường dẫn đến file FXML, nội dung của stage. Settille có thể tùy biến giá trị, là tên của stage.

## Chương trình mô phỏng bãi đỗ xe

### Ý tưởng thực hiện:

Dựa trên cấu trúc dữ liệu ngăn xếp, nhóm em hình thành ý tưởng mô phỏng bãi đỗ xe như sau:

Mỗi làn xe tương ứng với một ngăn xếp trong đó các xe là các phần tử của ngăn xếp. Thao tác thêm làn cũng chính là tạo thêm stack mới cho chương trình.

Các hoạt động trong bãi đỗ xe tương ứng với các hoạt động của ngăn xếp bao gồm Push (thêm xe vào bãi đỗ), Pop (lấy xe ra khỏi bãi đỗ).

Chức năng lấy xe yêu cầu chuyển các xe bên ngoài xuống một làn phụ, sau đó lấy xe cần lấy, rồi lại chuyển các xe từ làn phụ về làn cũ. Điều này là do trong cấu trúc dữ liệu ngăn xếp, chúng ta chỉ có thể truy cập vào phần tử cuối cùng được thêm vào trong ngăn xếp (xe ngoài cùng). Vì vậy để lấy một phần tử ở bên trong (xe cần lấy), ta cần lần lượt pop các phần tử ở ngoài, push chúng vào một stack phụ, rồi mới có thể truy cập các phần tử đó. Và để đem các xe trở lại làn xe ban đầu thì ta lại pop lần lượt các phần tử trong stack phụ ra rồi push chúng trở lại stack đầu.

Dựa trên các yếu tố trên, nhóm em quyết định chọn các Button để làm các xe để phù hợp với các yêu cầu đề ra.

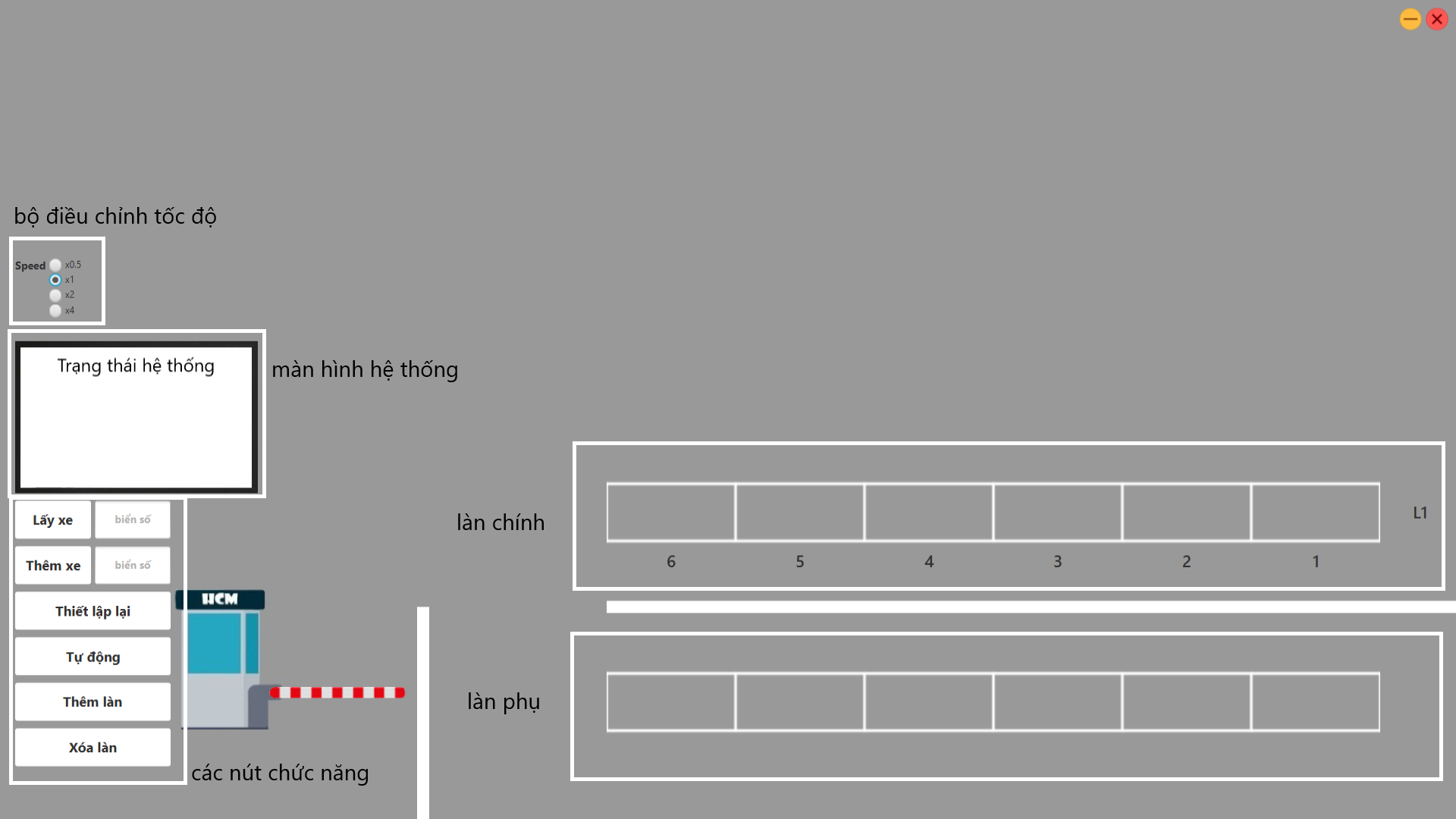
### Tiến hành thực hiện:

#### Tạo lập Stack

Trong Java, tạo một Class Stack đóng vai trò là Làn xe trong đó các phần tử là các Button (xe). Stack này có các tính chất và hoạt động tương tự như Stack thông thường với các phần tử là các nút (Button). Tuy nhiên, ta chỉ cho stack có tối đa 6 phần tử để phù hợp với chương trình. Sau đó ta tạo các stack cần thiết cho chương trình (được đề cập bên dưới). Riêng đối với 5 stack có tác dụng làm các làn xe chính, ta tạo một mảng các stack để chứa 5 stack này để tiện cho việc truy cập sau này.

#### Thiết kế giao diện chương trình

Sử dụng Javafx Scene Builder ta thiết kế giao diện chương trình như sau:



Ta tạo các yếu tố hình ảnh trang trí bằng ImageView trong Javafx. Các xe được tạo ra từ các Button kết hợp với hiệu ứng ImageInput để tạo hình ảnh cho Button.

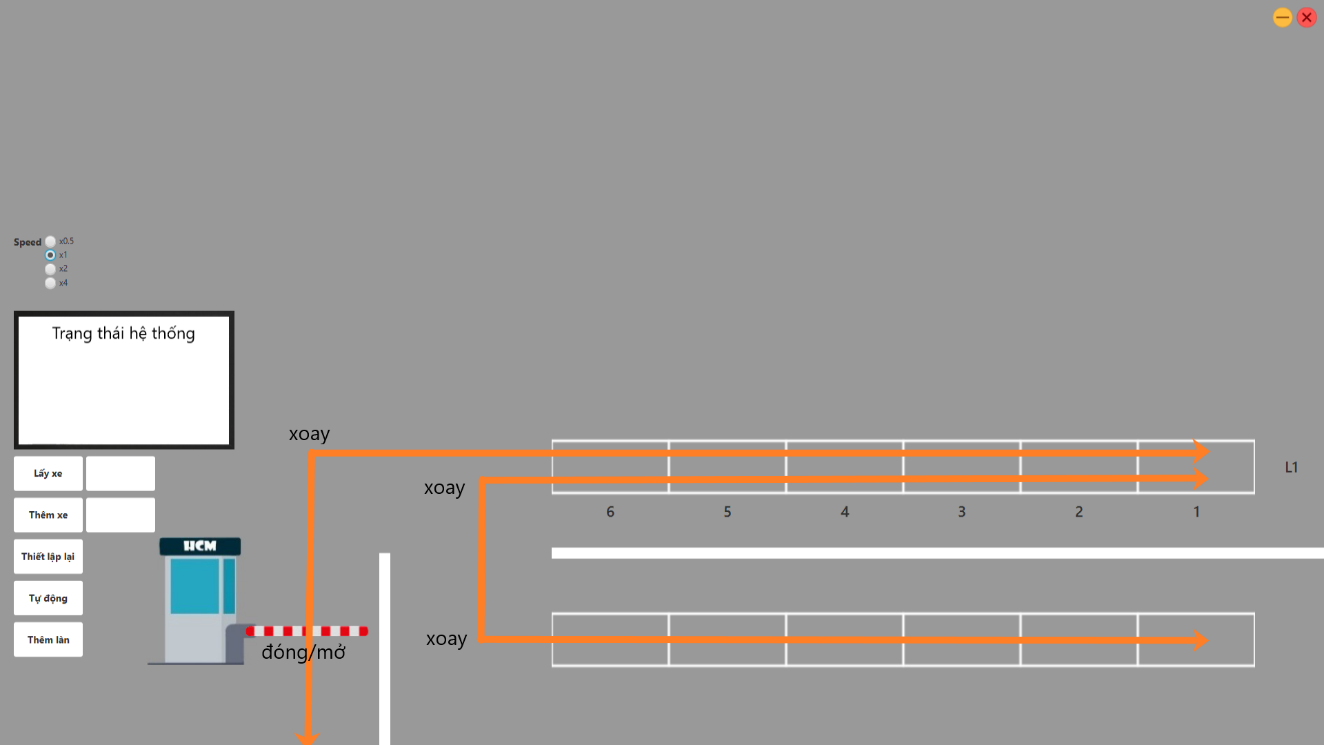
#### Các chức năng của chương trình

1. Thiết kế các yếu tố cần thiết của chương trình

* Các chuyển động trong chương trình:

Giống với chương trình mô phỏng ngăn xếp cơ bản, nhóm cũng sử dụng các công cụ có sẵn trong javafx để thực hiện các chuyển động trong chương trình.

Các chuyển động trên gồm nhiều chuyển động nhỏ để trông thực tế nhất cho người dùng sử dụng. Mô hình chuyển động trong chương trình đa phần đều có dạng như sau:



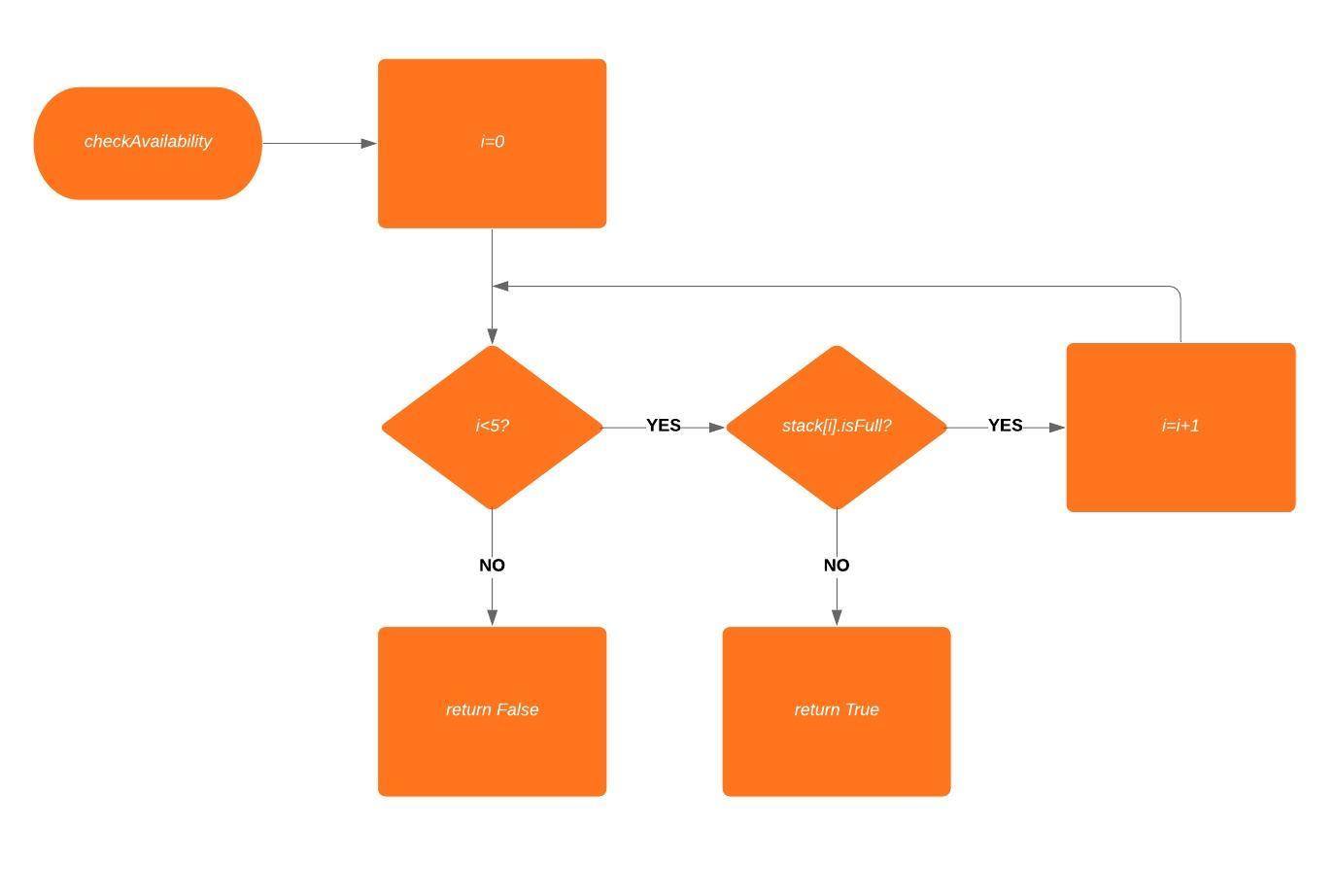
Để thực hiện các di chuyển nhiều bước trên, nhóm sử dụng SequentialTransition trong javafx để kết hợp nhiều chuyển động lại với nhau một cách tuần tự.

* Các vị trí (tọa độ) trong chương trình:

Để có thể chuyển động trong chương trình, trước hết ta cần các vị trí (điểm bắt đầu, điểm kết thúc) của chuyển động. Vì vậy nhóm tạo một class là XY gồm các số x, y để tạo các tọa độ trong chương trình. Trong đó, các tọa độ quan trọng là top của các stack (vị trí đầu của mỗi làn). Đối với mỗi stack, nhóm tạo một đối tượng XY ứng với top của stack đó.

* Kiểm tra trạng thái bãi đỗ xe:

Tạo một hàm checkAvailability() để kiểm tra tình trạng bãi đỗ xe và trả về true nếu bãi đỗ xe còn chỗ và false nếu bãi đỗ xe hết chỗ. Hàm này sẽ dựa trên biến lanes (số làn hiện tại trong bãi đỗ xe) để kiểm tra từng làn một dựa trên chức năng kiểm tra xem có đầy không của stack. Cơ chế hoạt động của hàm này được thể hiện cụ thể như sau:



* Chức năng hiển thị các thông báo:

Đây là chức năng cơ bản của chương trình và hầu hết các chức năng còn lại đều cần có chức năng này.

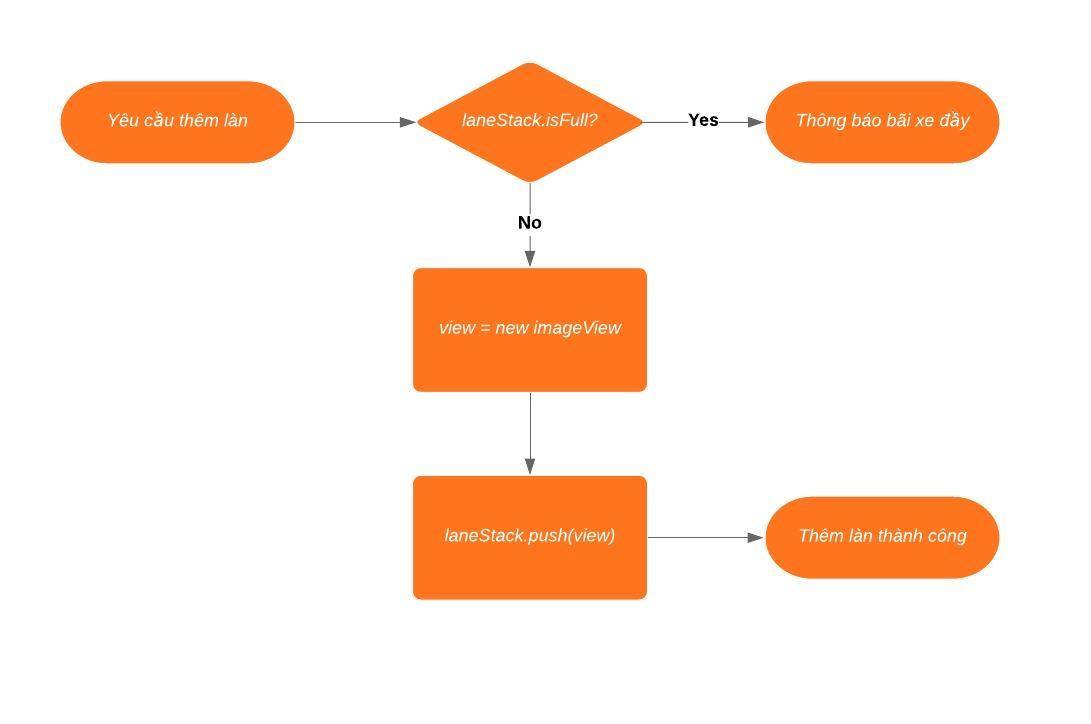
Đầu tiên ta thiết kế một màn hình máy tính như trong phần thiết kế giao diện.

Việc hiển thị thông báo được thực hiện dựa trên cơ chế sau:

Ở bước hiển thị label, ta sử dụng FadeTransition để có thể kết hợp với SequentialTransition tạo thành một chuỗi hành động của chương trình chứ không đơn thuần là tách rời 2 việc ra.

1. Thiết kế các chức năng chung:

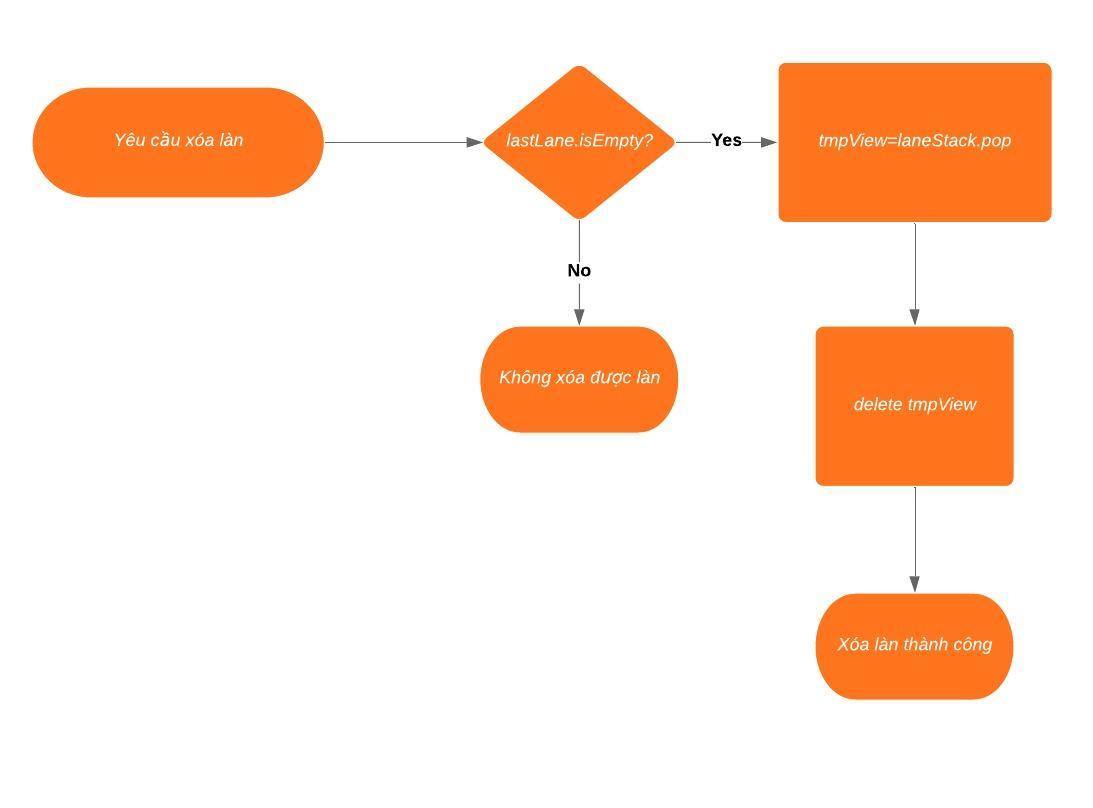
* Chức năng thêm làn:

Nhóm thực hiện chức năng này với thuật toán như sau:

Để quản lý chức năng này, chương trình sử dụng một stack là laneStack. Stack này là stack có các phần tử là ImageView (hình ảnh) và mặc định có 1 phần tử chính là làn chính, số phần tử được thay đổi dựa vào việc thực hiện thêm làn hay bớt làn. Các hoạt động thêm làn được kiểm soát bằng việc laneStack đã đầy chưa. Nếu chưa thì ta tiến hành chèn hình ảnh nên chương trình đồng thời push một phần tử vào laneStack và tăng biến lanes thêm 1. Còn nếu laneStack đã đầy chương trình sẽ hiển thị thông báo.

* Chức năng xóa làn:

Chức năng xóa làn được thực hiện dựa trên thuật toán như sau:

Chức năng này khá đơn giản, chương trình tiến hành kiểm tra xem làn cần xóa hiện đang có xe hay không, nếu không có thì sẽ tiến hành truy cập tới làn bằng laneStack.pop() sau đó thực hiện xóa làn. Còn nếu làn đang có xe, chương trình sẽ hiển thị thông báo không xóa được.

* Chức năng thiết lập lại chương trình (Reset):

Chức năng này có tác dụng thiết lập lại tất cả các giá trị và trạng thái của bãi đỗ xe về lại trạng thái khởi điểm để đáp ứng các nhu cầu của người dùng.

Để thực hiện chức năng này, ta xóa tất cả các Button (xe) đang có trong chương trình, đồng thời thiết lập lại các giá trị vị trị top, xóa hết các phần tử trong các stack, thiết lập lại giá trị lanes về 1, dừng sequence và xóa các lịch hiện có trong sequence. Các thao tác trên đều có thể dễ dàng thực hiện. Riêng đối với việc xóa các phần tử trong stack, ta cần thêm chức năng clear() cho stack. Chức năng này cho phép ta xóa các phần tử trong stack bằng việc chuyển giá trị top của stack về -1.

* Chức năng chuyển đổi chế độ tự động và thủ công:

Chức năng này hoạt động tương tự chức năng thiết lập lại chương trình bên trên, tuy nhiên ta thêm việc thay các nút chức năng của nhóm chức năng thủ công bằng các nút chức năng của nhóm tự động và ngược lại.

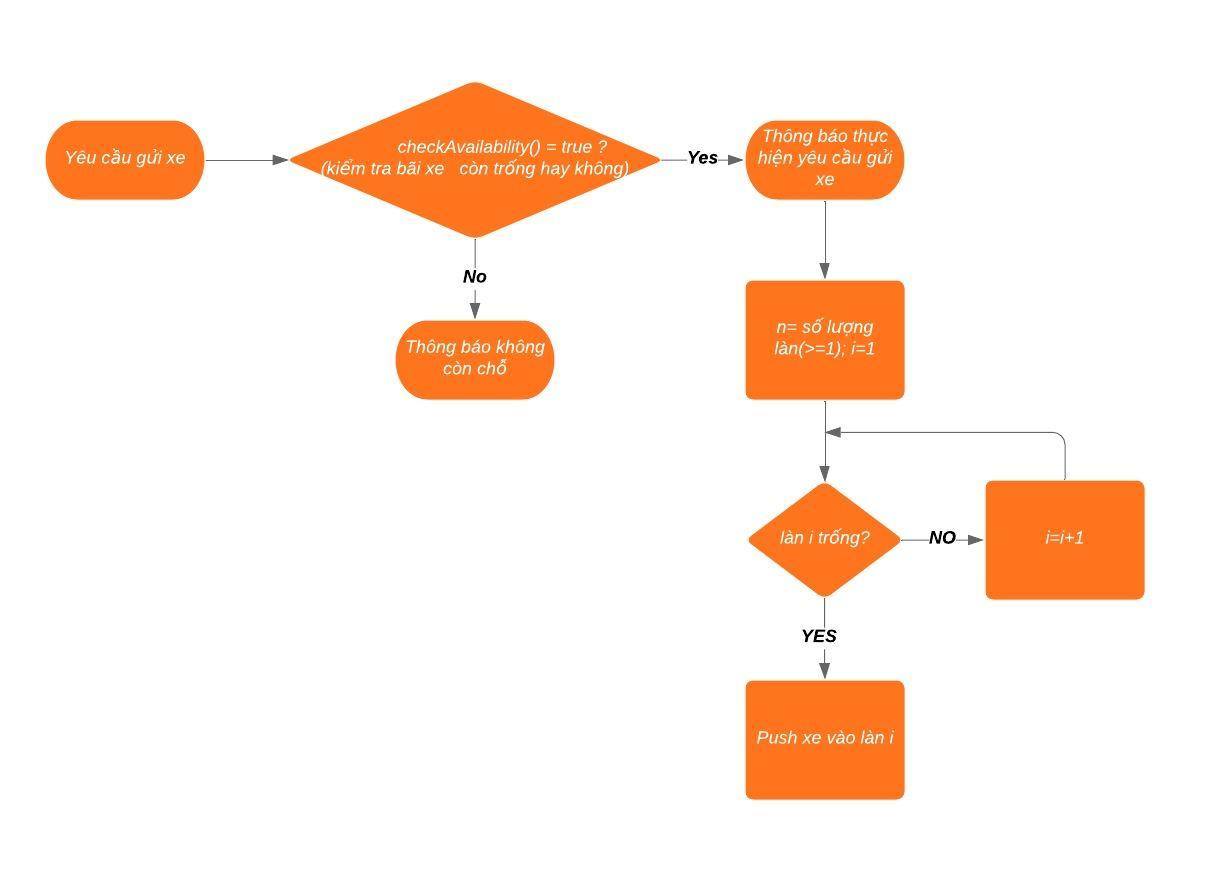
* Chức năng điều chỉnh tốc độ chương trình:

Chức năng này sử dụng một nhóm ToogleGroup gồm các Radio Button (các nút chọn). Mỗi khi ta chọn một giá trị tốc độ, ToogleGroup này sẽ trả về giá trị đó và đồng thời đổi giá trị của biến double SPEED\_VALUE trong chương trình. SPEED\_VALUE này được gắn vào tất cả các chuyển động trong chương trình và sẽ điều chỉnh tốc độ của các chuyển động.

1. Thiết kế nhóm chức năng thủ công:

* Chức năng thêm xe vào bãi đỗ xe:

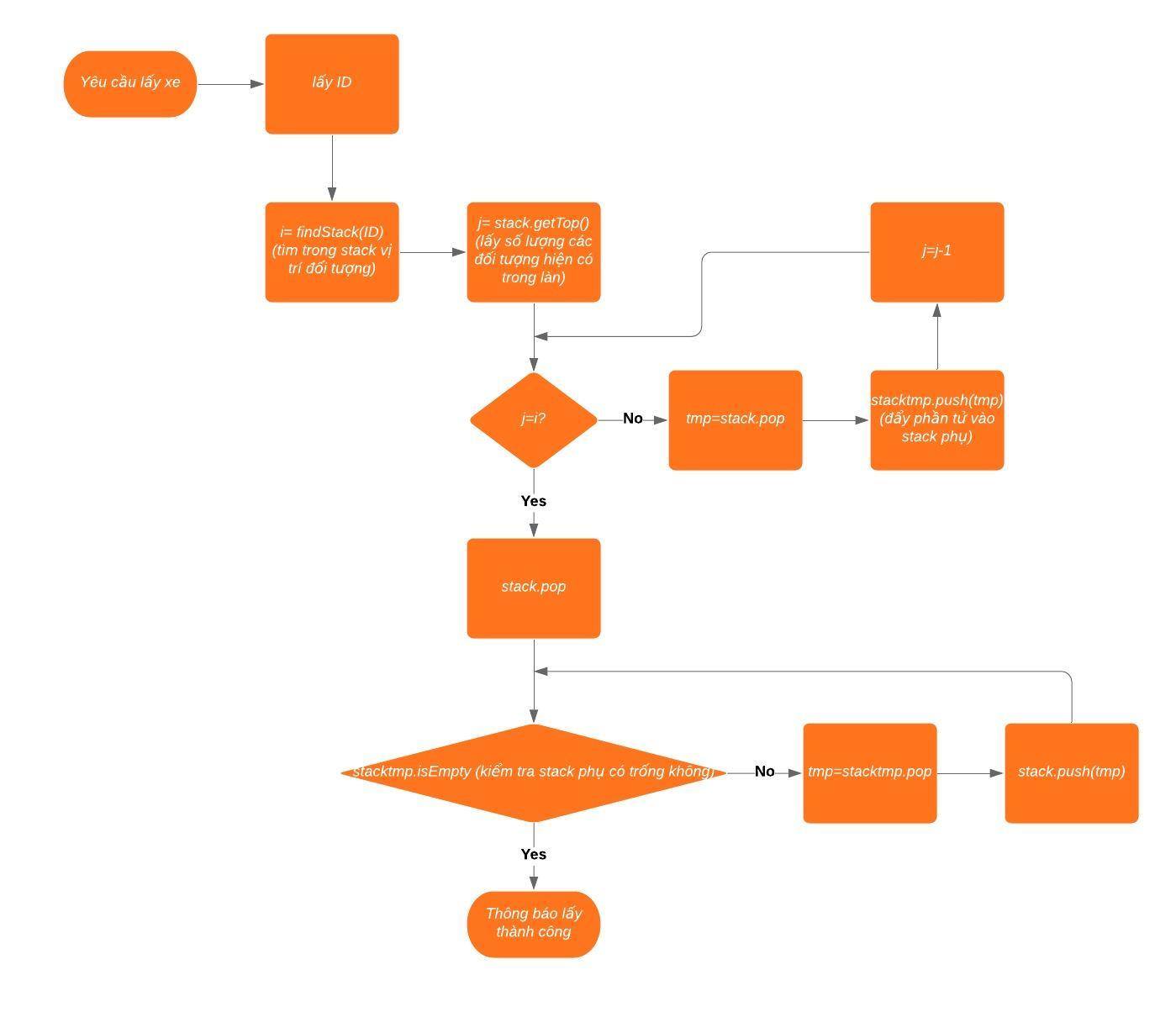
Thuật toán của nhóm được trình bày như sau:

 Đối với phần thêm xe vào làn:

* Người dùng có thể nhập biển số xe (nếu cần) hoặc có thể ghi chú thông tin mình muốn lên xe trước khi thực hiện thêm xe.
* Chương trình tiến hành tạo xe bằng cách tạo một button mới, và thêm Effect ImageInput cho Button để tạo hình cho xe. Sau đó ta sẽ thêm xe này vào chương trình, khi này xe sẽ ở vị trí bên ngoài phần người dùng có thể nhìn được và chỉ có thể thấy khi xe được di chuyển vào trong khi ta tiến tới bước di chuyển xe.
* Sau khi xe được tạo, chương trình bắt đầu thực hiện di chuyển xe. Xe sẽ được chuyển tới vị trí dưới thanh chắn, sau đó chương trình sẽ thực hiện mở thanh chắn và đưa xe vào vị trí top của stack.
* Sau khi hoàn tất quá trình trên, ta cập nhật trạng thái của tọa độ top (vì hiện tại top của stack cũng đã thay đổi).
* Chức năng lấy xe ra khỏi bãi đỗ xe:

Ở phần này, người dùng có 2 lựa chọn, hoặc là lấy xe qua việc kiểm tra biển số xe trùng khớp, hoặc có thể lấy xe bằng cách click trực tiếp vào xe trên chương trình.

Thuật toán của chức năng này được nhóm thực hiện như sau:



Trên sơ đồ trên, findStack là một phương thức được nhóm thêm vào trong đối tượng stack được tạo. findStack sẽ tìm kiếm trong stack và trả về vị trí của phần tử được chọn trong stack. Từ vị trí tìm được này, ta thực hiện thao tác lấy xe ra khỏi làn. Đối với cơ chế lấy xe bằng cách click trực tiếp, ta thực hiện findStack với Button mà ta click vào, còn với cơ chế lấy xe bằng cách tìm biển số, ta thực hiện findStack trên stack mà ta dùng để chứa biển số xe.

* Sau khi biết được vị trí của xe trong làn, ta sử dụng chính con số đó để lập lên 1 vòng lặp. Trong vòng lặp này, mỗi lần lặp, ta truy cập tới xe ngoài cùng của stack, và chuyển xe này xuống dưới top của làn phụ đồng thời cập nhật giá trị của vị trí top của cả làn chính lẫn làn phụ. Quá trình di chuyển này cũng được thực hiện tương đương như trong quá trình di chuyển xe vào trong bãi đỗ ở chức năng bên trên. Sau khi đã hoàn thành vòng lặp trên, lúc này xe cần lấy sẽ ở ngoài cùng của làn và có thể được truy cập.
* Ta lấy xe cần lấy ra khỏi bãi đỗ như cách thêm xe vào bãi đỗ đã được giải thích bên trên nhưng ngược lại. Ta tiến hành chuyển xe ra bên ngoài phần nhìn thấy của chương trình, đồng thời cập nhật vị trí top của làn chính.
* Sau khi đã lấy được xe cần lấy, ta sử dụng con số vị trí của xe để tạo một vòng lặp thích hợp. Vòng lặp này có tác dụng đưa các xe ở dưới làn phụ về lại làn chính đồng thời cập nhật vị trí top của 2 làn.

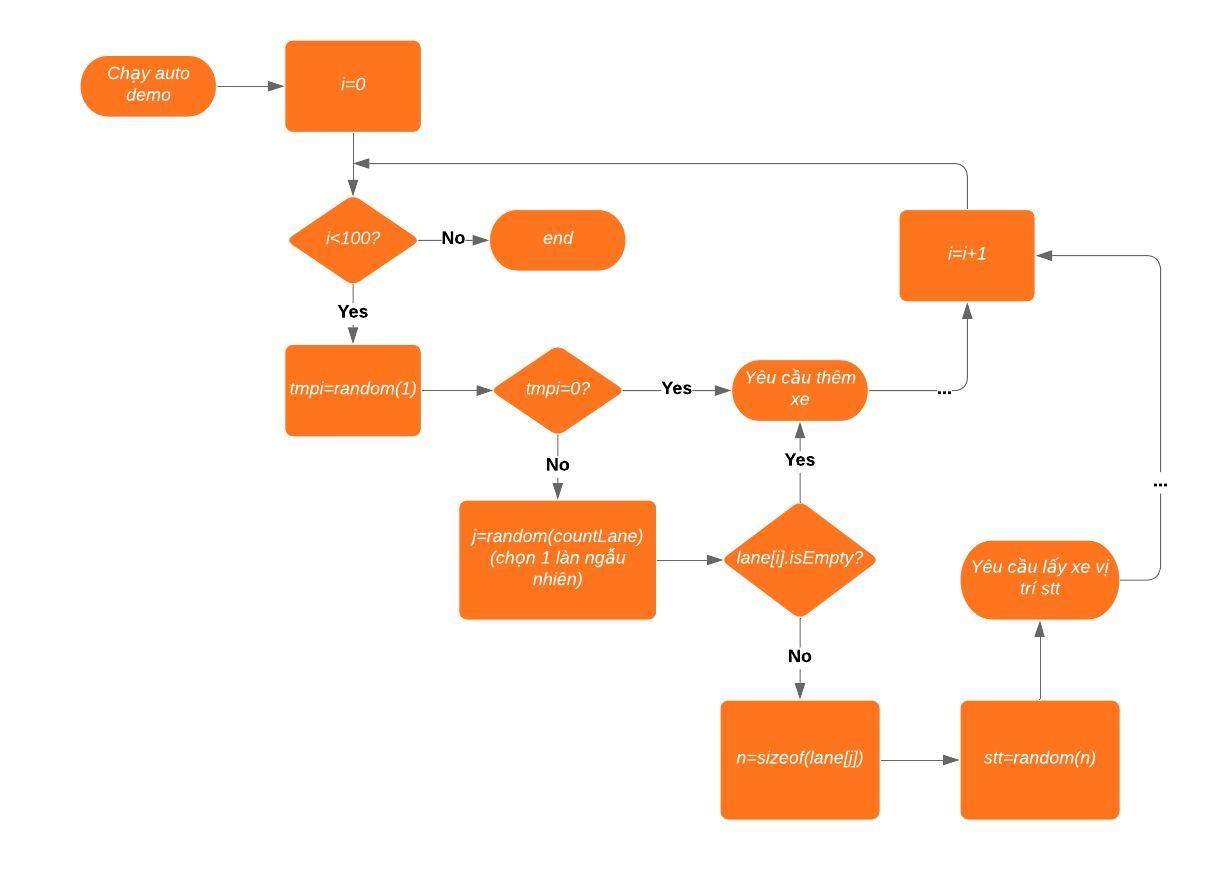
1. Nhóm chức năng tự động:

Nhóm chức năng tự động bao gồm các chức năng bắt đầu chạy tự động, tạm dừng và tiếp tục chạy tự động.

* Chức năng bắt đầu chạy tự động:

Ý tưởng để thực hiện chức năng này là sử dụng một vòng lặp trong đó liên tục tạo các giá trị random với tỉ lệ 50%. Với mỗi giá trị này chương trình sẽ ra yêu cầu thêm xe vào bãi đỗ hoặc lấy xe bất kỳ ra khỏi bãi đỗ.

Vòng lặp này được biểu diễn như sau:



Trong đó các thao tác thêm xe và lấy xe được thực hiện tương tự trong phần thủ công.

Để đảm bảo chương trình tự động chạy liên tục, sau khi chạy vòng lặp, nhóm sử dụng chức năng setCycleCount(Timeline.INFINITE) để đảm bảo chương trình chạy liên tục đến khi nào được dừng.

* Chức năng tạm dừng/ tiếp tục chạy tự động:

Chức năng này sử dụng một chức năng có sẵn của Sequential Transition. Ta tạm dừng sequence bằng method sequence.pause() và tiếp tục chạy bằng method sequence.play().

1. Các chức năng bổ sung:

* Chức năng hiện biển số xe:
* Để tạo biển số xe cho các xe, ta sử dụng Label trong javafx.
* Mỗi khi ta tạo xe trong quá trình cho xe vào bãi đỗ xe, nếu người dùng nhập thêm biển số xe ở bên phần text field bên cạnh, chương trình sẽ tạo một label với nội dung chính là phần text mà người dùng đã nhập.
* Hoạt động của label này sẽ được tích hợp vào tất cả các chức năng khác của chương trình. Ví dụ như khi lấy một xe ra ngoài, biển số của này cũng sẽ được đưa ra ngoài và các biển số của các xe khác cũng được di chuyển về vị trí phù hợp.
* Các hoạt động trên tương tự với hoạt động của các xe (Button) nhưng thay đổi node của hoạt động là các biển số (Label).
* Để quản lý hệ thống biển số xe, nhóm sử dụng các stack label, một hệ thống vị trí mới nằm trên hệ thống thống của xe một chút để khi hiện lên, biển số xe sẽ hiển thị nằm trên xe.
* Để tiện cho việc quan sát, biển số xe khi được tạo sẽ mặc định không hiển thị và chỉ hiển thị khi người dùng chỉ chuột vào xe. Điều này được thực hiện qua các thiết lập cho Button (xe). Nhóm sử dụng setOnMouseEntered để thiết lập hành động khi chỉ chuột vào xe. Khi chuột chỉ vào xe ta sẽ dùng Fade Transition để chuyển giá trị Opacity của biển số xe này từ 0 thành 1. Ngược lại khi chuột ra khỏi xe ta dùng SetOnMouseExited và dùng Fade Transition chuyển giá trị Opacity từ 1 thành 0.
* Quản lý hoạt động của các nút chức năng:

Để hạn chế việc các di chuyển của hệ thống bị mâu thuẫn khi người dùng nhấn các nút chức năng khác nhau trong khi chức năng này vẫn chưa chạy xong, nhóm chọn giải pháp là disable các nút có ảnh hưởng tới hoạt động của chương trình và enable chúng trở lại khi sequence được chạy xong.

* Điều này được thực hiện thông qua method sequence.setOnFinished và điều chỉnh các trạng thái disabled và enabled trong đó.
* Ví dụ: Khi người dùng bấm nút thêm xe vào bãi đỗ, nút này sẽ được disable và người dùng không thể bấm tiếp cho tới khi quá trình thêm xe được thực hiện xong.

# Kết quả - Đánh giá

## Chương trình mô phỏng ngăn xếp đơn giản

### Kết quả:

Chương trình đạt được các yêu cầu được đề ra ban đầu:

* Thực hiện đúng các chức năng của stack.
* Có đầy đủ các chức năng cần thiết.
* Các chuyển động tương đối ổn định.
* Chương trình có cải tiến các yếu tố tinh chỉnh để phù hợp với nhu cầu sử dụng của người dùng, cũng như làm chương trình trông bắt mắt hơn.
* Chương trình đạt được đúng với ý tưởng của môn Project 1, giúp sinh viên hiểu hơn về cấu trúc dữ liệu thuật toán ngăn xếp và một số cấu trúc dữ liệu khác, đồng thời cũng nâng cao kỹ thuật lập trình của sinh viên.

Khi khơi động chương trình, chương trình sẽ vào màn hình chính và có các chức năng như đã trình bày trong báo cáo:



### Đánh giá:

* Chương trình có đầy đủ chức năng theo dự kiến, mô phỏng được các hoạt động của stack cũng như các hiệu ứng di chuyển xuất hiện đầy đủ.
* Tuy nhiên vẫn còn hạn chế trong chương trình:
* Đồ họa còn thô sơ, không đẹp mắt.
* Các phần tử di chuyển khá cứng.
* Code viết chương trình còn lộn xộn, chưa tối ưu.
* Các cải tiến trong tương lai:
* Có thể thêm đồ họa để chương trình sinh động hơn.
* Thêm chức năng di chuyển vị trí 2 stack.
* Cải tiến đường đi khi push pop.
* Tối ưu hóa bảng điều khiển chương trình.

## Chương trình mô phỏng bãi đỗ xe

### Kết quả:

Chương trình đạt được các yêu cầu được đề ra ban đầu:

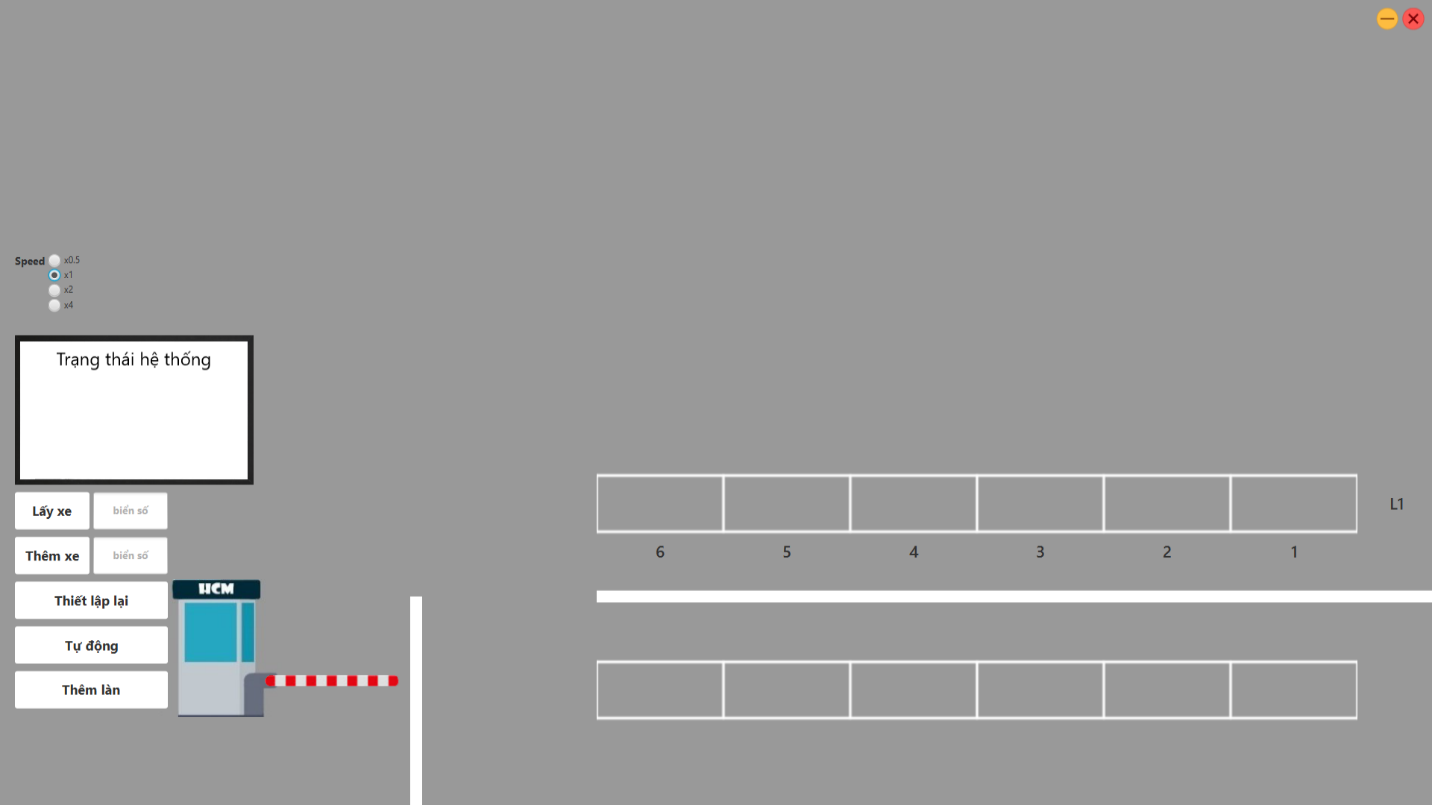
* Có đầy đủ các chức năng cần thiết.
* Yếu tố đồ họa ổn, các chuyển động có sự ổn định.
* Chương trình được tối đa hóa tính thực thế, phù hợp với các nhu cầu chính của người dùng.
* Chương trình đạt được đúng với ý tưởng của môn Project 1, giúp sinh viên hiểu hơn về cấu trúc dữ liệu thuật toán ngăn xếp và một số cấu trúc dữ liệu khác, đồng thời cũng nâng cao kỹ thuật lập trình của sinh viên.

Khi khởi động chương trình, ta sẽ vào trang khởi động:



Các nút bắt đầu và thoát có tác dụng bắt đầu chương trình và thoát chương trình.

Khi nhấn bắt đầu chương trình, người dùng sẽ thấy giao diện chính của chương trình với nhiều các chức năng đã được trình bày:



### Đánh giá:

* Chương trình tương đối hoàn thiện, có đủ các chức năng cần thiết cũng như có yếu tố đồ họa ổn.
* Các cải tiến trong tương lai:
* Sử dụng các hình ảnh đẹp hơn, cải thiện yếu tố đồ họa.
* Cải thiện giao diện chương trình cho hợp lý.
* Cải tiến các phương thức gửi xe, lấy xe (thêm lối ra, thực hiện lấy nhiều xe cùng một lúc…).
* Tinh chỉnh code để cải thiện hiệu năng chương trình.

# Mô tả kế hoạch thực hiện - Đóng góp của các thành viên

## Kế hoạch thực hiện:

* Tuần 3: Nhận đề tài, chọn nhóm trưởng, phân chia nhiệm vụ trong nhóm. Lên kế hoạch thực hiện sơ bộ cho nhóm. Tìm hiểu kiến thức nền tảng về cấu trúc dữ liệu ngăn xếp cũng như một số cấu trúc dữ liệu cần thiết.
* Tuần 4: Tiếp tục tìm hiểu kiến thức nền tảng, đồng thời tìm kiếm ngôn ngữ lập trình cũng như các công cụ phù hợp để thực hiện đề tài.
* Tuần 5: Thống nhất sử dụng ngôn ngữ java cùng với thư viện javafx để thực hiện đề tài. Nghiên cứu sâu về javafx để nắm vững được các kỹ thuật. Tiến hành lên ý tưởng chương trình mô phỏng ngăn xếp cơ bản.
* Tuần 6 - 8 : Tiến hành xây dựng chương trình mô phỏng trực quan hóa ngăn xếp.
* Tuần 9 - 10: Kiểm tra, hoàn thiện chương trình. Bổ sung các tính năng phụ.
* Tuần 11: Lên ý tưởng chương trình mô phỏng bãi đỗ xe. Tìm kiếm các hình ảnh, mô hình liên quan.
* Tuần 12 - 14: Tiến hành xây dựng chương trình mô phỏng bãi đỗ xe. Hoàn thành các chức năng cơ bản.
* Tuần 14 - 15: Kiểm tra, hoàn thiện chương trình. Tiến hành đóng gói 2 chương trình. Tham khảo các mẫu báo cáo để chuẩn bị viết báo cáo.
* Tuần 16 - 17: Viết báo cáo đồ án, quay video demo sản phẩm.
* Tuần 18: Bảo vệ đồ án, nộp sản phẩm.

## Đóng góp của các thành viên:

### Nguyễn Trung Kiên - 20173209:

* Tìm hiểu các kiến thức cơ bản, nắm rõ cấu trúc dữ liệu và các thuật toán.
* Lên ý tưởng sơ bộ chương trình mô phỏng ngăn xếp.
* Góp ý hoàn thiện chương trình mô phỏng ngăn xếp.
* Thiết kế chính chương trình mô phỏng bãi đỗ xe.
* Viết báo cáo đồ án.

### Nguyễn Đăng Trung Kiên – 20173203:

* Tìm hiểu các kiến thức cơ bản, nắm rõ cấu trúc dữ liệu và các thuật toán.
* Lên ý tưởng sơ bộ chương trình mô phỏng bãi đỗ xe.
* Góp ý hoàn thiện chương trình mô phỏng bãi đỗ xe.
* Thiết kế chính chương trình mô phỏng hoạt động ngăn xếp.
* Viết báo cáo đồ án.

Danh mục tài liệu tham khảo

* Nguyễn Đức Nghĩa. “Giáo trình Cấu trúc dữ liệu và Thuật toán”, NXB Bách Khoa, Hà Nội
* “Effective Java”, Joshua Bloch
* “JavaFX Programming Cookbook”, Andreas Pomarolli
* “JavaFX Tutorial \* A Step-by-Step Guide to JavaFX”, Su TP
* Website: JavaFX 8 Oracle oracle.com