ĐẠI HỌC BÁCH KHOA HÀ NỘI

**TRƯỜNG CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

Ảnh có chứa văn bản, Phông chữ, Đồ họa, biểu tượng

Mô tả được tạo tự động

BÁO CÁO BÀI TẬP LỚN

**Học phần: Lập trình hướng đối tượng – IT3103**

**Nhóm 29:**

**ỨNG DỤNG MÔ PHỎNG CÁC THUẬT TOÁN SẮP XẾP**

**INSERTION SORT, BUBBLE SORT, QUICK SORT**

|  |  |  |
| --- | --- | --- |
| Sinh viên thực hiện: | **Phạm Thường Vi** | **20226097** |
|  | **Đinh Đình Hải Việt** | **20225683** |
|  | **Hoàng Quốc Việt** | **20225684** |
|  | **Nguyễn Hoàng Việt** | **20226098** |
|  | **Vũ Hoàng Việt** | **20225777** |
| Mã lớp: | **151965** |  |
| GV hướng dẫn: | **TS. Nguyễn Thị Thu Trang** |  |

**Hà Nội, tháng 12 năm 2024**

[PHÂN CÔNG NHIỆM VỤ 3](#_Toc1761960405)

[1. MÔ TẢ BÀI TOÁN 4](#_Toc515044384)

[1.1 Giao diện chính 5](#_Toc1640092336)

[1.2 Tạo mảng. 5](#_Toc869838936)

[1.3 Hiển thị kết quả 5](#_Toc1626870895)

[2. THIẾT KẾ 6](#_Toc1196601944)

[2.1. Phân tích kế thừa (Inheritance) 11](#_Toc1725175888)

[2.2. Phân tích Association 14](#_Toc148701856)

[2.2.1. SortingAlgorithm và SortElement 14](#_Toc1835443659)

[2.2.2. CreateArrayController và ArrayModel 15](#_Toc834009613)

[2.3. Các mối liên kết giữa các lớp 15](#_Toc1412190372)

[2.3.1. Main và MainMenu 16](#_Toc1571831182)

[2.3.2. MainMenuController và ChooseOptionController 16](#_Toc1224828232)

[2.3.3. ChooseOptionController và CreateArrayController 16](#_Toc1585748657)

[2.3.4. CreateArrayController và SortingAlgorithm 16](#_Toc2015257488)

[2.4. Đa hình 16](#_Toc825530615)

[2.5. Một số phương thức quan trọng 18](#_Toc969255444)

[2.5.1. Phương thức perform() trong các thuật toán sắp xếp 18](#_Toc1842754993)

[2.5.2. Phương thức tạo mảng 19](#_Toc681693686)

# PHÂN CÔNG NHIỆM VỤ

|  |  |  |
| --- | --- | --- |
| Họ và tên | MSSV | Nhiệm vụ |
| Phạm Thường Vi | 20226097 | * Báo cáo (50%) + slide (50%) * Hoàn thiện class diagram chi tiết (thêm package) * Hoàn thiện class diagram tổng quát (thêm package) |
| Nguyễn Hoàng Việt | 20226098 | * Leader * Quản lý các version trên github * Test version 1 * Fix bug version 1 và chỉnh sửa giao diện version 1 * Tạo version cuối cùng và push lên github |
| Hoàng Quốc Việt | 20225684 | * Báo cáo (50%) + slide (50%) |
| Đinh Đình Hải Việt | 20225683 | * Thiết kế use case diagram * Thiết kế class diagram chi tiết * Code version 1 (giao diện chính, giao diện nhập mảng, giao diện thực thi 3 thuật toán) |
| Vũ Hoàng Việt | 20225777 | * Thiết kế class diagram tổng quát * Code version 1 (chỉnh sửa, bổ sung các tính năng cho giao diện thực thi 3 thuật toán: xem từng bước, xem bước trước đó) * Test version 1 và fix bug version 1 |

# MÔ TẢ BÀI TOÁN

1.1 Giao diện chính

Trong giao diện chính (main menu), người dùng có thực hiện những thao tác sau:

* Chọn một trong ba thuật toán sắp xếp (Select sorting type): Quick sort, Bubble sort, Insertion sort.
* Xem giao diện hỗ trợ (View help menu). Giao diện hỗ trợ chứa các thông tin về mục đích và cách sử dụng của ứng dụng.
* Thoát khỏi ứng dụng (Quit application). Khi ấn nút thoát (Quit), hệ thống sẽ hiển thị thông báo xác nhận.

1.2 Tạo mảng

Sau khi chọn một trong ba thuật toán sắp xếp, người dùng có thể tạo mảng (Create new array). Quy trình tạo mảng gồm các bước:

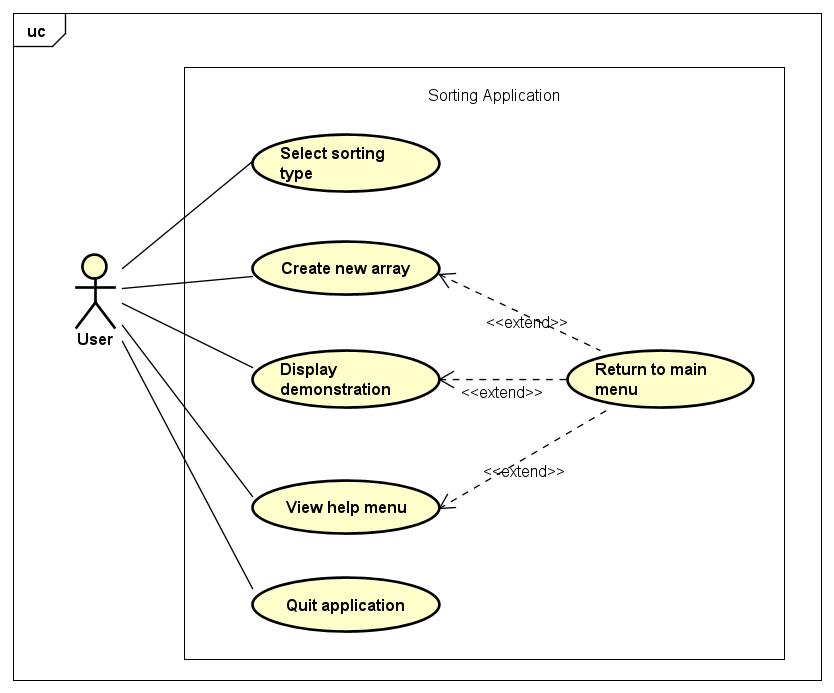
* Người dùng lựa chọn nhập từng phần tử của mảng hoặc tạo mảng ngẫu nhiên.
* Nhập mảng thủ công: Người dùng nhập số phần tử của mảng (quy ước: là một số nguyên dương từ 5 đến 10). Người dùng nhập từng phần tử cho mảng (cách nhau bởi dấu cách). Người dùng ấn xác nhận (Finish), nếu số phần tử chưa đủ, hệ thống sẽ báo lỗi.
* Tạo mảng tự động: Người dùng ấn nút (Show result) để nhận mảng được tạo. Người dùng ấn xác nhận (Finish) để bắt đầu.

Người dùng có thể quay lại main menu (Return to main menu) bất cứ lúc nào (hệ thống có thông báo xác nhận).

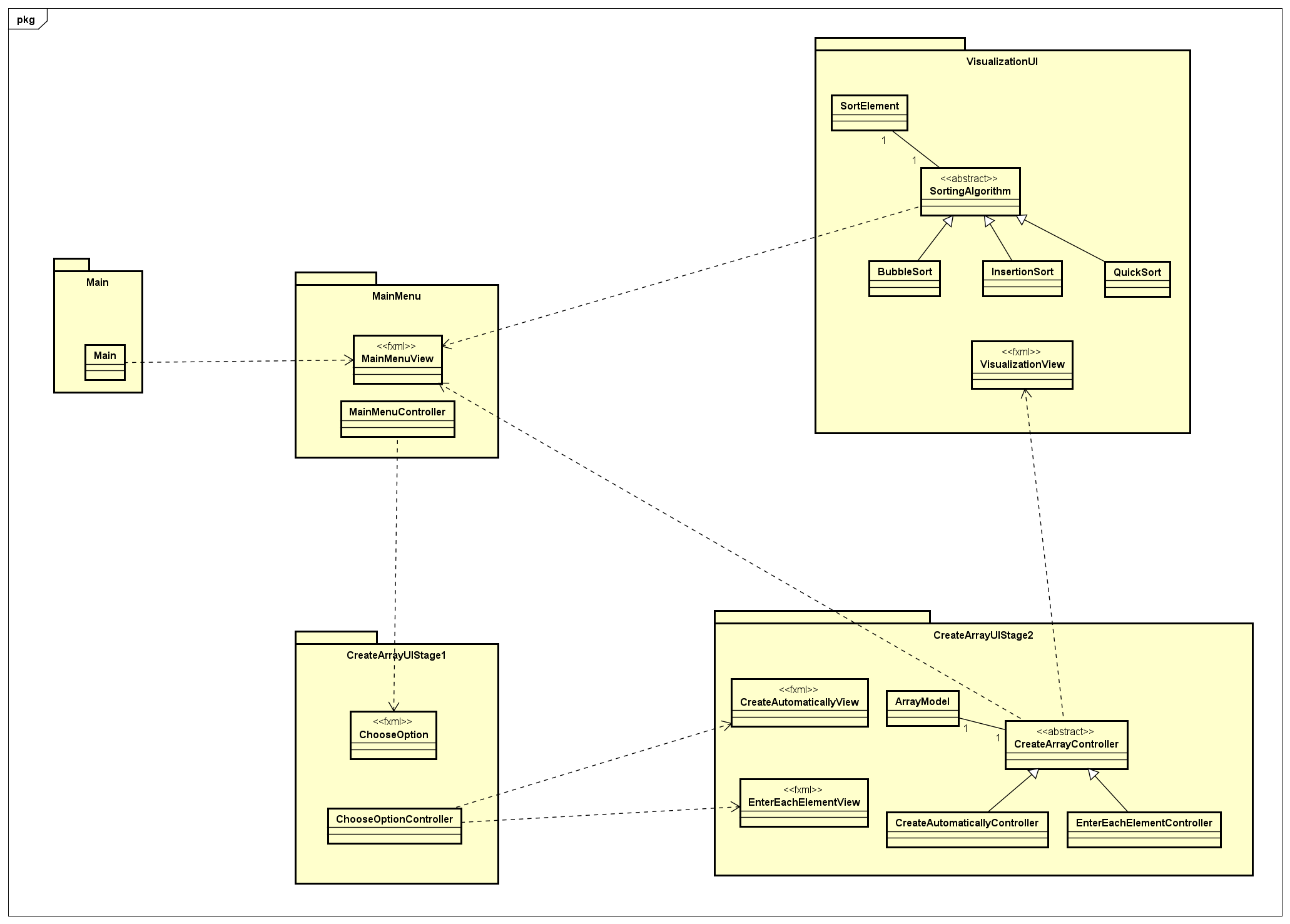
1.3 Hiển thị kết quả

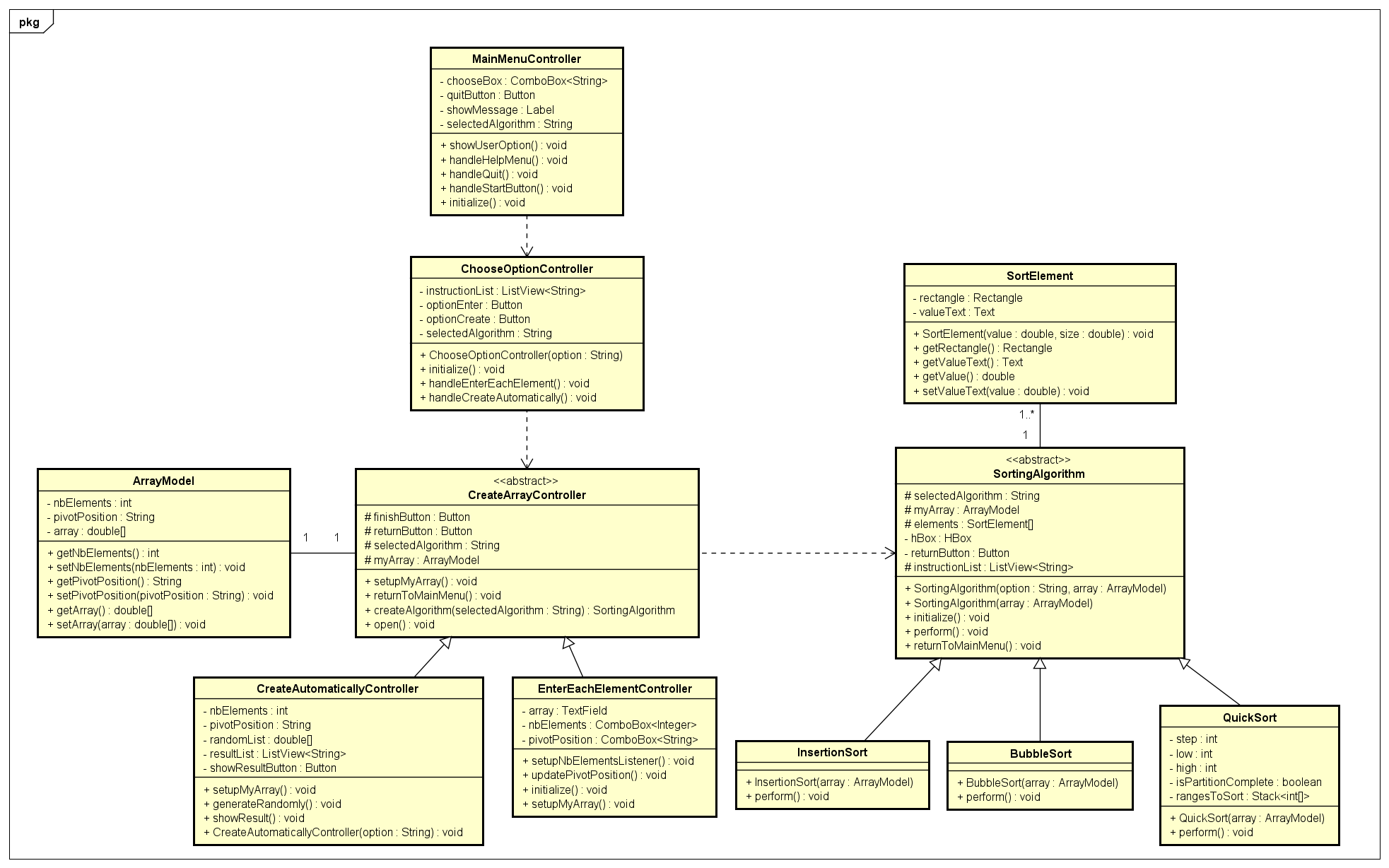
Khi mảng đã được tạo thành công, người dùng có thể xem cách thuật toán hoạt động (Display demostration):

* Người dùng chọn (Start) để bắt đầu, đối với Bubble sort và Insertion sort thuật toán sẽ tự động hiển thị từng bước, còn đối với Quick sort người dùng ấn nút (Start) để bắt đầu và tiếp tục ấn để chuyển bước tiếp theo cho đến khi kết thúc.
* Trong và sau quá trình thuật toán đang được trình diễn, Người dùng có thể quay lại main menu (Return to main menu) bất cứ lúc nào (hệ thống có thông báo xác nhận).



# THIẾT KẾ





**MainMenuController**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Mục đích** | | |
| chooseBox: ComboBox<String> | Menu lựa chọn thuật toán. | | |
| quitButton: Button | Nút để thoát chương trình. | | |
| showMessage: Label | Hiển thị thông báo. | | |
| selectedAlgorithm: String | Thuật toán được chọn. | | |
| **Phương thức** | | **Mục đích** |
| showUserOption() | | Hiển thị lựa chọn của người dùng. |
| handleHelpMenu() | | Hiển thị menu trợ giúp. |
| handleQuit() | | Xử lý khi người dùng nhấn nút thoát. |
| handleStartButton() | | Xử lý khi nhấn nút bắt đầu. |
| initialize() | | Khởi tạo các thành phần giao diện. |

**ChooseOptionController**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **Mục đích** | | |
| instructionList: ListView<String> | Danh sách hướng dẫn. | | |
| optionEnter: Button | Nút chọn nhập thủ công. | | |
| optionCreate: Button | Nút chọn tạo dữ liệu tự động. | | |
| selectedAlgorithm: String | Thuật toán được chọn. | | |
| **Phương thức** | | **Mục đích** |
| ChooseOptionController(option: String) | | Hàm khởi tạo. |
| initialize() | | Khởi tạo các thành phần giao diện. |
| handleEnterEachElement() | | Xử lý khi nhập từng phần tử. |
| handleCreateAutomatically() | | Xử lý khi tạo dữ liệu tự động. |

**ArrayModel**

|  |  |
| --- | --- |
| **Thuộc tính** | **Mục đích** |
| nbElements: int | Số lượng phần tử trong mảng. |
| pivotPosition: String | Vị trí của phần tử pivot (nếu cần). |
| array: double[] | Mảng dữ liệu. |
| **Phương thức** | **Mục đích** |
| getNbElements() | Trả về số lượng phần tử. |
| setNbElements(nbElements: int) | Đặt số lượng phần tử. |
| getPivotPosition() | Trả về vị trí pivot. |
| setPivotPosition(pivotPosition: String) | Cập nhật vị trí pivot. |
| getArray() | Lấy mảng dữ liệu. |
| setArray(array: double[]) | Đặt mảng dữ liệu. |

**CreateArrayController**

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | | **Mục đích** | |
| finishButton: Button | | Nút hoàn tất tạo mảng. | |
| returnButton: Button | | Nút quay lại menu chính. | |
| selectedAlgorithm: String | | Thuật toán được chọn. | |
| myArray: ArrayModel | | Đối tượng chứa mảng dữ liệu. | |
| **Phương thức** | **Mục đích** | |
| setupMyArray() | Thiết lập đối tượng mảng. | |
| returnToMainMenu() | Trở về menu chính. | |
| createAlgorithm(selectedAlgorithm: String) | Tạo đối tượng thuật toán. | |
| open() | Mở giao diện. | |

**CreateAutomaticallyController**

|  |  |
| --- | --- |
| **Thuộc tính** | **Mục đích** |
| nbElements: int | Số lượng phần tử. |
| pivotPosition: String | Vị trí pivot. |
| randomList: double[] | Mảng ngẫu nhiên. |
| showResultButton: Button | Nút hiển thị kết quả tạo mảng. |
| **Phương thức** | **Mục đích** |
| setupMyArray() | Thiết lập đối tượng mảng. |
| generateRandom() | Tạo mảng ngẫu nhiên. |
| showResult() | Hiển thị kết quả. |
| CreateAutomaticallyController(option: String) | Hàm khởi tạo. |

**EnterEachElementController**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Mục đích** | |
| array: TextField | Nhập mảng dữ liệu. | |
| nbElements: ComboBox<Integer> | Số lượng phần tử. | |
| pivotPosition: ComboBox<String> | Nhập vị trí pivot. | |
| **Phương thức** | | **Mục đích** |
| setupNbElementsListener() | | Lắng nghe và xử lý thay đổi số lượng phần tử. |
| updatePivotPosition() | | Cập nhật vị trí pivot. |
| setupMyArray() | | Thiết lập đối tượng mảng. |

**SortElement**

|  |  |
| --- | --- |
| **Thuộc tính** | **Mục đích** |
| rectangle: Rectangle | Biểu diễn phần tử dưới dạng hình chữ nhật. |
| valueText: Text | Giá trị của phần tử. |
| **Phương thức** | **Mục đích** |
| SortElement(value: double, size: double) | Khởi tạo phần tử. |
| getRectangle() | Trả về hình chữ nhật đại diện. |
| getValueText() | Trả về giá trị dạng văn bản. |
| getValue() | Trả về giá trị số. |
| setValueText(value: double) | Cập nhật giá trị. |

**SortingAlgorithm**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Mục đích** | |
| selectedAlgorithm: String | Thuật toán được chọn. | |
| myArray: ArrayModel | Mảng dữ liệu cần sắp xếp. | |
| elements: SortElement[] | Mảng phần tử để sắp xếp. | |
| hBox: HBox | Giao diện chứa các phần tử. | |
| returnButton: Button | Nút quay lại menu chính. | |
| instructionList: ListView<String> | Hướng dẫn và hiển thị từng bước sắp xếp. | |
| **Phương thức** | **Mục đích** |
| SortingAlgorithm(option: String, array: ArrayModel) | Hàm khởi tạo. |
| initialize() | Khởi tạo giao diện. |
| perform() | Thực hiện thuật toán. |
| returnToMainMenu() | Quay lại menu chính. |

**InsertionSort**

|  |  |
| --- | --- |
| **Phương thức** | **Mục đích** |
| InsertionSort(array: ArrayModel) | Hàm khởi tạo thuật toán. |
| perform() | Thực hiện sắp xếp chèn. |

**BubbleSort**

|  |  |
| --- | --- |
| **Phương thức** | **Mục đích** |
| BubbleSort(array: ArrayModel) | Hàm khởi tạo thuật toán. |
| perform() | Thực hiện sắp xếp nổi bọt. |

**QuickSort**

|  |  |  |
| --- | --- | --- |
| **Thuộc tính** | **Mục đích** | |
| step: int | Biến điều khiển các bước trong quá trình phân hoạch (partition) của thuật toán. | |
| low: int | Chỉ số đầu của phạm vi đang xét trong mảng cần sắp xếp. | |
| high: int | Chỉ số cuối của phạm vi đang xét trong mảng cần sắp xếp. | |
| isPartitionComplete: boolean | Cờ đánh dấu quá trình phân hoạch hiện tại đã hoàn tất. | |
| rangesToSort: Stack<int[]> | Stack lưu các phạm vi cần tiếp tục sắp xếp sau khi hoàn thành phân hoạch. | |
| **Phương thức** | | **Mục đích** |
| QuickSort(array: ArrayModel) | | Hàm khởi tạo thuật toán. |
| perform() | | Thực hiện sắp xếp nhanh. |

## 2.1. Phân tích kế thừa (Inheritance)

2.1.1. Nhóm kế thừa : CreateArrayController

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, chữ viết tay

Mô tả được tự động tạo

**CreateArrayController**

Mục đích: Đóng vai trò là lớp cơ sở để xử lý việc tạo mảng ban đầu, cung cấp các phương thức và thuộc tính chung cho các cách tạo mảng khác nhau.

Thuộc tính chung:

+ finishButton và returnButton: điều khiển giao diện.

+ selectedAlgorithm: thuật toán sắp xếp được chọn.

+ myArray: đối tượng mảng cần sắp xếp.

Phương thức chung:

+ setupMyArray(): thiết lập đối tượng mảng (abstract)

+ returnToMainMenu(): quay lại menu chính.

+ createAlgorithm(selectedAlgorithm: String): tạo đối tượng thuật toán sắp xếp tương ứng dựa trên lựa chọn.

**CreateAutomaticallyController**

Thuộc tính riêng:

+ randomList: mảng ngẫu nhiên đã tạo.

+ showResultButton: Nút để hiển thị kết quả mảng đã tạo.

Phương thức riêng:

+ generateRandom(): Sinh các giá trị ngẫu nhiên cho mảng.

+ showResult(): Hiển thị mảng sau khi tạo.

+ setupMyArray(): tạo mảng tự động (ghi đè)

**EnterEachElementController**

Thuộc tính riêng:

+ array và nbElements: Xử lý thông tin mảng do người dùng nhập.

+ pivotPosition: Chọn pivot trong quá trình tạo mảng.

Phương thức riêng:

+ setupNBElementsListener(): Lắng nghe sự kiện nhập số phần tử.

+ updatePivotPosition(): Cập nhật vị trí pivot

+ setupMyArray(): tạo mảng thủ công (ghi đè)

2.1.2. Nhóm kế thừa : SortingAlgorithm

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, biểu đồ

Mô tả được tự động tạo

**SortingAlgorithm**

Mục đích: Cung cấp khuôn mẫu cho tất cả các thuật toán sắp xếp.

Thuộc tính chung:

+ selectedAlgorithm: Tên thuật toán sắp xếp.

+ myArray: Đối tượng ArrayModel chứa dữ liệu cần sắp xếp.

+ elements: Mảng các phần tử cần sắp xếp (SortElement[]).

+ returnButton: Nút quay lại menu chính.

+ instructionList: Lưu hướng dẫn và mô tả quá trình sắp xếp.

Phương thức chung:

+ initialize(): Khởi tạo giao diện.

+ perform(): Thực thi thuật toán sắp xếp (abstract).

+ returnToMainMenu(): Quay lại menu chính.

**InsertionSort**

Phương thức riêng: Ghi đè phương thức perform() để thực hiện thuật toán sắp xếp chèn.

**BubbleSort**

Phương thức riêng: Ghi đè phương thức perform() để thực hiện thuật toán sắp xếp nổi bọt.

**QuickSort**

Thuộc tính riêng:

+ step, low, high: Dùng để quản lý trạng thái của thuật toán.

+ isPartitionComplete: Theo dõi trạng thái phân hoạch.

+ rangesToSort: Stack quản lý các phạm vi cần sắp xếp.

Phương thức riêng: Ghi đè phương thức perform() để thực hiện thuật toán sắp xếp nhanh.

## 2.2. Phân tích Association

### 2.2.1. SortingAlgorithm và SortElement

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tự động tạo

SortingAlgorithm sử dụng một danh sách các đối tượng SortElement để biểu diễn và thao tác với các phần tử trong quá trình sắp xếp.

**Ý nghĩa:**

Biểu diễn trực quan dễ dàng:

* SortElement giúp tách rời việc xử lý logic sắp xếp khỏi việc hiển thị trực quan.
* SortingAlgorithm chỉ cần thao tác trên các đối tượng SortElement mà không cần quan tâm đến cách hiển thị chúng trên giao diện.

Tái sử dụng:

* SortElement có thể được sử dụng cho bất kỳ thuật toán sắp xếp nào (QuickSort, BubbleSort, InsertionSort).
* Mọi thuật toán đều hoạt động trên cùng một cấu trúc dữ liệu (SortElement[]), giúp tái sử dụng và giảm mã lặp.

Mở rộng dễ dàng:

* Nếu cần thay đổi cách hiển thị hoặc thêm thuộc tính (ví dụ: kích thước hình chữ nhật), chỉ cần sửa trong lớp SortElement, không cần thay đổi thuật toán sắp xếp.

### 2.2.2. CreateArrayController và ArrayModel

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, số

Mô tả được tự động tạoCreateArrayController sử dụng ArrayModel như một thuộc tính (myArray).

Một đối tượng CreateArrayController liên kết với một đối tượng ArrayModel.

**Ý nghĩa:**

Tách biệt trách nhiệm:

* ArrayModel chỉ chịu trách nhiệm lưu trữ và quản lý dữ liệu mảng.
* CreateArrayController chịu trách nhiệm tạo hoặc khởi tạo mảng dựa trên yêu cầu.

=> Điều này giúp mỗi lớp tập trung vào chức năng riêng, dễ dàng mở rộng hoặc thay đổi mà không ảnh hưởng đến lớp kia.

Tăng tính linh hoạt:

* CreateArrayController có thể thao tác với nhiều đối tượng ArrayModel khác nhau (nếu cần), không bị ràng buộc với một instance duy nhất.

Dễ dàng mở rộng:

* Khi cần thêm các cách tạo mảng mới, các lớp con của CreateArrayController chỉ cần làm việc với ArrayModel mà không cần thay đổi cấu trúc lưu trữ dữ liệu.

## 2.3. Các mối liên kết giữa các lớp

### 2.3.1. Main và MainMenu

* **Mối quan hệ:**
  + Lớp Main khởi chạy ứng dụng và liên kết với lớp MainMenuController.
* **Phân tích:**
  + Lớp Main chịu trách nhiệm khởi tạo giao diện chính thông qua tệp MainMenuView.fxml.
  + MainMenuController quản lý logic của giao diện chính, bao gồm các chức năng chọn thuật toán, hiển thị trợ giúp, hoặc thoát chương trình.

### 2.3.2. MainMenuController và ChooseOptionController

* **Mối quan hệ:**
  + Lớp MainMenuController gọi đến ChooseOptionController khi người dùng chọn tạo mảng.
* **Phân tích:**
  + Đây là mối liên kết (association) giữa hai lớp controller, MainMenuController chỉ khởi tạo ChooseOptionController khi cần thiết, và hai lớp hoạt động độc lập.

### 2.3.3. ChooseOptionController và CreateArrayController

* **Mối quan hệ:**
  + Lớp ChooseOptionController liên kết đến CreateArrayController thông qua các lựa chọn của người dùng.
* **Phân tích:**
  + ChooseOptionController xử lý logic để xác định người dùng muốn tạo mảng bằng cách nào (tự động hay thủ công). Sau đó, nó khởi tạo đối tượng tương ứng (CreateAutomaticallyController hoặc EnterEachElementController).

### 2.3.4. CreateArrayController và SortingAlgorithm

* **Mối quan hệ**: CreateArrayController **tạo và sử dụng** SortingAlgorithm.
* **Phân tích:**
* Mục đích của phương thức **createAlgorithm** trong CreateArrayController là tạo ra một đối tượng SortingAlgorithm dựa trên tham số selectedAlgorithm.
* Ví dụ: Khi người dùng chọn "QuickSort" hoặc "BubbleSort", createAlgorithm sẽ khởi tạo lớp con tương ứng của SortingAlgorithm (QuickSort, BubbleSort, hoặc InsertionSort).

## 2.4. Đa hình

|  |  |  |
| --- | --- | --- |
| Lợi ích | CreateArrayController và các con | SortingAlgorithm và các con |
| Giao diện chung | Giao diện thống nhất cho các cách tạo mảng khác nhau. | Giao diện thống nhất cho các thuật toán sắp xếp khác nhau. |
| Giảm phụ thuộc | Logic xử lý không phụ thuộc vào cách tạo mảng cụ thể. | Mã gọi thuật toán không phụ thuộc vào thuật toán cụ thể. |
| Dễ mở rộng | Nếu muốn thêm một cách tạo mảng mới (VD: tạo mảng từ file dữ liệu) chỉ cần: Tạo một lớp mới kế thừa CreateArrayController, ghi đè phương thức setupMyArray(), không cần thay đổi mã trong lớp cha hay các lớp hiện có. | Khi cần thêm thuật toán sắp xếp mới (ví dụ: MergeSort), chỉ cần tạo một lớp con mới kế thừa từ SortingAlgorithm và ghi đè phương thức perform(), không cần thay đổi mã hiện tại. |
| Tái sử dụng mã | Tái sử dụng các tác vụ: trở về menu chính, tạo đối tượng thuật toán, mở giao diện | Tái sử dụng giao diện và các thuộc tính như mảng, thuật toán được chọn,...hàm khởi tạo giao diện, hàm quay lại menu chính. |
| Dễ kiểm tra, bảo trì | Các lớp con độc lập, dễ sửa lỗi hoặc thay đổi logic tạo mảng mà không ảnh hưởng đến lớp khác. | Các lớp con độc lập, dễ sửa lỗi hoặc thay đổi logic thuật toán mà không ảnh hưởng đến lớp khác. |
| Lợi ích cho người dùng | Cho phép người dùng linh hoạt chọn cách tạo mảng phù hợp (ngẫu nhiên, nhập tay,...). | Người dùng dễ dàng chọn thuật toán sắp xếp mà không cần thay đổi cách sử dụng chương trình. |
| Tối ưu hóa runtime | Nhờ cơ chế dynamic method dispatch trong Java, khi gọi setupMyArray() từ lớp cha, JVM tự động thực thi phương thức phù hợp của lớp con. Điều này giúp chương trình linh hoạt mà không cần kiểm tra thủ công lớp của đối tượng bằng các câu lệnh if-else hoặc switch. | Nhờ cơ chế dynamic method dispatch trong Java, khi gọi perform() từ lớp cha, JVM tự động thực thi phương thức phù hợp của lớp con. Điều này giúp chương trình linh hoạt mà không cần kiểm tra thủ công lớp của đối tượng bằng các câu lệnh if-else hoặc switch. |

## 2.5. Một số phương thức quan trọng

### 2.5.1. Phương thức perform() trong các thuật toán sắp xếp

**a) BubbleSort:**

* **Mục đích**: Thực hiện thuật toán BubbleSort, nơi các phần tử được so sánh và hoán đổi cho đến khi mảng được sắp xếp hoàn toàn.
* **Chi tiết hoạt động**:
  + **Bước 1**: Duyệt qua mảng và so sánh từng cặp phần tử kề nhau (sử dụng các chỉ số i và j).
  + **Bước 2**: Nếu phần tử tại vị trí j lớn hơn phần tử tại j+1, chúng sẽ được hoán đổi. Phần tử này sẽ được đánh dấu bằng màu **Color.YELLOW** để thể hiện chúng đang được so sánh.
  + **Bước 3**: Sau mỗi lần hoán đổi hoặc so sánh, các phần tử sẽ được cập nhật lại màu sắc (về **Color.LIGHTGRAY** nếu không còn xử lý, hoặc **Color.LIGHTGREEN** khi hoàn thành).
  + **Mô phỏng bằng Timeline**: Mỗi bước của thuật toán được thực hiện theo thời gian, giúp người dùng trực quan hóa quá trình sắp xếp. Khi mảng đã hoàn thành, tất cả các phần tử sẽ chuyển sang màu **Color.LIGHTGREEN**.

**b) InsertionSort:**

* **Mục đích**: Thực hiện thuật toán InsertionSort, nơi phần tử tại vị trí i được chèn vào vị trí đúng trong phần mảng đã được sắp xếp trước đó.
* **Chi tiết hoạt động**:
  + **Bước 1**: Đặt phần tử tại chỉ số i làm "key" (phần tử cần chèn).
  + **Bước 2**: So sánh phần tử "key" với các phần tử trước đó trong mảng (sử dụng vòng lặp while qua chỉ số j).
  + **Bước 3**: Nếu phần tử tại chỉ số j lớn hơn "key", nó sẽ được dịch chuyển sang phải.
  + **Bước 4**: Khi tìm thấy vị trí đúng cho "key", nó sẽ được chèn vào vị trí đó.
  + **Trực quan hóa**: Các phần tử được đánh dấu bằng màu **Color.YELLOW** khi đang xử lý, và màu **Color.LIGHTGRAY** khi đã hoàn thành. Khi tất cả các phần tử đã được sắp xếp, chúng sẽ chuyển sang màu **Color.LIGHTGREEN**.

**c) QuickSort:**

* **Mục đích**: Thực hiện thuật toán QuickSort, nơi mảng được chia thành các phần con dựa trên một phần tử gọi là **pivot** và đệ quy sắp xếp các phần con này.
* **Chi tiết hoạt động**:
  + **Bước 1**: Chọn phần tử **pivot** (phần tử cuối trong mảng) và đánh dấu nó bằng màu **Color.ORANGE**.
  + **Bước 2**: Duyệt qua mảng, so sánh từng phần tử với **pivot**. Nếu phần tử nhỏ hơn hoặc bằng **pivot**, sẽ hoán đổi phần tử đó với phần tử tại chỉ số i.
  + **Bước 3**: Sau khi duyệt hết mảng, hoán đổi phần tử **pivot** với phần tử tại vị trí i+1 để đặt **pivot** vào đúng vị trí của nó.
  + **Bước 4**: Đệ quy tiếp tục sắp xếp hai phần con mảng bên trái và bên phải của **pivot**.
  + **Trực quan hóa**: Các phần tử được đánh dấu bằng màu **Color.YELLOW** khi đang xử lý và **Color.LIGHTGRAY** sau khi hoàn thành. **pivot** sẽ có màu **Color.ORANGE**, và sau khi hoàn thành, các phần tử chuyển sang **Color.LIGHTGREEN**.

### 2.5.2. Phương thức tạo mảng

**a) Tạo mảng thủ công (setupMyArray() trong EnterEachElementController):**

* **Mục đích**: Cho phép người dùng nhập vào các phần tử của mảng thủ công và kiểm tra tính hợp lệ.
* **Chi tiết hoạt động:**
* **Lấy dữ liệu từ người dùng**:
  + **String input**: Lấy dữ liệu từ một trường nhập liệu (textfield).
  + Các phần tử được tách ra bằng khoảng trắng thông qua **input.split("\\s+")**.
* **Kiểm tra và xử lý từng giá trị đầu vào**:
  + Mỗi giá trị được kiểm tra để đảm bảo nó là số hợp lệ:
    - **Hợp lệ**: Giá trị được chuyển đổi thành double và lưu vào mảng.
    - **Không hợp lệ**: Hiển thị thông báo lỗi qua hộp thoại **Alert**.
* **Kiểm tra số lượng phần tử**:
  + Nếu số lượng phần tử nhập vào khớp với số lượng được chọn (nbElements), mảng sẽ được lưu trữ trong đối tượng ArrayModel:
    - **Lưu mảng**: Gọi **myArray.setArray(list)**.
    - **Thiết lập pivot (nếu cần)**: Nếu thuật toán Quick Sort được chọn, xác định vị trí pivot thông qua **pivotPosition**.
    - Chuyển giao diện sang bước tiếp theo với **open()**.
  + **Không khớp số lượng**: Hiển thị thông báo lỗi qua hộp thoại **Alert**.

**b) Tạo mảng tự động (setupMyArray() trong CreateAutomaticallyController):**

* **Mục đích**: Tạo một mảng số ngẫu nhiên với số lượng phần tử do người dùng chỉ định.
* Chi tiết hoạt động:
* **Thiết lập số lượng phần tử**:
* Lấy số lượng phần tử từ biến **nbElements** (do người dùng chọn trước đó qua một spinner hoặc mặc định).
* **Sinh giá trị ngẫu nhiên**:
* Mảng **randomList** đã được tạo sẵn với các giá trị ngẫu nhiên (giả định nó đã được khởi tạo trước hoặc là tham số được truyền vào phương thức).
* **Lưu mảng vào ArrayModel**:
* **myArray.setNbElements(nbElements)**: Lưu số lượng phần tử.
* **myArray.setArray(randomList)**: Lưu mảng ngẫu nhiên.
* Nếu thuật toán được chọn là Quick Sort, thiết lập vị trí pivot thông qua **myArray.setPivotPosition(pivotPosition)**.
* **Chuyển giao diện**:
* Gọi phương thức **open()** để chuyển sang giao diện tiếp theo.