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**

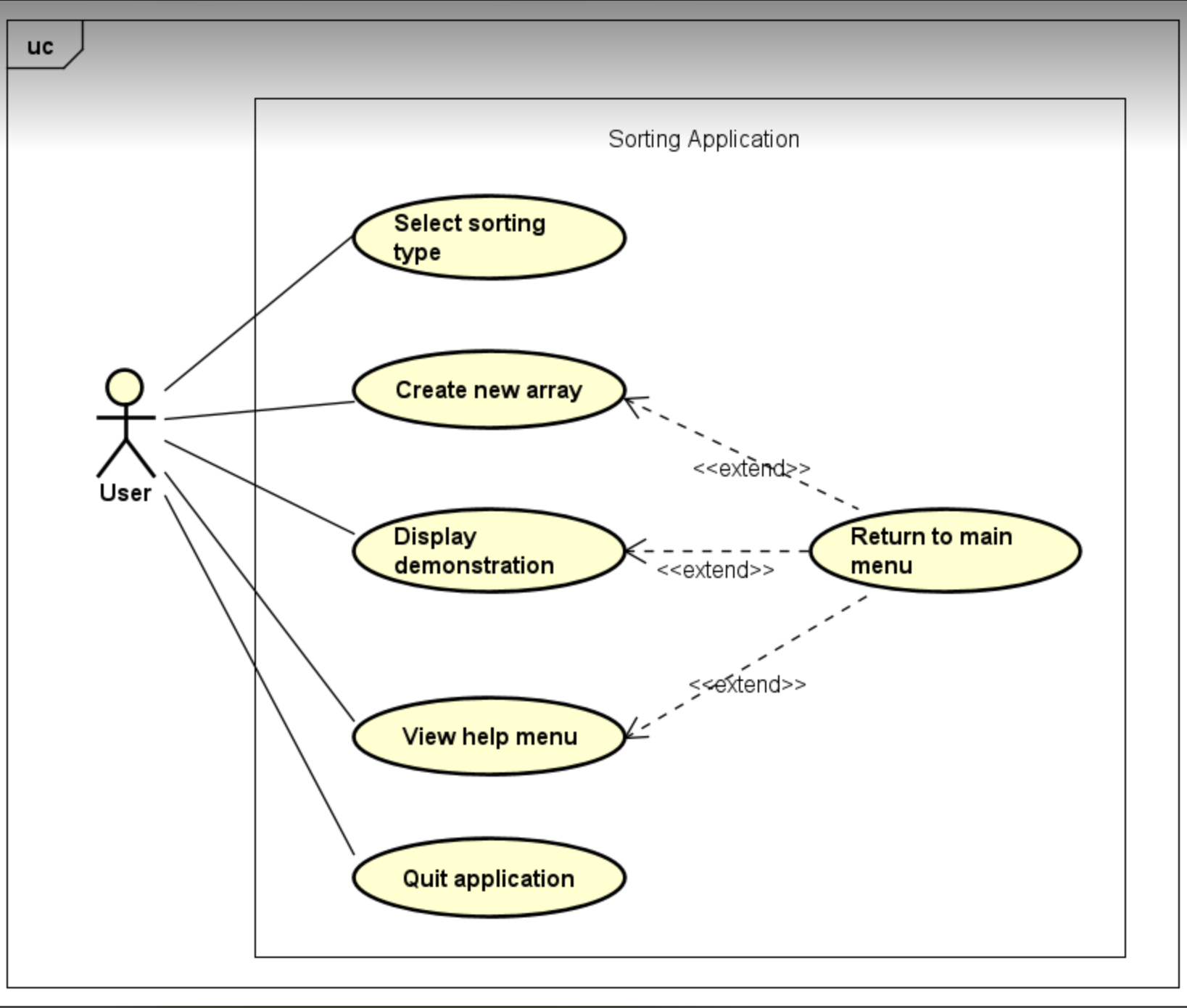
# PHÂN CÔNG NHIỆM VỤ

|  |  |  |
| --- | --- | --- |
| Họ và tên | MSSV | Nhiệm vụ |
| Phạm Thường Vi | 20226097 | Báo cáo + slide |
| Nguyễn Hoàng Việt | 20226098 | Code |
| Hoàng Quốc Việt | 20225684 | Báo cáo + slide |
| Đinh Đình Hải Việt | 20225683 | Code |
| Vũ Hoàng Việt | 20225777 | Code |

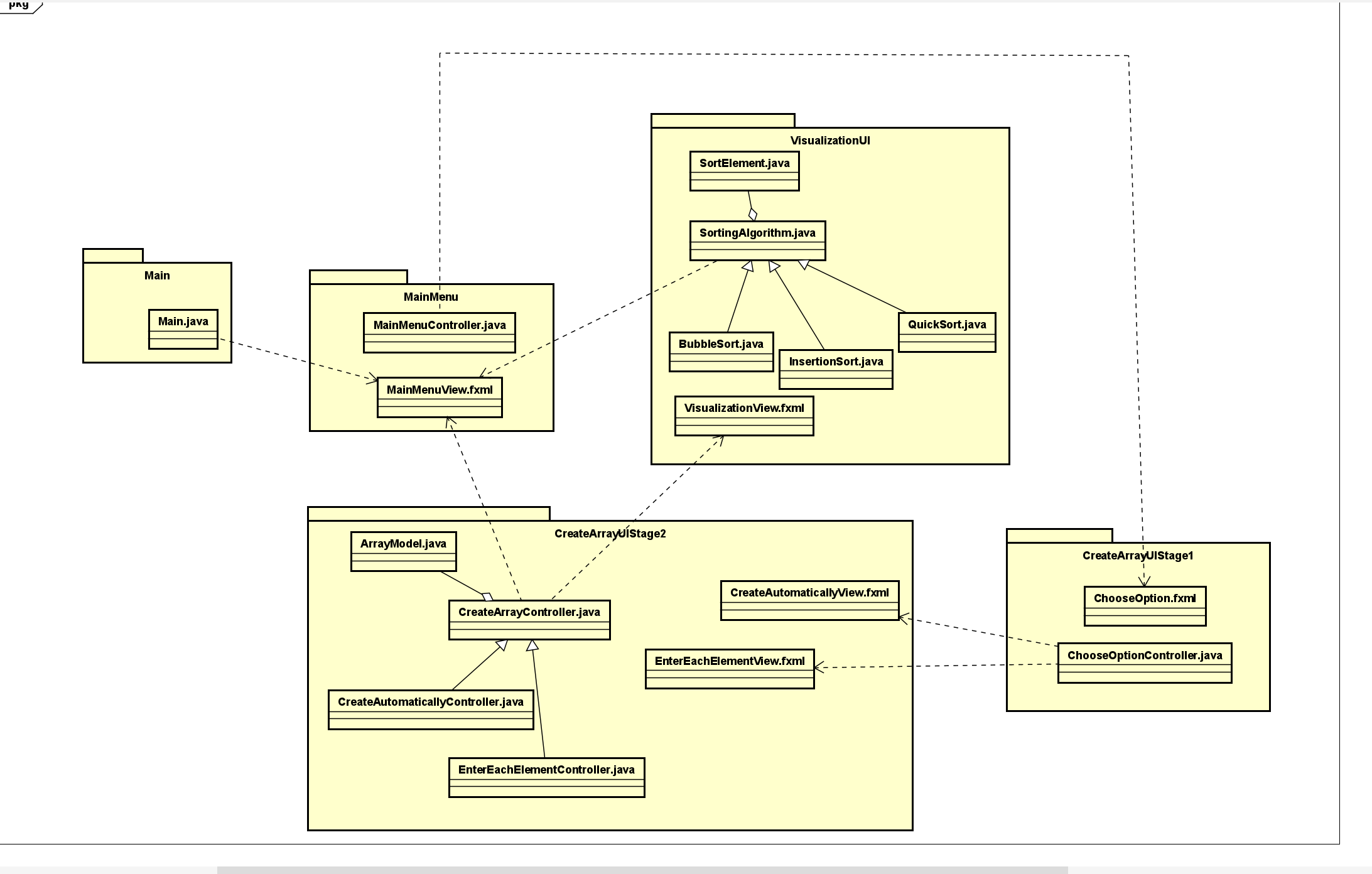
# MÔ TẢ BÀI TOÁN

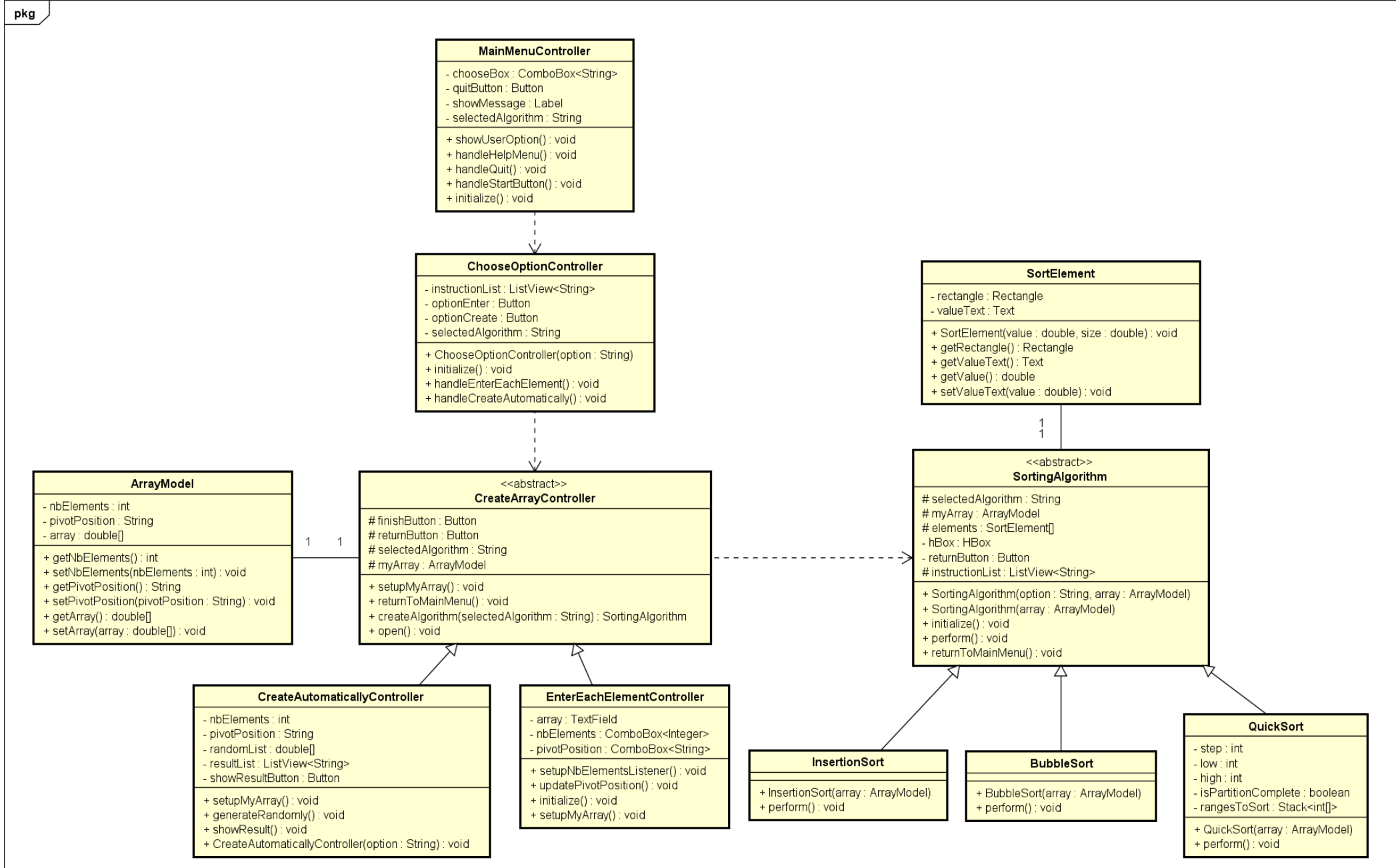
* Trong giao diện chính (main menu), người dùng có thực hiện những thao tác sau:
  + Chọn một trong ba thuật toán sắp xếp (Select sorting type): Quick sort, Bubble sort, Insertion sort.
  + Xem giao diện hỗ trợ (View help menu). Giao diện hỗ trợ chứa các thông tin về mục đích và cách sử dụng của ứng dụng.
  + Thoát khỏi ứng dụng (Quit application). Khi ấn nút thoát (Quit), hệ thống sẽ hiển thị thông báo xác nhận.
* 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 nhập số phần tử của mảng (*quy ước: là một số nguyên dương không lớn hơn 10*).
  + Người dùng lựa chọn nhập từng phần tử của mảng hoặc tạo mảng ngẫu nhiên. Mảng ngẫu nhiên được hệ thống tạo tự động theo số phần tử người dùng đã nhập.
  + Người dùng nhập vị trí phần tử trụ (chỉ dành cho thuật toán quick sort).
  + Người dùng ấn xác nhận. Trong trường hợp người dùng nhập mảng thủ công, nếu số phần tử chưa đủ, hệ thống sẽ báo lỗi.
  + Người dùng có thể quay lại main menu (Return to main menu) bất cứ lúc nào (hệ thống có thông báo xác nhận).
* Khi mảng đã được tạo thành công, người dùng có thể xem cách thuật toán hoạt động (Display demostration). Có 2 chức năng xem:
  + Xem từng bước (step by step).
  + Xem sau khi đã hoàn thành bước cuối cùng (final step).

Trong và sau quá trình thuật toán đang được trình diễn, người dùng có thể quay lại main menu (hệ thống có thông báo xác nhận).



# THIẾT KẾ





1. Phân tích kế thừa (Inheritance)

**Nhóm kế thừa : SortingAlgorithm**

* **Lớp cha:** SortingAlgorithm
* **Lớp con:** BubbleSort, InsertionSort, QuickSort
* **Ý nghĩa:**
  + Lớp SortingAlgorithm định nghĩa các thuộc tính và phương thức chung cho tất cả các thuật toán sắp xếp.
  + Các lớp con BubbleSort, InsertionSort, QuickSort kế thừa từ lớp cha và thực thi (override) phương thức perform(), displayStep(int stepIndex : int) để thực hiện logic riêng biệt cho từng thuật toán.

**Nhóm kế thừa : CreateArrayController**

* **Lớp cha:** CreateArrayController
* **Lớp con:** CreateAutomaticallyController, EnterEachElementController
* **Ý nghĩa:**
  + Lớp CreateArrayController đóng vai trò lớp cơ sở để xử lý việc tạo mảng (array) từ người dùng
  + Lớp con: kế thừa từ lớp cha và thực thi (override) phương thức setupMyArray()
    - CreateAutomaticallyController: Chịu trách nhiệm tạo mảng ngẫu nhiên.
      * Phương thức: generateRandom(), showResultButton().
    - EnterEachElementController: Chịu trách nhiệm cho phép người dùng nhập từng phần tử của mảng.
      * Phương thức: setupNbElementsListener(), updatePivotPosition().

2. Phân tích Association

**SortingAlgorithm và SortElement**

* **SortingAlgorithm** có một mảng các đối tượng **SortElement**.
* **Ý nghĩa:**
  + Lớp **SortingAlgorithm** **sử dụng** các đối tượng **SortElement** để biểu diễn các phần tử trong mảng cần sắp xếp.
  + Các phần tử **SortElement** chứa thông tin như giá trị và hình ảnh hiển thị, thuật toán sắp xếp (ở lớp SortingAlgorithm) sẽ thao tác trên các đối tượng này

**CreateArrayController và ArrayModel**

* **CreateArrayController** có một đối tượng **ArrayModel**.
* **Ý nghĩa:**
  + Lớp **CreateArrayController** **sử dụng** đối tượng **ArrayModel** để lưu trữ và quản lý mảng dữ liệu mà người dùng nhập.
  + **ArrayModel** chỉ là một đối tượng chứa các thông tin về mảng (số lượng phần tử, các giá trị mảng, v.v.) mà **CreateArrayController** sẽ thao tác trên đó.

3. Các mối liên kết giữa các lớp

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.

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.

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).

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).

4. Một số phương thức quan trọng

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

**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**.

**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**.

**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. Phương thức tạo mảng

**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**.

**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.