Báo cáo MiniProject

Học phần: Lập trình hướng đối tượng

Mã HP: IT3103

Mã lớp: 143577

Giảng viên: Nguyễn Thị Thu Trang

Trợ giảng: Lê Thanh Giang

Có sự tham khảo từ: https://chat.openai.com/

Toàn bộ sourcecode tại đây: github

Table of Contents

I. Assignment of members

Nguyễn Văn Nhâm - 20215105 (leader): UI, general class diagram

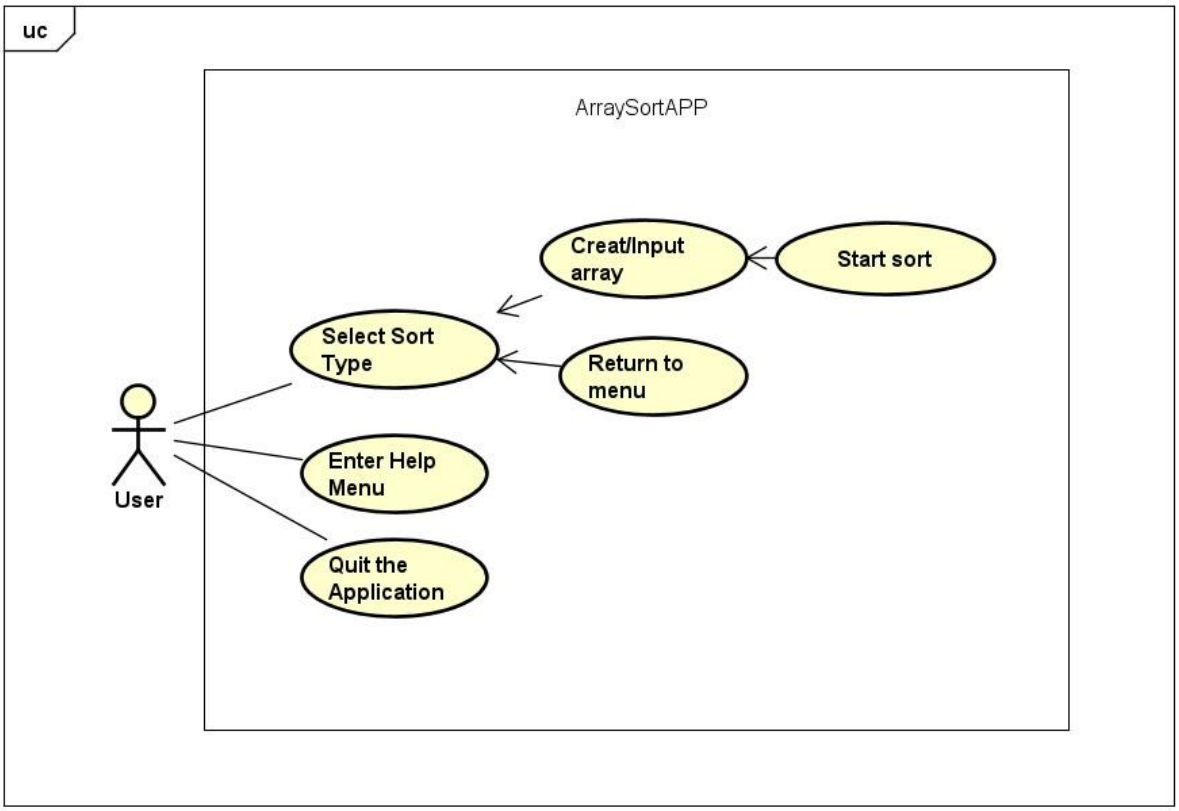
Ngọ Doãn Ngọc - 20215103 : UI, use case diagram

Dương Văn Nhất - 20215106 : Algorithm, class diagram, report

Lê Cao Phong - 20215113 : Algorithm, use case diagram, slide

II. Mini-project description

1. Mini-project requirement
   - Tạo một chương trình để giải thích ba thuật toán sắp xếp trên một mảng: merge sort, counting sort, and radix sort.
2. Use case diagram and explanation



- Trên menu chính: Tiêu đề của ứng dụng, 3 loại thuật toán sắp xếp cho người dùng chọn, menu trợ giúp, thoát .
- Người dùng phải chọn một loại sắp xếp để bắt đầu trình diễn.
- Menu Trợ giúp hiển thị cách sử dụng và mục tiêu cơ bản của chương trình.
- Tùy chọn thoát khỏi chương trình. Có yêu cầu xác nhận.

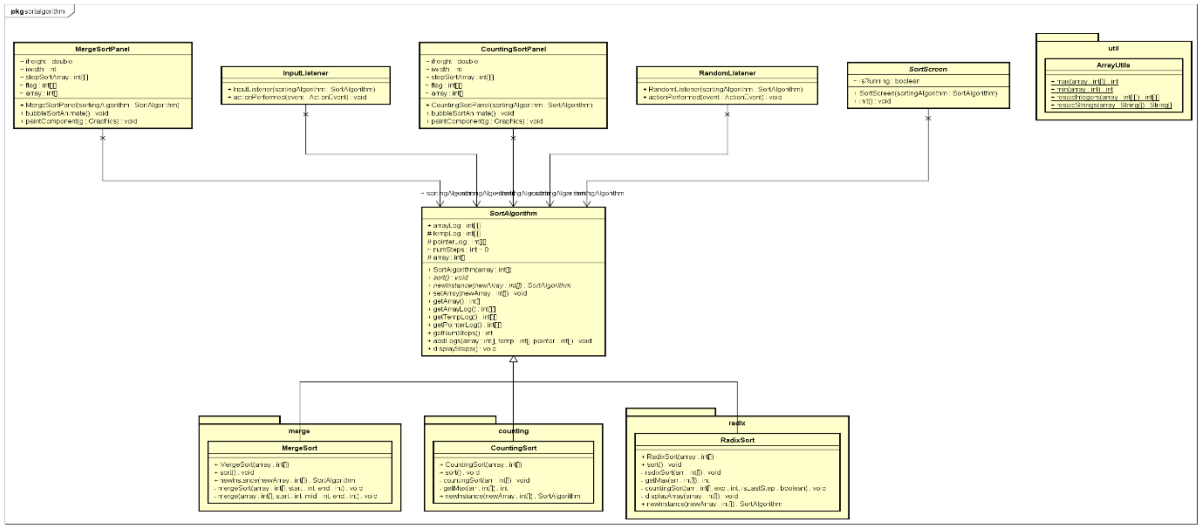
+, Khi chọn 1 loại sắp xếp

- Nút để tạo mảng: Người dùng có thể chọn ngẫu nhiên tạo một mảng hoặc nhập một mảng cho chương trình.

- Một nút để bắt đầu thuật toán với mảng được tạo.

- Nút quay lại để người dùng quay lại menu chính bất cứ lúc nào.

III. Design



1. **Lớp SortAlgorithm trong gói sortalgorithm**:
    - Lớp trừu tượng chứa một mảng và phương thức trừu tượng **sort**.
    - Phương thức **setArray:đặt mảng mới**
    - Phương thức **addLogs:** lưu trạng thái của mảng sau từng bước sắp xếp
    - Phương thức **newInstance**: Phương thức trừu tượng để tạo một thể hiện mới của **SortAlgorithm** với mảng mới.
2. **Lớp CountingSort trong gói sortalgorithm.counting**:
    - Phương thức **countingSort**: Triển khai thuật toán sắp xếp đếm.
    - Phương thức **getMax**: Trả về giá trị lớn nhất trong mảng
    - Phương thức **newInstance**: Tạo một thể hiện mới của **CountingSort** với mảng mới.
3. **Lớp MergeSort trong gói sortalgorithm.merge**:
    - Phương thức **mergeSort** và **merge**: Triển khai thuật toán sắp xếp trộn.
    - Phương thức **newInstance**: Tạo một thể hiện mới của **MergeSort** với mảng mới.
4. **Lớp RadixSort trong gói sortalgorithm.radix**:
    - Phương thức **radixSort** và **countingSort**: Triển khai thuật toán **RadixSort**.
    - Phương thức **newInstance**: Tạo một thể hiện mới của **RadixSort** với mảng mới.
5. **Lớp ArrayUtils:**

    Chứa các tiện tích xử lí mạng như tăng length, tìm max;
6. **Lớp MainMenu:**

    Phương thức **init:** tạo Jframe chứa các các button tạo thành 1 menu
7. **Các lớp listener:**

Xử lý sự kiện khi click button

8. **Lớp SortScreen:**

   Lớp trừu tượng chứa các button input, random, back

9. **Lớp MergeSortScreen,CountingSortScreen,RadixSortScreen:**

   Gọi 1 đối tượng JPanel để mô phỏng thuật toán

10. **Lớp MergeSortPanel,CountingSortPanel:**

    Phương thức **SortAnimate** và **PaintComponet**: vẽ mô phỏng thuật toán sắp xếp

11. **Lớp Main**

    - Phương thức **main**: Chạy ứng dụng bằng cách tạo một thể hiện của **MainMenu** và hiển thị menu chính.

Giải thích, mô tả mối quan hệ giữa các lớp và cài đặt của một số phương pháp quan trọng:

Các Lớp:

1. **SortAlgorithm:**
   - Lớp cơ sở cho các thuật toán sắp xếp.
   - Chứa một mảng cần được sắp xếp và có phương thức trừu tượng **sort()**.
2. **MergeSort:**
   - Kế thừa từ **SortAlgorithm**.
   - Triển khai thuật toán Merge Sort.
3. **CountingSort:**
   - Kế thừa từ **SortAlgorithm**.
   - Triển khai thuật toán Counting Sort.
4. **RadixSort:**
   - Kế thừa từ **SortAlgorithm**.
   - Triển khai thuật toán Radix Sort.
5. **MainMenu:**
   - Hiển thị menu chính cho người dùng để chọn thuật toán sắp xếp.
   - Tạo các button và gọi đến các lớp listener để xử lý sự kiện khi click button
6. **Các lớp listener:**

   Xử lý sự kiện khi click button:

   BackListener: trở về MainMenu

   ExitListener: thoát khỏi chương trình

HelpListener:hiện thông tin của ứng dụng

RandomListener: tạo random một array

InputListener: nhập mảng từ người dùng

Các lớp SortListener: tạo các đối tượng của view và algorithm để hiện thị dữ liệu

7. **Lớp SortScreen:**

   Lớp trừu tượng chứa các button input, random, back

8. **Lớp MergeSortScreen,CountingSortScreen,RadixSortScreen:**

   Gọi 1 đối tượng JPanel để mô phỏng thuật toán

9. **Lớp MergeSortPanel,CountingSortPanel:**

   Phương thức **SortAnimate** và **PaintComponet**: vẽ mô phỏng thuật toán sắp xếp

10. **Main:**

    - Chứa hàm **main()** để bắt đầu chạy ứng dụng.

Mối Quan Hệ:

- SortAlgorithm, MergeSort, CountingSort, RadixSort: Các lớp này kế thừa từ SortAlgorithm, chia sẻ chung giao diện và cài đặt cơ bản để sắp xếp mảng.
- SortingDemoInterface: Sử dụng một thể hiện của SortAlgorithm để tương tác với người dùng, tạo mảng, sắp xếp và hiển thị kết quả.
- MainMenu: Hiển thị menu và tạo các thể hiện của các thuật toán sắp xếp dựa trên lựa chọn của người dùng.

- Mỗi lớp sắp xếp có một phương thức **newInstance()** để tạo một thể hiện mới của chính nó với một mảng mới. Điều này giúp tái sử dụng và cập nhật mảng một cách linh hoạt.
- Mỗi lớp sắp xếp có thể hiển thị kết quả sau mỗi bước của quá trình sắp xếp, giúp người dùng theo dõi quá trình.
- Giao diện người dùng được thiết kế để cho phép người dùng lựa chọn cách tạo mảng (ngẫu nhiên hoặc từ người dùng) và thực hiện sắp xếp.