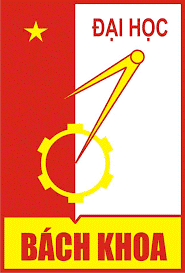
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**



Mini-projects

**Object-Oriented Language and Theory**

**GVHD: Nguyễn Thị Thu Trang**

**Nhóm: 07**

**Thành Viên: Nguyễn Khắc Thắng - 20176869**

**Võ Quang Thành Đạt - 20176714**

**Mã HP: IT3102**

**Mã Lớp: 113833**

Mục Lục

[I. Mini-project description 2](#_Toc42707913)

[*1.* *Mô tả chi tiết về mini-project* 2](#_Toc42707914)

[*2.* *Use-case diagram* 2](#_Toc42707915)

[II. Explanation of the design ideas 3](#_Toc42707916)

[*1.* *Class diagram* 3](#_Toc42707917)

[*2.* *Giải thích chi tiết* 3](#_Toc42707918)

[- Package: 3](#_Toc42707919)

[- Chi tiết các class chính: 3](#_Toc42707920)

[- Phân tích quan hệ giữa các class: 4](#_Toc42707921)

[III. Assignment of members 5](#_Toc42707922)

[*1.* *Nguyễn Khắc Thắng – 20176869* 5](#_Toc42707923)

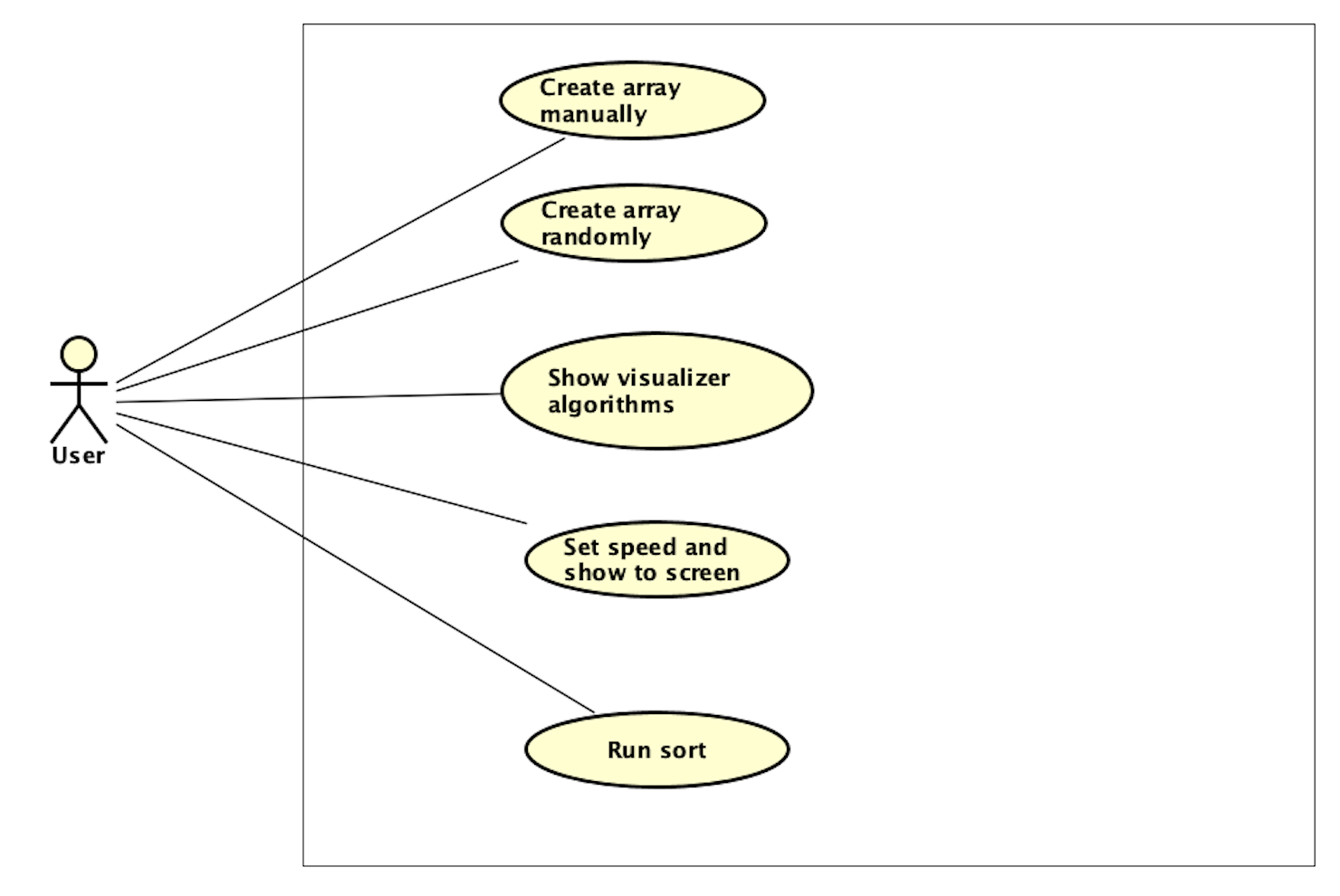
[*2.* *Võ Quang Thành Đạt – 20176714* 5](#_Toc42707924)

[IV. Lanch Application 6](#_Toc42707925)

[V. Summary 7](#_Toc42707926)

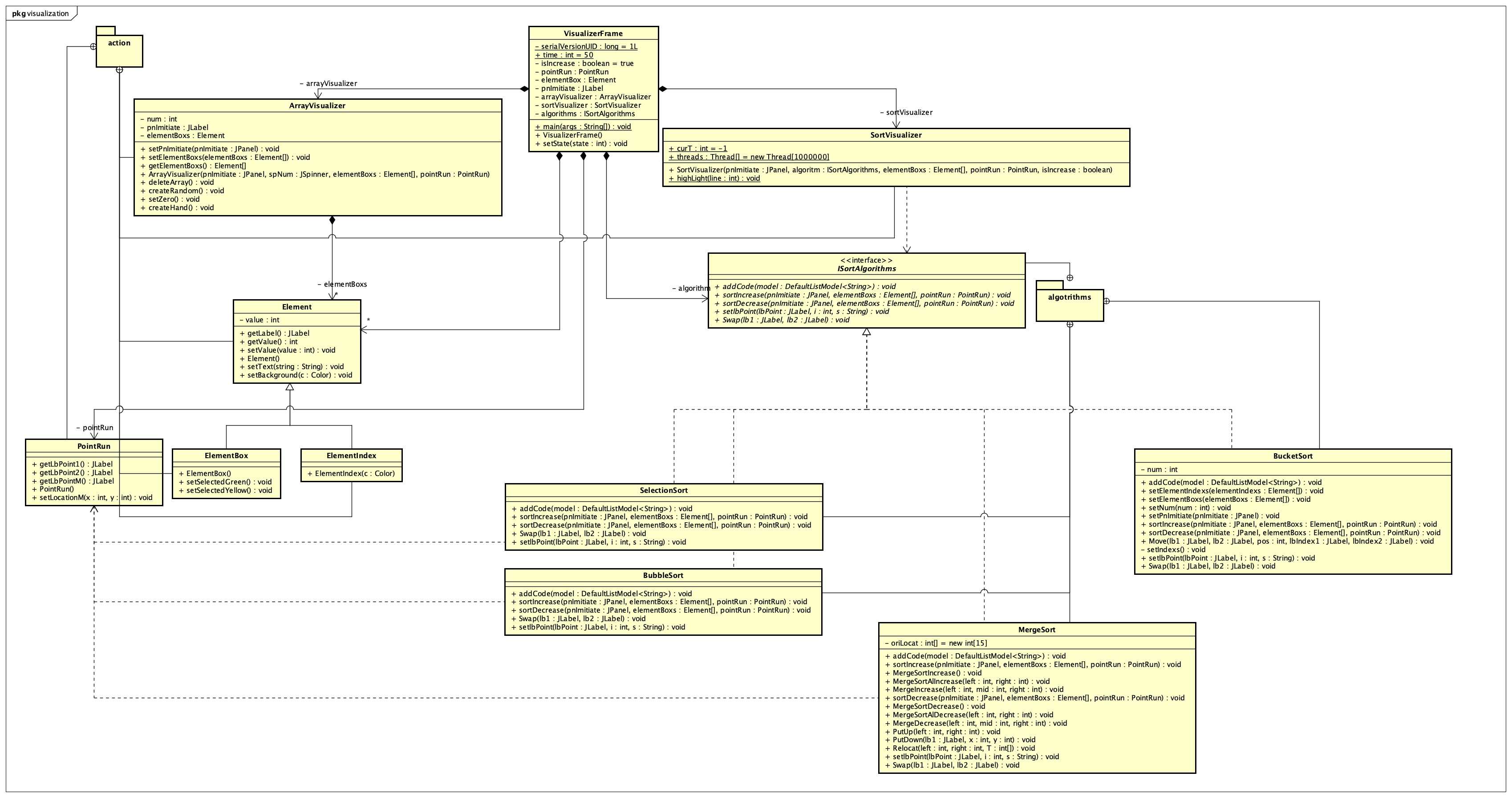
1. **Mini-project description**
   1. *Mô tả chi tiết về mini-project*

* Đề tài: 2.1.10 - An application to (visually) demonstrate sorting algorithms:
  + Bubble sort
  + Selection sort
  + Merge sort
  + Bucket sort
* Mục tiêu:
  + Tạo ra một ứng dụng trình diễn về các thuật toán ở trên
  + Cách phân tích yêu cầu đề bài, thiết kế ứng dụng thông qua việc thiết kế use-case diagram và class diagram
  + Củng cố và vận dụng kiến thức lập trình hướng đối tượng vào bài toán cụ thể
  + Vận dụng Swing để thiết kế GUI trong Java
  1. *Use-case diagram*
* Bản vẽ trên Astah UML:



* Giải thích:
  + Create array manually: Người dùng tạo một mảng bằng cách chọn số lượng mảng và nhập từng phần tử trong mảng
  + Create array randomly: Người dùng tạo một mảng bằng cạch chọn số lượng mảng và các phần tử được sinh ngẫu nhiên
  + Show visualizer algorithms: Người dùng sau khi lựa chọn một thuật toán:
    - Phần mã giả sẽ được hiện thị
    - Thuật toán đang lựa chọn sẽ được khởi tạo chuẩn bị cho công việc Run sort
  + Set speed and show to screen: Điều chỉnh tốc độ và trả về cho người dùng kết quả tốc độ được thay đổi trong quá trình hiển thị sắp xếp
  + Run sort: Bắt đầu việc sắp xếp:
    - Lấy mảng đã được khởi tạo
    - Lấy thuật toán đã được khởi tạo
    - Lấy tốc độ đã được cài đặt
    - Hiển thị kết quả quá trình sắp xếp cho người dùng

1. **Explanation of the design ideas**
   1. *Class diagram*



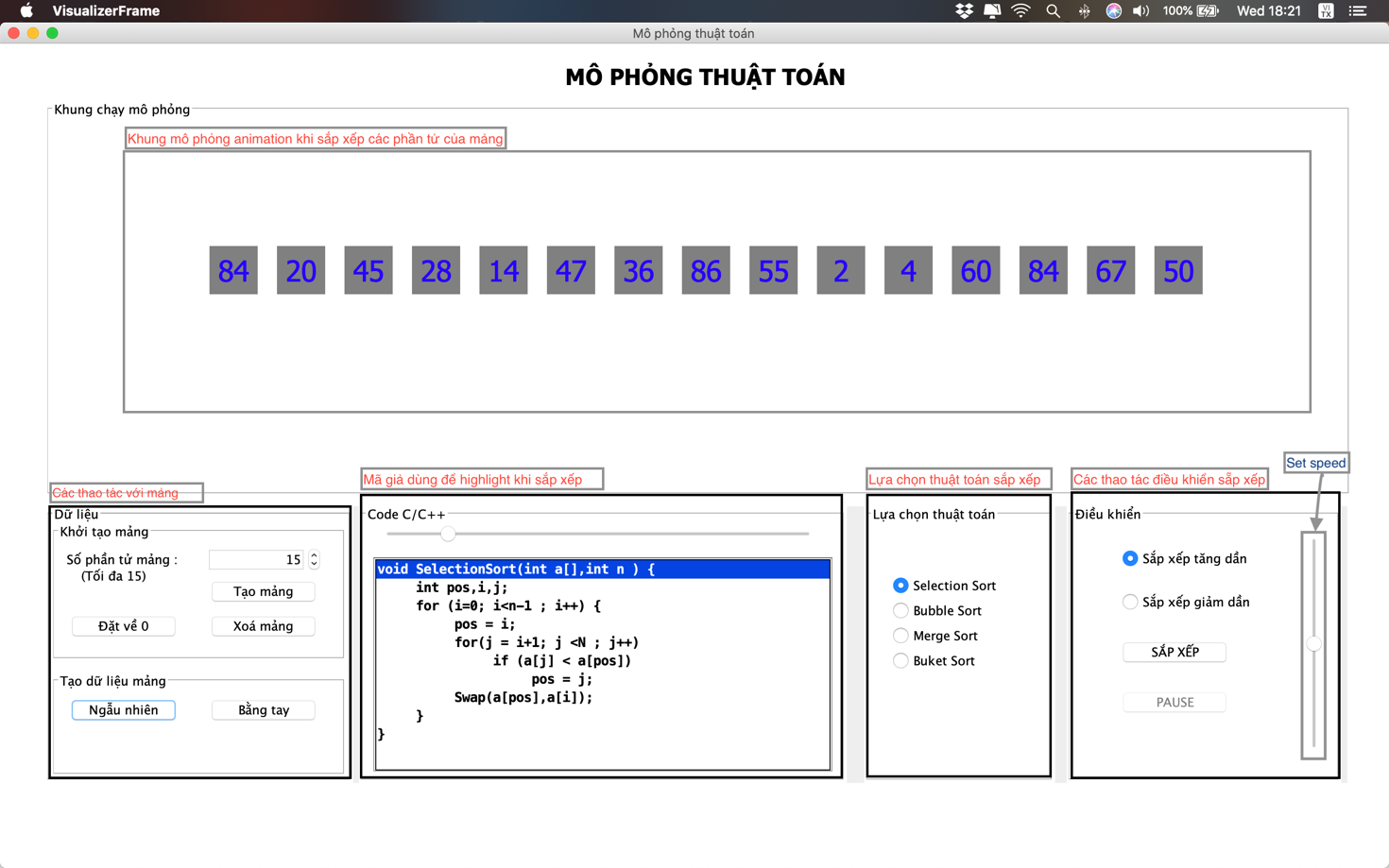
* 1. *Giải thích chi tiết*
* Package:
  + Visualization: Chứa toàn bộ các thành phần giao diện chính của ứng dụng
  + Visualization.action: Thực hiện các thao tác với mảng
  + Visualization.algorithms: Chứa các thuật toán sắp xếp
* Chi tiết các class chính:
  + Class VisualizerFrame:
    - Tạo ra giao diện chính của ứng dụng
    - Chứa các thao tác xử lý action của các button, scroll,…
  + Class ArrayVisualizer:
    - Tạo ra một mảng các Element (toàn phần tử 0)
    - Tạo ngẫu nhiên hoặc nhập tay giá trị các phần tử trong mảng
    - Reset giá trị mảng về 0 và xoá mảng
  + Class SortVisualizer:
    - Lấy thuật toán sắp xếp đang được lựa chọn thông qua Interface IsortAlgorithms
    - Lấy tốc độ đang được cài đặt
    - Lấy chiều sắp xếp (tăng dần / giảm dần)
    - Tiến hành gọi phương thức sắp xếp
  + Class Element: Đối tượng được khởi tạo ra sẽ là một phần tử của mảng
  + Class PointRun:
    - Biểu thị vị trí các phần tử đang được duyệt
    - Ví dụ: i, j, min, max,…
  + Class IsortAlgorithms:
    - Đây là một interface, các thuật toán sắp xếp được gọi thông qua nó
    - Class SelectionSort, BubbleSort, MergeSort, BucketSort implements từ interface này
* Phân tích quan hệ giữa các class:
  + Composition:
    - VisualizerFrame với ArrayVisualizer, SortVisualizer, Element, PointRun, IsortAlgorithms
      * Trong class VisualizerFrame, các thuộc tính là đối tượng được khởi tạo từ các class kia
      * Khi VisualizerFrame bị huỷ thì đối tượng của class kia cũng sẽ huỷ theo (do chương trình chỉ có một khung duy nhất tạo ra trong quá trình thực hiện)
    - ArrayVisualizer với Element
      * Trong class ArrayVisualizer, thuộc tính elementBoxs được khởi tạo là một mảng các đối tượng từ class Element
      * Khi ArrayVisualizer bị huỷ thì mảng các đối tượng từ class Element cũng bị huỷ theo
  + Dependency:
    - SelectionSort, BubbleSort, MergeSort với PointRun
      * Các thuật toán sắp xếp sử dụng đến class PointRun dưới dạng khởi tạo một đối tượng local (chỉ dùng trong method)
      * Khi PointRun thay đổi (ví dụ i, j,… thay đổi) thì hiển thị khi chạy thuật toán cũng thay đổi theo
    - SortVisualizer với ISortAlgorithms: Khi đối tượng khởi tạo của ISortAlgorithms thay đổi (down casting xuống các thuật toán khác nhau) thì SortVisualizer cũng bị thay đổi theo
  + Realization:
    - SelectionSort, BubbleSort, MergeSort, BucketSort đều implements từ interface ISortAlgorithms
    - Các thao tác với một thuật toán cụ thể đều được gọi, sử dụng thông qua interface
    - Ứng dụng tính chất đa hình trong việc gọi các method của thuật toán đang được lựa chọn
  + Generalization:
    - ElementBox, ElementIndex kế thừa từ Element
    - ElementBox và ElementIndex có thuộc tính và phương thức khá tương đồng nhau 🡪 khái quát hoá bởi class cha Element

1. **Assignment of members**
   1. *Nguyễn Khắc Thắng – 20176869*

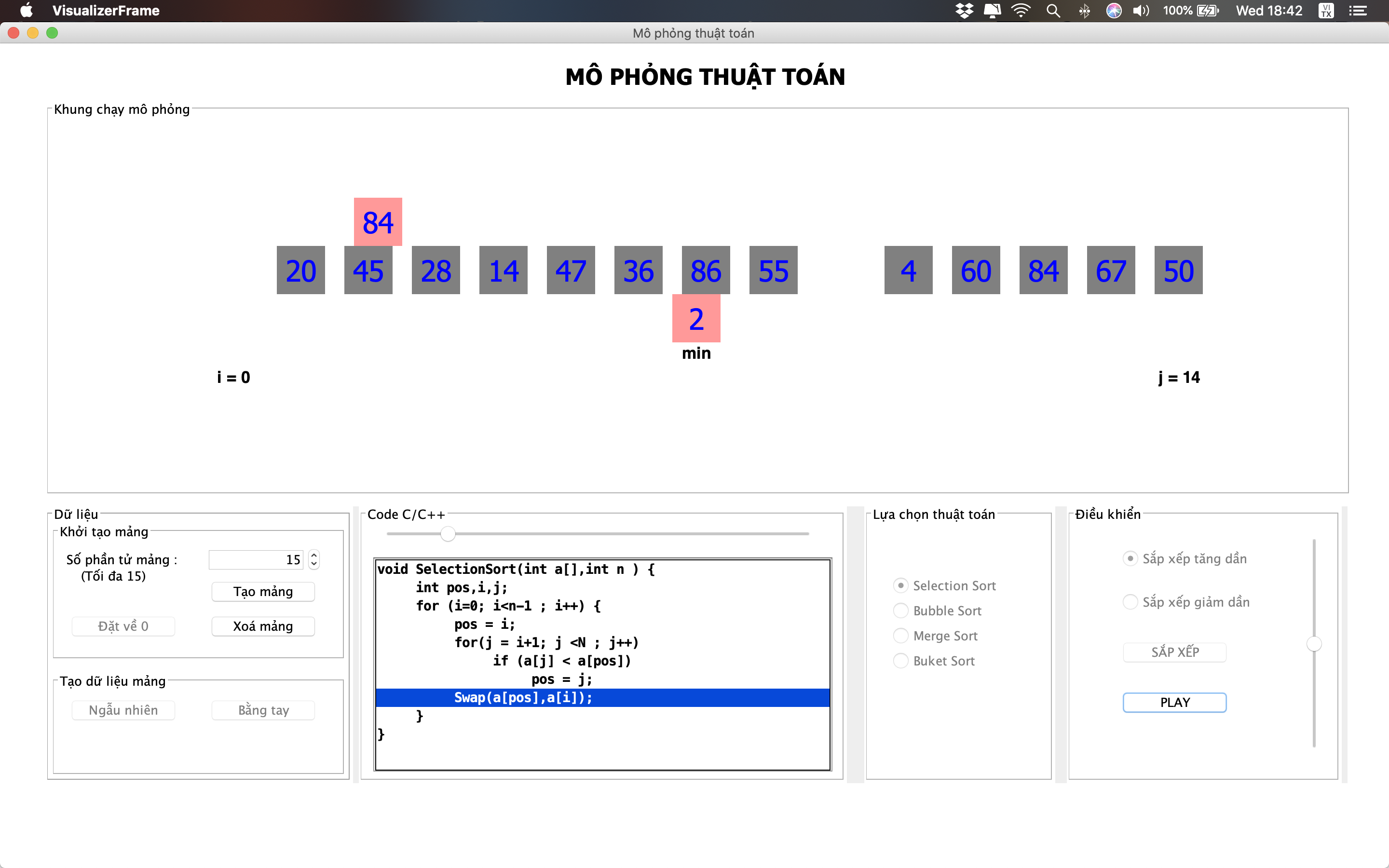
* Thiết kế class:
  + Chia pakage kèm theo chức năng chính của nó
  + Các class chính kèm theo chức năng cụ thể của nó
  + Áp dụng tính chất của OOP vào thiết kế class
* Class ArrayVisualizer
* Class SortVisualizer
* Class Element, ElementBox, ElementIndex
* Interface ISortAlgorithms
* Class SelectionSort
* Class MergeSort: Phần sortIncrease
* Class BucketSort: Phần sortIncrease
  1. *Võ Quang Thành Đạt – 20176714*
* Thiết kế use-case diagram
* Class VisualizerFrame:
  + Thông qua một vài cách design GUI từ một số open-source-code trên github
  + Chọn lọc và vẽ ra bản thiết kế giao diện tổng quát cho ứng dụng
* Class BubbleSort:
  + Override các phương thức từ Interface ISortAlgorithms
  + Cài đặt PointRun (i, j chạy ở dưới)
* Class MergeSort: Phần sortDecrease
* Class BucketSort: Phần sortDecrease

1. **Lanch Application**

* Khởi tạo:



* Khi tạm dừng một thuật toán đang sắp xếp (ví dụ: SelectionSort)



1. **Summary**

* Kiến thức thu được sau khi làm project:
  + Phân tích yêu cầu của một bài toán cụ thể
  + Thiết kế use-case diagram và class diagram
  + Sử dụng Swing tạo một ứng dụng GUI trong java
  + Ôn tập tính chất của OOP và vận dụng vào thiết kế class
  + Làm quen với xử lý luồng để tạo animation khi sắp xếp
* Kỹ năng thu được:
  + Kỹ năng làm việc nhóm
  + Kỹ năng sử dụng git quản lý project
  + Kỹ năng phân tích vấn đề và tổng hợp viết báo cáo