HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

──────── \* ───────

****

**Object-Oriented Language and Theory**

**PROJECT: Demonstration of Sorting Algorithms on Array**

**Instructor: Dr. Nguyễn Thị Thu Trang**

**Group members:**

Nguyễn Văn Việt - 20194883

Đỗ Văn Tuấn - 20194874

Vũ Anh Tuấn - 20194878

***Hà Nội, 2022***

**Contents**

**Contents2**

**Project Overview 3**

**Assignment task for each member4**

**CHAPTER 1: Mini-Project Description5**

1. Describe in detail mini-project requirement5
2. Usecase diagram and explanation5

**CHAPTER 2: Design7**

1. General Class Diagram7
2. Class Diagram for each Package7
   1. Class diagram for screen 8
   2. Class diagram for entity9
   3. Class diagram for algorithms11
3. Explanation of the design 15

**PROJECT OVERVIEW**

The mission of our project is to build an application that visualizes three sorting algorithms: Merge Sort, Counting Sort, Radix Sort

Due to the main purpose of visualization is to help the user get a better insight about how an algorithm work, we have put some restrictions in our application:

* Only non-negative(>0) numbers are allowed to be an array’s element.
* The array’s size used for visualization has to be less than 15

**ASSIGNMENT FOR EACH MEMBER**

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Email** | **Task** | **Evaluation** |
| Nguyễn  Văn Việt | viet.nv194883@sis.hust.edu.vn | ElementBox.java  ElementBoxs.java  InitialElementBoxs.java  Instruction.java  PointRun.java  MergeSort.java | Work is complete in time |
| Đỗ  Văn  Tuấn | tuan.dv194874@sis.hust.edu.vn | CountingSort.java  RadixSort.java  Sort.java  SortFrame | Work is complete in time |
| Vũ  Anh Tuấn | tuan.va194878@sis.hust.edu.vn | HelpFrame  MainApplication  MainFrame  SortFrame | Work is complete in time |

**CHAPTER 1: Mini-project Description**

**1.Describe in detail about mini project requirement**

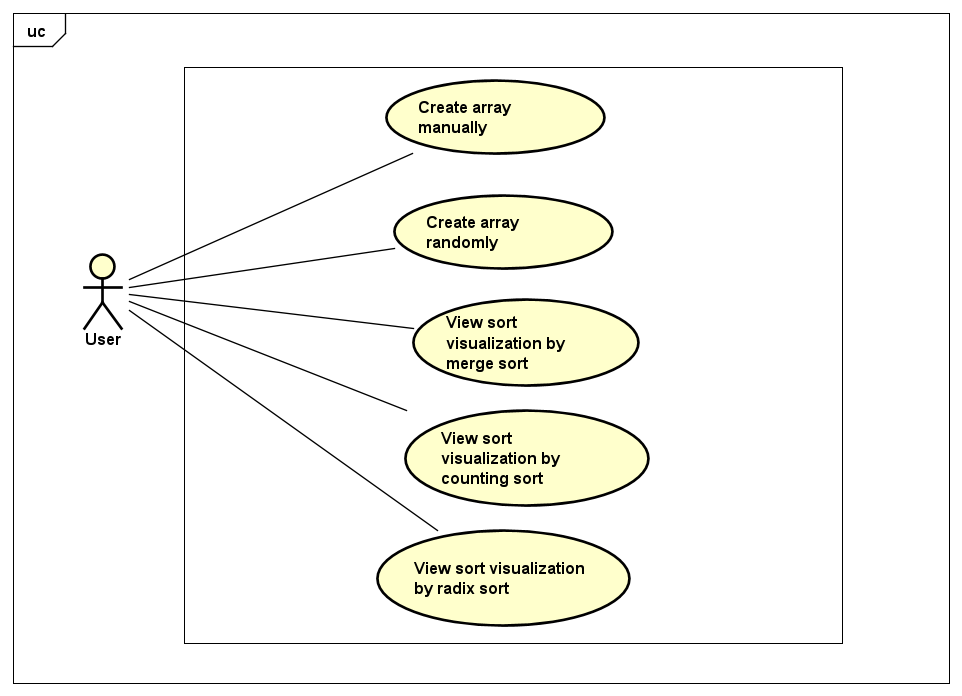
On the main menu: title of the application, 3 types of sort algorithms for a user to choose, helping, quit:

* User must select a sort-type in order to start the demonstration.
* The help menu shows the basic usage and aim of the program.
* Quit exiting the program. The application asks for confirmation before closing.

In the demonstration:

* A button for creating the array: The user can choose to randomly create an array or input an array for the program, and sort ascending or descending.
* A button for starting the algorithm with the created array. Remember to clearly show each step of the sorting.
* Pause and Slider button: users can pause the visualization or change the speed of the visualization
* A back button for users to return to the main menu at any time

**2.Usecase diagram and explanation: How the users interact with the software with use cases**

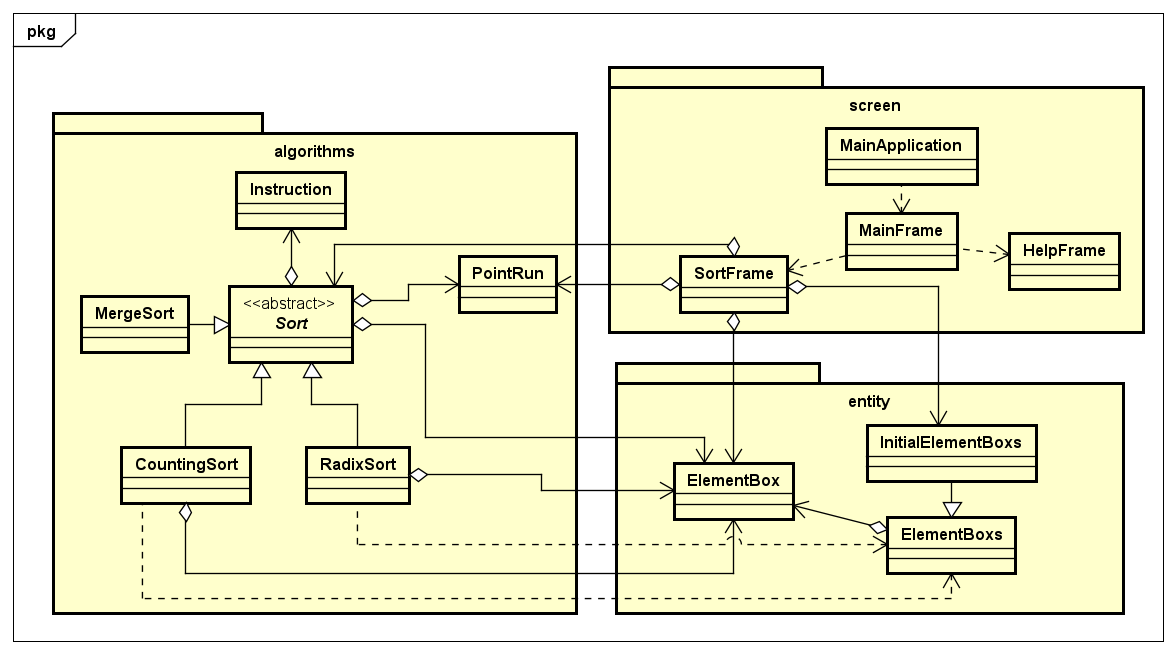


Based on all the requirements we decided to develop five use cases (as shown in the figure for our application.

1. Take all user commands from the GUI to create an array such as: whether customizing or randomizing array, which size of the array to create.
2. Users can view Merge Sort, Counting Sort, and Radix Sort Visualization by clicking on the button with the same name.

**CHAPTER 2: DESIGN**

**1. General class diagram**

****

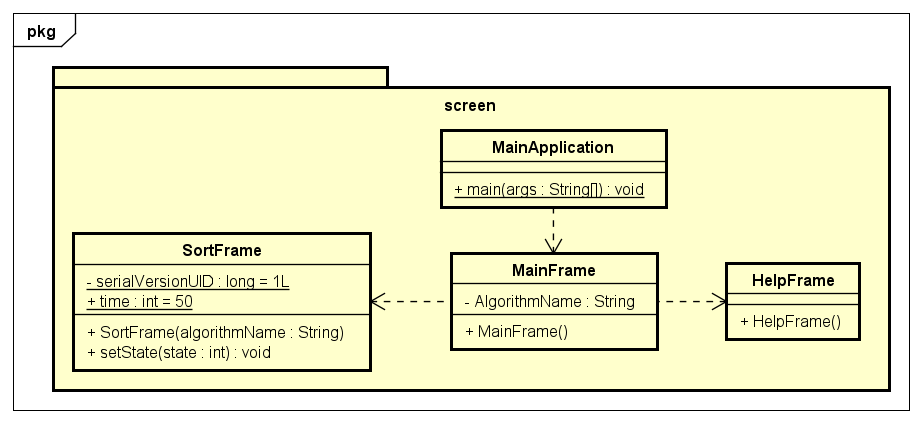
* Package “algorithms”: store the main data structure and 3 sorting algorithms for visualizing used throughout this project
* Package “screen”: store the main screen class of the application
* Package “entity”: store the class “ElementBox” which is used for visualization in the GUI.

**2. Class diagrams for each package**

In our project, we divided into 3 packages with several classes inside them.

Below are class diagrams for each package.

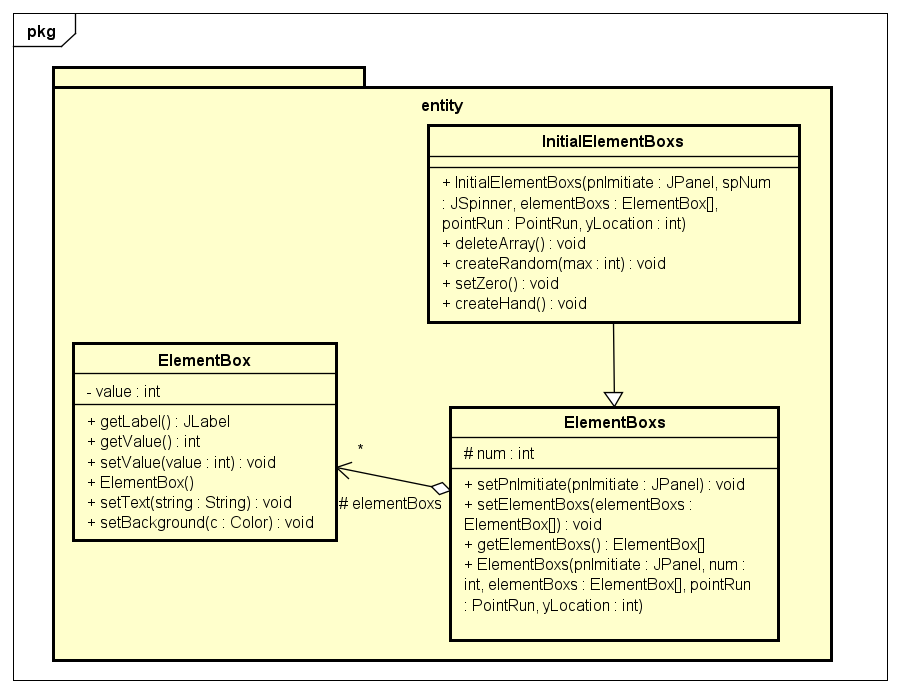
**2.1 Class diagrams for package screen**

****

We create three frames, which all extend JFrame, corresponding to three windows of our application:

* The MainFrame class is used to display the main menu window, where the user can choose a sort algorithm to visualize or view help.
* The SortFrame class is used to display the sort visualization window, where the user can view the visualization of a sort algorithm and control it by buttons (Pause, Slider). Refer to some sources on github.
* The HelpFrame class is used to display the help window, which provides instructions for users to use our application properly.

**2.2 Class diagrams for package entity**

****

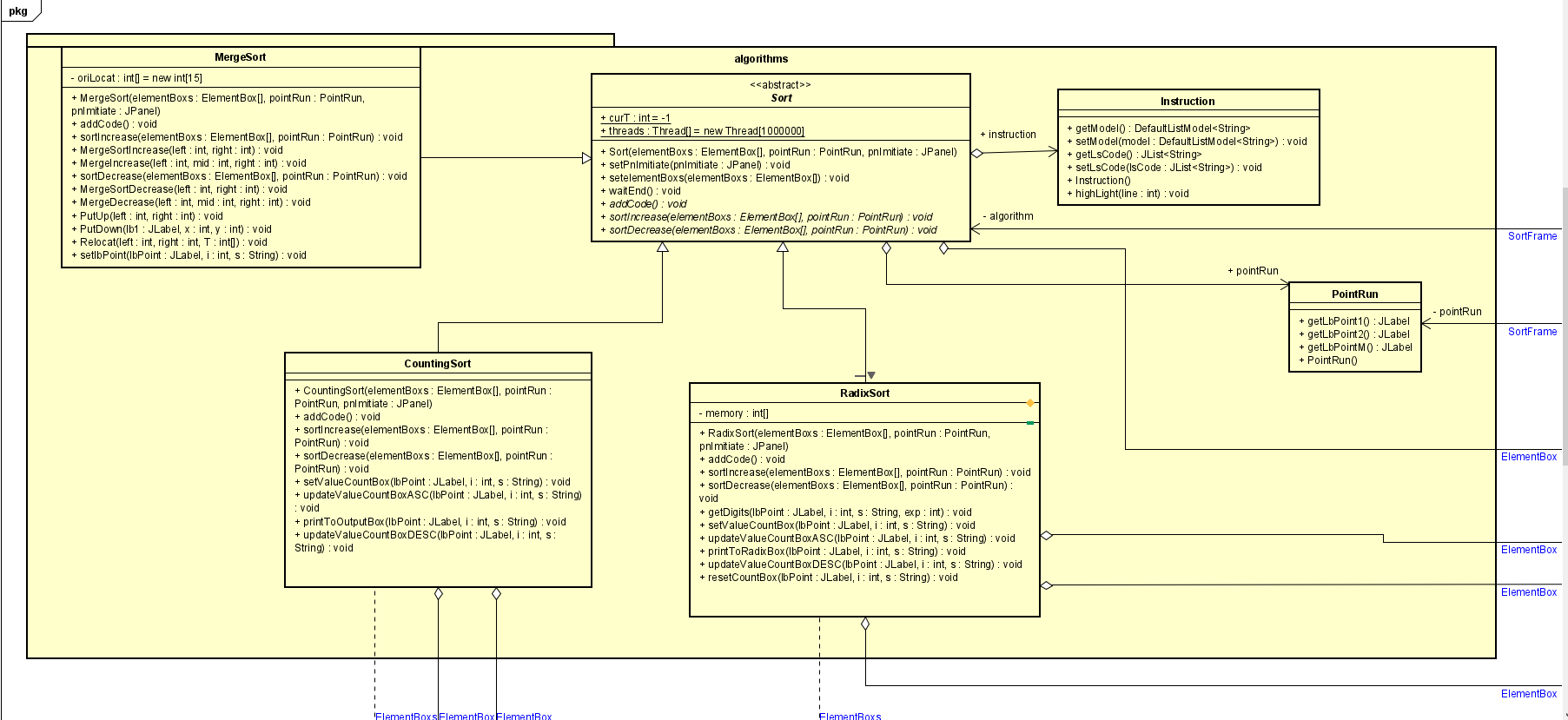
* The ElementBox class is used to create a square JLabel having a value for each member in the array
* The ElementBoxs class is used to create an array of ElementBox having x and y location in PnImitiate
* The InitialElementBoxs class inherits from ElementBoxs class and is used to input array from the user.

|  |  |
| --- | --- |
| Class ElementBox | |
| Attribute & Method | Explanation |
| value | Store the value of the ElementBox |
| getLabel | Get JLabel of the ElementBox |
| getValue | Get value of the ElementBox |
| setValue | Set value of the ElementBox |
| ElementBox | Constructor of ElementBox |
| setText | Set the text of the JLabel of the ElementBox |
| setBackground | Set background of the JLabel of the ElementBox |

|  |  |
| --- | --- |
| Class ElementBoxs | |
| Attribute & Method | Explanation |
| num | Store the number of ElementBox |
| setPnImitiate | Set the value of pnImitiate |
| setElementBoxs | Set the value of ElementBoxs |
| getElementBoxs | Get the value of ElementBoxs |
| ElementBoxs | Constructor of ElementBoxs |

|  |  |
| --- | --- |
| Class InitialElementBoxs | |
| Attribute & Method | Explanation |
| initialElementBoxs | Constructor of InitialElementBoxs |
| deleteArray | Delete all the ElementBoxs |
| createRandom | Create random values of ElementBoxs |
| createHand | Input each value of ElementBoxs manually |
| setZero | Set all values of ElementBoxs to 0 |

**2.3 Class diagrams for package algorithms**



* Class Sort is used to used to declare attributes and methods for use in 3 algorithm simulation classes.
* Class PointRun is used to generate running points below the labels during algorithm simulation
* Class Instruction is used to store, display and highlight the pseudo code of the algorithms
* Class MergeSort inherits from Sort class and is used to perform merge sort algorithm simulation. We inherit GUI idea from a public github repo.
* Class CountingSort inherits from Sort class and is used to perform counting sort algorithm simulation. Refer to <https://www.youtube.com/watch?v=7zuGmKfUt7s&ab_channel=GeeksforGeeks>
* Class RadixSort inherits from Sort class and is used to perform radix sort algorithm simulation. Refer to

<https://www.youtube.com/watch?v=nu4gDuFabIM&ab_channel=GeeksforGeeks>

|  |  |
| --- | --- |
| Abstract Class Sort | |
| Attribute & Method | Explanation |
| curT | Save the number of threads used |
| threads | Simulation requires multiple threads |
| setPnImitiate | Set the value of PnImitiate |
| setelementBoxs | Set the value of elementBoxs |
| Sort | Constructor of Sort |
| addCode | Add source code of algorithm into Pseudo code frame to display on screen. |
| sortIncrease | Do ascending sort . |
| sortDecrease | Do descending sort. |
| waitEnd | Announce the simulation process is complete |

|  |  |
| --- | --- |
| Class PointRun | |
| Attribute & Method | Explanation |
| getLbPoint1 | Get the value of lbPoint1 |
| getLbPoint2 | Get the value of lbPoint2 |
| getLbPointM | Get the value of lbPointM |
| PointRun | Constructor of PointRun |

|  |  |
| --- | --- |
| Class Instruction | |
| Attribute & Method | Explanation |
| getModel | Get the value of model |
| setModel | Set the value of model |
| getLsCode | Get the value of lsCode |
| setLsCode | Set the value of lsCode |
| Instruction | Constructor of Instruction |
| highlight | Highlight the line in lsCode |

|  |  |
| --- | --- |
| Class MergeSort | |
| Attribute & Method | Explanation |
| orilocat[] | Store the x coordinate of ElementBox |
| addCode | Add source code of algorithm into Pseudo code frame to display on screen. |
| sortIncrease | Set up some variables for visualization and call MergeSortIncrease to start ascending sort. |
| sortDecrease | Set up some variables for visualization and call MergeSortDecrease to start descending sort. |
| MergeSortIncrease | Start ascending sort. |
| MergeIncrease | After sort two arrays, put it again into one array |
| MergeSortDecrease | Start descending sort. |
| MergeDecrease | After sort two arrays, put it again into one array |
| PutUp | Put the ElementBox up and highlight it to swap during visualization |
| PutDown | Put the ElementBox down after putting it up during visualization |
| Relocat | Relocation of the ElementBox after MergeIncrease or MergeDecrease. |
| setlbPoint | Set the value and location of PointRun during visualization |

|  |  |
| --- | --- |
| Class CountingSort | |
| Attribute & Method | Explanation |
| CountingSort | Constructor of CountingSort |
| addCode | Add source code of algorithm into Pseudo code frame to display on screen. |
| sortIncrease | Do ascending sort. |
| sortDecrease | Do descending sort. |
| setValueCountBox | store count of each character into count array(countBoxs). |
| updateValueCountBoxASC | Change value of count array(countBoxs), now countBoxs[i] contains actual position of " i " character in output array (using in sort ascending). |
| updateValueCountBoxDESC | Change value of count array(countBoxs), now countBoxs[i] contains actual position of " i " character in output array (using in sort descending). |
| printToOutputBox | Put the characters in order into the output array(outputBoxs) and display on the screen. |

|  |  |
| --- | --- |
| Class RadixSort | |
| Attribute & Method | Explanation |
| memory[] | store value of original numbers through each loop. |
| RadixSort | Constructor of RadixSort |
| addCode | Add source code of algorithm into Pseudo code frame to display on screen. |
| sortIncrease | Do ascending sort. |
| sortDecrease | Do descending sort. |
| getDigits | Get the digit of the corresponding row of the original numbers |
| setValueCountBox | Store count of each character into count array(countBoxs). |
| updateValueCountBoxASC | Change value of count array(countBoxs), now countBoxs[i] contains actual position of " i " character in output array (using in sort ascending). |
| updateValueCountBoxDESC | Change value of count array(countBoxs), now countBoxs[i] contains actual position of " i " character in output array (using in sort descending). |
| printToRadixBox | Put the characters in order into the output array(radixBoxs), display on the screen and update the order in the original array(elementBoxs) through each loop. |
| resetCountBox | Reset value of count array through each loop. |

1. **Explanation of the design**

* **Inheritance:**

Three sort types (Merge sort, Counting sort, Radix sort) inherit from **Sort** class.

**InitialElementBoxs** inherits from **ElementBoxs**.

* **Polymorphism:**

All specific sort algorithms are upcasted to the abstract class **Sort**, and used in **SortFrame** as attribute (algorithm: Sort).

* **Abstraction:**

We make **Sort** abstract because it is too general, and three sort types inherit **Sort** to create their instances.

* **Aggregation:**

**PointRun, Sort, ElementBox** and **InitialElementBoxs** are parts of **SortFrame**.

**ElementBox** is a part of **ElementBoxs**.

**PointRun, Instruction** and **ElementBox** are parts of **Sort**.

Beside one attribute of **ElementBox** inheriting from **Sort, CountingSort** and **RadixSort** have some other attributes of type **ElementBox**.