A red and yellow sign

Description automatically generated with low confidence

**Report on the Mini Project**

Topic: Demonstration of sorting algorithms on an array  
Algorithms: Merge sort, radix sort, counting sort

Team 16

Nguyễn Tống Minh – 20204885  
Lý Nhật Nam – 20204886  
Phạm Thanh Nam – 20204921  
Nguyễn Xuân Nam – 20200422

Course: Object-oriented Programming 131678 – 2021.2  
Dr. Nguyen Thi Thu Trang  
14/07/2022

# ACKNOWLEDGEMENT

To finish this project, we have received many helps from people. We would like to express my sincere thanks and gratitude to all these respected people.

First, we are thankful to Dr. Nguyen Thi Thu Trang for letting us work on this project as well as her enthusiastic support and detailed guidance in completing this project. We got to learn a lot more about this project especially in OOP technique and Java language which will be very helpful for us not only now abut also in the future.

Secondly, we would like to thank some of our classmate for their times and effort that help us answer some questions support our report.

# TABLE OF CONTENT

[ACKNOWLEDGEMENT 2](#_Toc108693117)

[TABLE OF CONTENT 3](#_Toc108693118)

[SUMMARY 4](#_Toc108693119)

[1. INTRODUCTION 5](#_Toc108693120)

[2. ASSIGNMENT OF MEMBER 6](#_Toc108693121)

[3. MINI PROJECT DESCRIPTION 7](#_Toc108693122)

[3.1. Project Overview 7](#_Toc108693123)

[3.2. Project Requirement 7](#_Toc108693124)

[3.3. Usecase diagram and explanation 8](#_Toc108693125)

[4. DESIGN 10](#_Toc108693126)

[4.1. General class diagram 10](#_Toc108693127)

[4.2. Detailed class diagram of each package 11](#_Toc108693128)

[4.2.1. Package view 11](#_Toc108693129)

[4.2.2. Package controller 16](#_Toc108693130)

[4.2.3. Package sorting 18](#_Toc108693131)

[4.2.4. Package listener 20](#_Toc108693132)

[4.2.5. Package util 23](#_Toc108693134)

[4.2.6. Package exception 25](#_Toc108693135)

[5. CONCLUSION 26](#_Toc108693136)

# SUMMARY

Our project concentrates to visualize 3 sorting algorithms: Merge Sort, Counting Sort and Radix Sort in bar char in the purpose of completing our class assignments and also creating an useful Java Swing application for every programmer to use when they need to visualize these algorithms in a colorful and interactive way. The report is to remark all our main point of the project that is contribution list, an overview, and the design. The overview is the place where we discuss about the requirements our project need to meet and the use case diagram that show the interaction between user and the application. The design is indicated the general relation between each part of the project and the detailed class in each package with every method and attribute as well as some explanation of the connection between them. Finally, we will present the OOP technique used in the project in our scope of knowledge.

# INTRODUCTION

Sorting is a basic concept of Computer Science that every programmer should have known. Evident by their definition, they are used to sorting data set with similar properties. There are a lot of efficient sorting algorithms all over the world with their own distinct implementations and strategies. However, to understand and absorb these algorithms is not every time easy. This leads to a solution of illustrating each step of the algorithms for learner can follow and eliminate the boredom from reading code by visualizing these steps in chart. Our project provides a Java application to deal with the visualization for 3 kinds of sorting algorithms: Merge Sort, Counting Sort and Radix Sort with the aim of making sorting algorithms become easier as well as meeting our requirements for our class mini project.

This report presents every aspect in our project in detail which include our team assignment, project description and project design. The assignment is for grading, which will clarify the contribution of each member to the project and our basic information to search in the student list. Moreover, the project will be described carefully via its feature, requirement and our use case diagram with detailed explanations and notes. Finally, the design provides an overview for the project through the general class diagram and a closer look to each package in the general diagram with all the method and attribute as well as the OOP technique used in the projects.

# ASSIGNMENT OF MEMBER

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Member | Nguyen Tong Minh | Ly Nhat Nam | Pham Thanh Nam | Nguyen Xuan Nam |
| Information | 20204885  [minh.nt204885@sis.hust.edu.vn](mailto:minh.nt204885@sis.hust.edu.vn) | 20204886  [nam.ln204886@sis.hust.edu.vn](mailto:nam.ln204886@sis.hust.edu.vn) | 20204921  [nam.pt204921@sis.hust.edu.vn](mailto:nam.pt204921@sis.hust.edu.vn) | 20200422  [nam.nx200422@sis.hust.edu.vn](mailto:nam.nx200422@sis.hust.edu.vn) |
| Work | - Package test  - Package sorting  - Class ArrayUtils  - UML Diagrams  - README.md  - *Pull request feedback* | - Package view  - Package controller  - Package listener  - Class ColorUtils  - Report writing  - *Pull request feedback* | - Class HomeScreen  - Class HomeController  - Class SortListener  - Package component (50%)  - Report writing  - *Pull request feedback* | - Package component (50%)  - Package exception  - Class DataUtils  - Presentation design  - *Exception handler* |

# MINI PROJECT DESCRIPTION

## Project Overview

The mission of our project is to build an application that visualizes three sorting algorithms, namely merge sort, counting sort, and radix sort.

Due to the main purpose of visualization is to help users get a better insight into how an algorithm works, we have put some restrictions on our application:

1. Only non-negative (>0) numbers are allowed to be an array's element.

2. The array size used for visualization has a max size of 100 elements.

3. A valid array in Radix Sort, Merge Sort only has the max value is 1000, and that value in Counting Sort is 20.

## Project Requirement

For a better understanding and visualization, we have added some features to our application:

- On the main menu: title of the application, 3 types of sort algorithms for a user to choose, help, about, and exit.

+) User must select a sort of type to start the demonstration

+) Help: show the basic usage and aim of the program

+) About: show the information about the course, the lecturer, the teaching assistants, and our team members.

+) Exit: exit the program. The application should ask for confirmation before closing.

- In the demonstration:

+) In the Navbar:

+) The name of the sorting algorithm

+) Back button: back to the main menu

+) Exit button: exit the program. The application should ask for confirmation before closing.

+) In the middle of the screen where the sorted representation of the algorithm

+) In the sidebar left has > button:

+) A button for creating the array: The user can choose to either randomly create an array or input an array for the program

+) A button for starting the algorithm with the created array. The application has to show clearly each step of the sorting

+) In the sidebar right has < button: show the information each action of sorting of the sorting algorithms.

+) In the footer of the screen:

+) A bar for the user to adjust the sorting speed

+) There are 5 buttons in media control when sorting: start, backward, pause, play, forward, and end. Besides that, users can control the sorting process drag and drop a bar next to these buttons.

+) Help: show the information about the sorting algorithm, time complexity, and space complexity its

+) About: show the information about the course, the lecturer, the teaching assistants, and our team members.

## Use case diagram and explanation

Figure Use case diagram

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

To be more specific:

- Select a sort of type use case: Allow the user to choose an algorithm to sort with an array.

- In the visualize algorithms use cases, the application will:

1. Take all user commands from the GUI such as: whether the user wants to customize/randomize the user's array, and which array size the user wants to create.

2. Notice the user if there is anything wrong with the user's input if he were to choose the customize array mode.

3. Run the designated algorithm to render user input into a sequence of steps of visualization.

4. Wait for the next command from the user, if the user wants to manipulate sort processing do the visualization manually or automatically.

5. Run/Reset the visualization based on user command.

6. See the steps in the sorting process

- View help menu uses case: Show the user manual and application restriction to the user under.

- Back to the main menu use case: Backward to the main menu where to choose sorting algorithms.

# DESIGN

## General class diagram

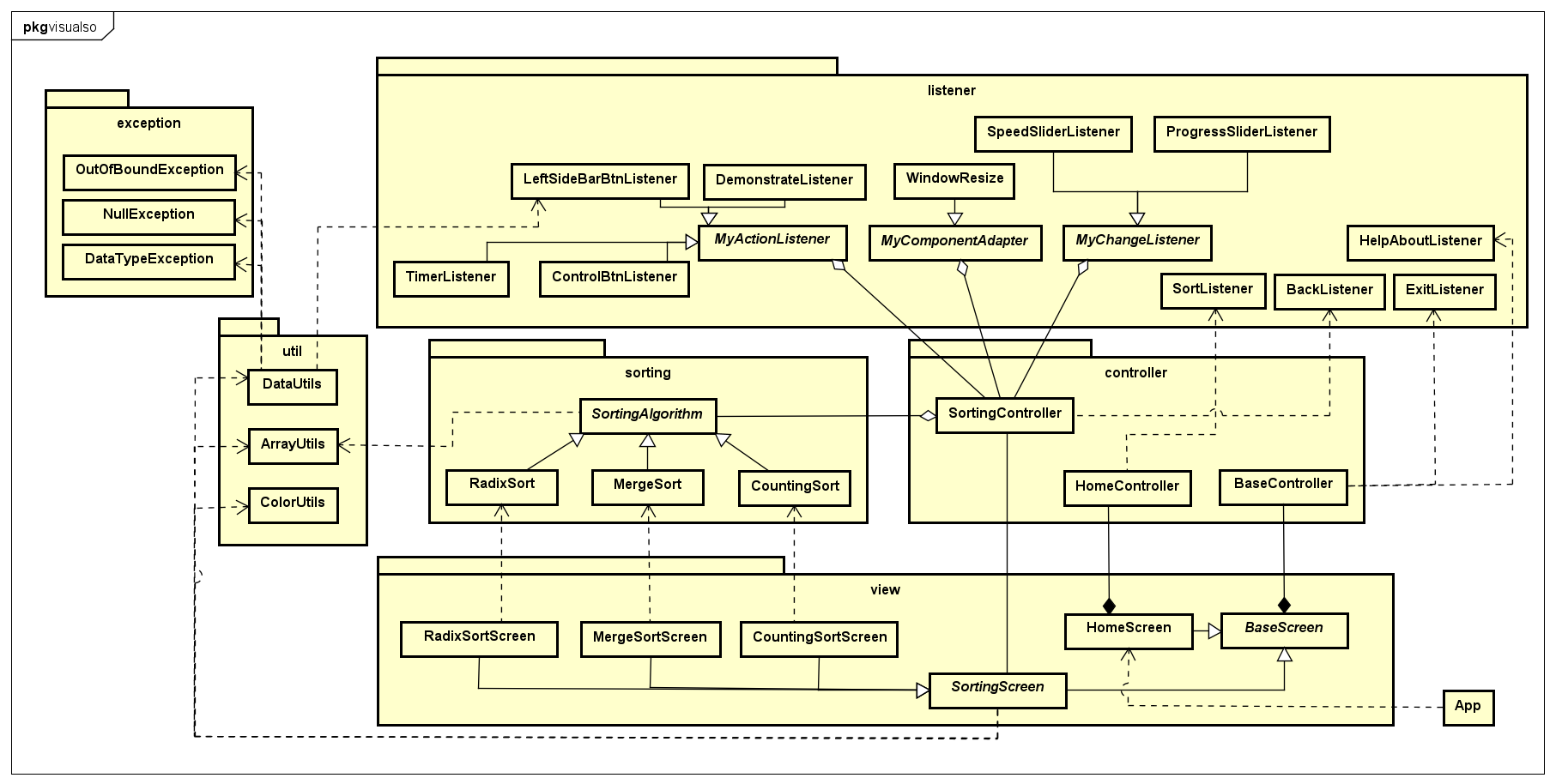


Figure : General Class Diagram

|  |  |  |
| --- | --- | --- |
| Package Name | Package Usage | Inside Class |
| view | Store all the screen class in charge of GUI | BaseScreen  - HomeScreen  - SortingScreen  + MergeSortScreen  + CountingSortScreen  + RadixSortScreen |
| controller | Store all the controller class that match with corresponding screen | BaseController  HomeController  SortingController |
| sorting | Store all the sorting backend model | SortingAlgorithm  - MergeSort  - CountingSort  - RadixSort |
| listener | Store an extended Listener for each button action in controller | SortListener  BackListener  ExitListener  HelpAboutListener  MyActionListener  - LeftSideBarBtnListener  - DemonstrateListener  - TimerListener  - ControlBtnListener  MyChangeListener  - SpeedSliderListener  - ProgressSliderListener  MyComponentAdapter  - WindowResize |
| util | Store all static method used generally in other packages | DataUtils  ArrayUtils  ColorUtils |
| exception | Store all the exception appeared in the project | OutOfBoundException  NullException  DataTypeException |

## Detailed class diagram of each package

### Package view

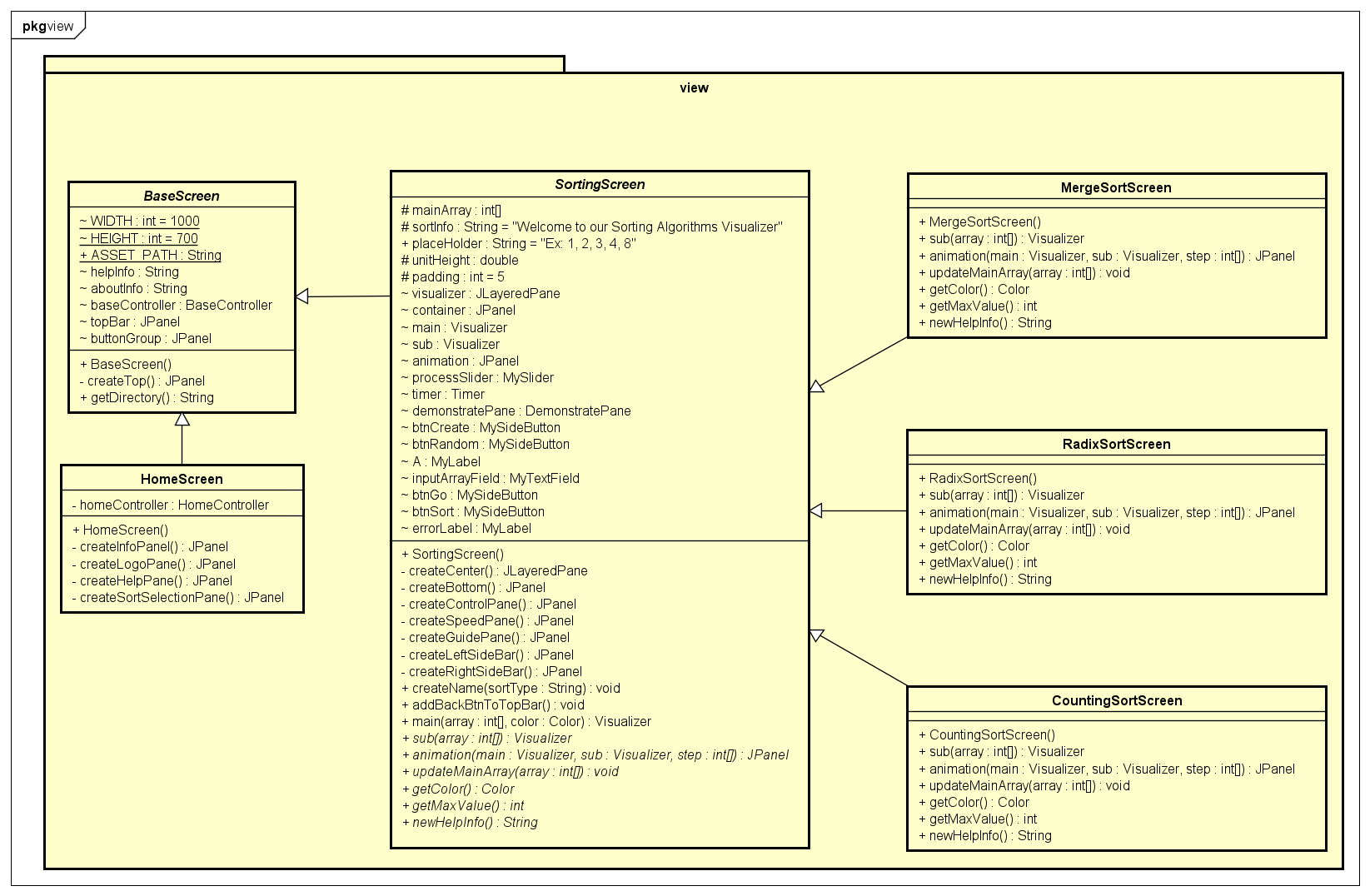


Figure View Package Class Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class Name | Class Usage | Attribute | Method | OOP Technique |
| BaseScreen | An abstract class, generalization of HomeScreen and SortingScreen share the same topBar | **-WIDTH**: default width of all windows  -**HEIGHT:** default height of all windows  **+ASSET\_PATH**: the directory to the asset folder (source for image,…)  ~**helpInfo:** information in help window of HomeScreen  ~**aboutInfo:** information in about window of all Screen  **~baseController**: connect to the its controller  **~topBar**: the bar on the top of window which contain logo, buttonGroup, sort name)  **~buttonGroup**: a component of topBar including back button for SortingScreen and exit button for all Screen | -**createTop()**: method to create and design the topBar  **+getDirectory():** return the ASSET\_PATH | Has-a BaseController and BaseController fully dependent on this (Composition) |
| HomeScreen | The menu screen | **-homeController:** connect to its controller | -**createInfoPane():** create the Information Pane (include LogoPane and HelpPane)  -**createLogoPane():** create logo pane( include logo and small description)  -**createHelpPane():** create HelpPane (include help button and about button in menu)  -**createSortSelectionPane():**  create SortSelectionPane (include 3 block with name sort type for user to choose) | Inherit from BaseScreen(Inheritance)+  Has-a HomeController and HomeController fully dependent on this (Composition) |
| SortingScreen | An abstract class, represent for the sort screen | **#mainArray:** the array of that Screen, when you updateMainArray, it means you call another Screen that have the different mainArray  **#sortInfo:** default text in demonstratePane  +**placeHolder:** the place holder in the inputArrayField, guide user to know how to pass the input  #**unitHeight**: height of a unit in pixel of column in bar char  #**padding**: space between two adjacent columns in bar chart  -**visualizer**: the center part of the window( include container, animation and some buttons)  -**container**: behind layer contain main and sub  -**main**: pane to visualizer every steps of sorting (link to arrayLog in soritng package)  -**sub**: pane to support main to visualize every steps of sorting(link to tempLog in sorting package)  **-animation:** above layer to support main to visualize every steps of sorting(link to pointerLog in sorting package)  -**processSlider:** show the process of the sorting by a slider and link to ProgressSliderListener  -**timer**: for the process can automatically run in playing state  -**demonstratePane**: show the steps in English on the right(link to guideLog in sorting package)  -**btnCreate**: button to open create button group  **-btnRandom:** button to create random data  -**A**: a label with text  -**inputArrayField**: the text field that receive input data  -**btnGo**: button to send the input data to controller  -**btnSort**: button to send signal to controller so that controller will connect to sort algorithms and do the sort  -**errorLabel**: label link to data util and go button that if button raise exception, then handle it | **-createCenter():** create the visualizer and add component to it  -**createBottom():** create control Bar ( which include ControlPane, SpeedPane and GuidePane)  -**createControlPane**(): pane that contain all the control button( play, resume, for/backward, skip, restart) and the processSider  -**createSpeedPane**(): pane that contain speed slider for changing speed  -**createGuidePane**(): pane with help button for each Screen and about button link to HelpAboutListener  -**createLeftSideBar**(): create left side with the button “>” link to LeftSideBarBtnListener  **-createRightSideBar():** create right side with the button “<” link to DemonstrateListener  +**createName(sortType**): create a pane with name label of each sort type  +**addBackBtnToTopBar**(): add back button to SortingScreen link to BackListener  +**main(array)**: visualize a array in bar chart corresponding with each step  *+sub(array):* visualize the supported pane for main, however, each sort will have different sub so the method must be abstract.  *+animation(main, sub, step):* visualize the pointer for main and sub  *+updateMainArray(array):*  When the main array change( due to go or random buttom) create a new Screen that have new mainArray = array  *+getColor():* return the Color of each SortingScreen  *+getMaxValue():* return the maximum value of mainArray of each SortingScreen  *+newHelpInfo():*return the content in help window of each SortingScreen | Inherit from BaseScreen(Inheritance)+  Has-a HomeController and HomeController fully dependent on this (Composition) |
| MergeSortScreen | Screen for merge sort |  | +**sub(array):**visualize the supported pane for main  +**animation(main,sub,step):** visualize the pointer for main and sub  +**updateMainArray():**When the main array change( due to go or random button) create a new Screen that have new mainArray = array  +**getColor():**return the Screen bar chart Color  +**getMaxValue():**return the maximum value of mainArray of the Screen  +**newHelpInfo():**return the content in help window of the Screen | Inherit from SortingScreen  (Inheritance)  +  Override method for each Screen(Polymorphism) |
| CountingSortScreen | Screen for counting sort |  |
| RadixSortScreen | Screen for radix sort |  |

### Package controller

Figure Controller Package Class Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class Name | Class Usage | Attribute | Method | OOP Technique |
| BaseController | A controller for BaseScreen |  | +**helpButtonClicked(name, helpInfo):** connect help and about button to their listener via title and content of the frame  **+exitButtonClicked():** connect exit button to ExitListerner |  |
| HomeController | A controller for HomeScreen |  | **+sortSelection(name):** connect exit button to SortListerner to select the type of sort via name( e.g Merge Sort,…) |  |
| SortingController | A controller for SortingScreen | -**speed**: integer speed of the sorting process, apply to the timer delay via formula *Delay = 1000 –speed\*10*  **-isPlay**: a boolean return state of the process( true if user click play, false if user click again)  **-isSorting:** return state of the process( true if in sorting process, false if initial state,or update array state)  **-curStep**: mark current step of sorting process  **+MAX\_LENGTH:** the valid maximum length of the input array  **-sortingScreen:** SortingScreen that connect to the view  **-sortingModel:**  SortingAlgorithm that connect to the sorting | **+changeWindowSize():** connect the controller to WindowSize listener to change the component size fit with window size  **+changeSpeed(count):** connect the controller to SpeedSliderListener to adjust the speed  **+changeProgressSlider():**  connect the controller to ProgressSliderListener to interact with the processSlider  **+rightSideBtnClicked():**  connect the controller to DemonstrateListener to show the steps on the right  **+leftSideBtnClicked():** connect the controller to LeftSideBarBtnListener to control the create, sort feature  **+setTimer():** connect the controller to TimerListener to automatic sort  **+controlButtonCLicked():** connect the controller to ControlBtnListener to manipulate sorting process( play, pause, for/backward, skip, restart)  **+getArrayStates():** get arrayLog attribute of model  **+getTempStates():**get tempLog attribute of model  **+getGuideStates():**get guideLog attribute of model  **+getPointerStates():**get pointerLog attribute of model  **+getStep():** get numSteps attribute of model  **+backButtonClicked():** connect the controller to the BackListener to back to HomeScreen | Has-a sortingModel(Aggregation) |

### Package sorting

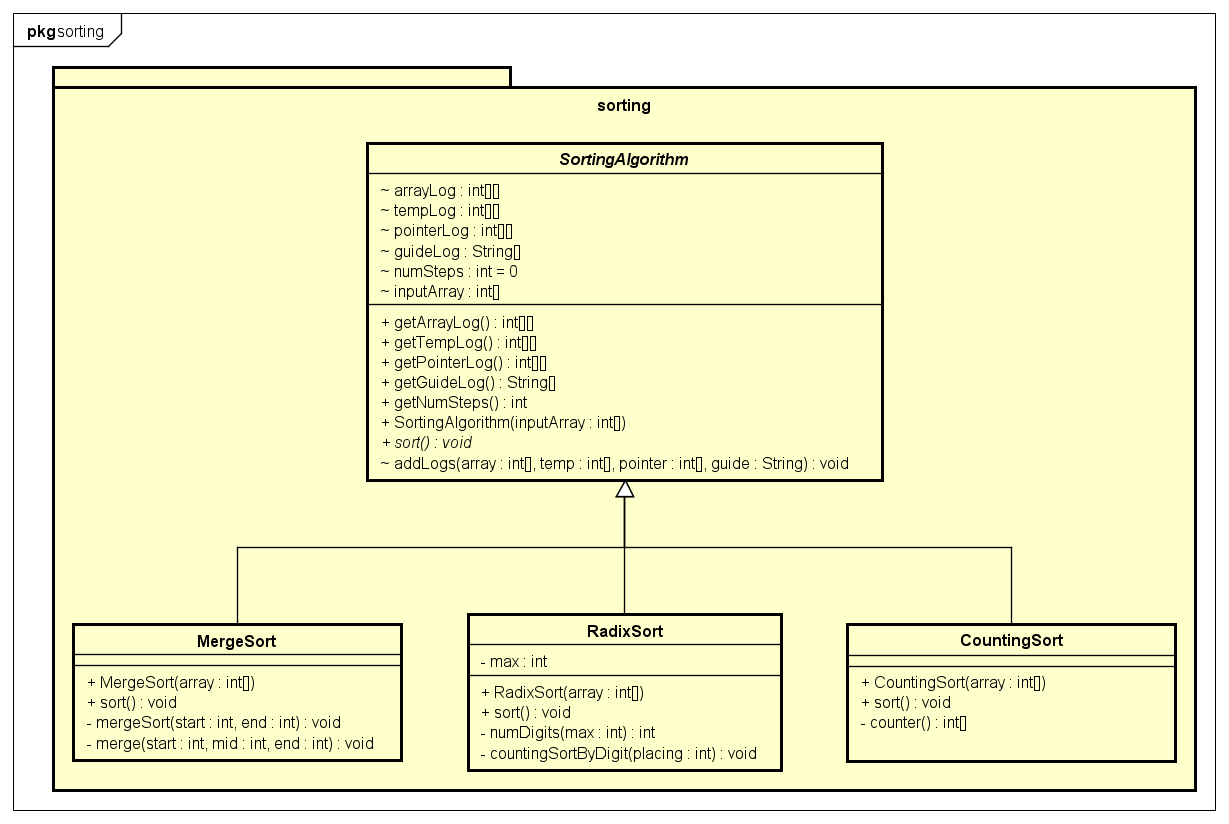


Figure Sorting Package Class Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class Name | Class Usage | Attribute | Method | OOP Technique |
| SortingAlgorithm | An abstract class, generalization of 3 types of sort | ~arrayLog: return the main array of the step in the sorting  ~tempLog: return the sub array of the step in the sorting  ~guideLog: instructions for converting between steps in processing sort  ~pointerLog: return 2 pointers per step, one pointer for arrayLog, and the other for tempLog  ~numSteps: the number of steps to be sorted  ~InputArray: the data of the array needs to be sorted | +getArrayLog(): get arrayLog attribute of model  +getTempLog(): get tempLog attribute of model  +getGuideLog(): get guideLog attribute of model  +getPointerLog(): get pointerLog attribute of model  +getNumSteps(): get numSteps attribute of model  *+sort():* abstract method  ~addLogs(array, temp, pointer, guide): store the corresponding stage/step of the sorting algorithm running |  |
| MergeSort | Merge Sort Algorithm |  | +sort(): sort by Merge Sort and add corresponding state by addLogs  -mergeSort(start, end): split main array into sub arrays when it has only 1 element  -merge(start, mid, end): merge the sub-arrays together and sort them in order | Inherit from SortingAlgorithm  (Inheritance) |
| CountingSort | Counting Sort Algorithm |  | +sort(): sort counter array in order  -counter(): count the number of elements with the same value in the array |
| RadixSort | Radix Sort Algorithm | -max: return the largest number in array | +sort(): sort array by digit  -numDigits(max): return the number of digits of *max*  -countingSortByDigit( digitplacing): sort array by (digitplacing)th digit |

### Package listener

### 

Figure Listener Package Class Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class Name | Class Usage | Attribute | Method | OOP Technique |
| SortListener | A listener in HomeScreen to choose the type of sort | **-name:** a string that equals to the name of sort type ( e.g Merge Sort,…) | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action(selection sort type) when this listener trigger | Inherit from awt.ActionListener  (Inheritance) |
| BackListener | A listener in SortingScreen to return to the HomeScreen |  | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action (go to HomeScreen) when this listener trigger |
| ExitListener | A listener in BaseScreen to ask user to quit and quit |  | **+actionPerformed (ActionEvent e)**: an override method from ActionListener, define the action (ask user to quit and quit ) when this listener trigger |
| HelpAboutListener | A listener in BaseScreen to show help window and about window for both HomeScreen and SortingScreen | **-helpInfo:** the content in the help or about window  **-name:** the title of the help or about window | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action (show guide and info) when this listener trigger |
| MyActionListener | An abstract listener to define listeners that require a controller | -**sortingController**: SortingController that its attribute value will be changed due to the listener | **+actionPerformed (ActionEvent e):** an override method from ActionListener | Inherit from awt.ActionListener(Inheritance)+  Has-a sortingController(Aggregation) |
| LeftSideBarBtnListener | A listener in SortingScreen applying for button of leftside | **-sorted:** boolean, true if the array have sorted and not update new array, false otherwise | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action (show the button on the left, create, random, get input, go and start sorting ) when this listener trigger | Inherit from MyActionListener(Inheritance) + Overide actionPerformed(Polymorphism) |
| DemonstrateListener | A listener in SortingScreen applying for button of leftside |  | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action (show the steps of the soritng process) when this listener trigger |
| TimerListener | A listener in SortingScreen applying for the timer |  | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action of the timer ( automatic sort when the controller in state isSorting and isPlay |
| ControlBtnListener | A listener in SortingScreen applying for control button |  | **+actionPerformed (ActionEvent e):** an override method from ActionListener, define the action (reset, backward, play, resume, forward, skip) when this listener trigger |
| MyChangeListener | An abstract listener to define listeners that require a controller | **-sortingController:** SortingController that its attribute value will be changed due to the listener | **+stateChanged(ChangeEvent e):** an override method from ChangeListener | Inherit from awt.ChangeListener(Inheritance)+  Has-a sortingController(Aggregation) |
| ProgressSliderListener | A listener in SortingScreen to show current step of sorting process |  | **+stateChanged (ChangeEvent e):** an override method from ChangeListener, its action related to the current step of sorting process, and redraw the visualizer to that step when the slider change value | Inherit from MyChangeListener(Inheritance) + Overide stateChanged(Polymorphism) |
| SpeedSliderListener | A listener in SortingScreen to show current speed of sorting process | **-count:** a Label that express the current speed of the process | **+stateChanged(ChangeEvent e):** an override method from ChangeListener, its action related to the current speed of sorting process, and redefine the timer delay when the slider change value |
| MyComponentAdapter | An abstract listener to define listeners that require a controller | **-sortingController:** SortingController that its attribute value will be changed due to the listener |  | Inherit from awt.ComponentAdapter(Inheritance)+ Has-a sortingController(Aggregation) |
| WindowResize | A listener in SortingScreen to change the size of every component when the frame’s size change |  | **+componentResize(ComponentEvent e):** an override method from ComponentAdapter, apply change for every component when the frame size change | Inherrit from MyComponentAdapter(Inheritance) + Overide componentResize(Polymorphism) |

### Package util

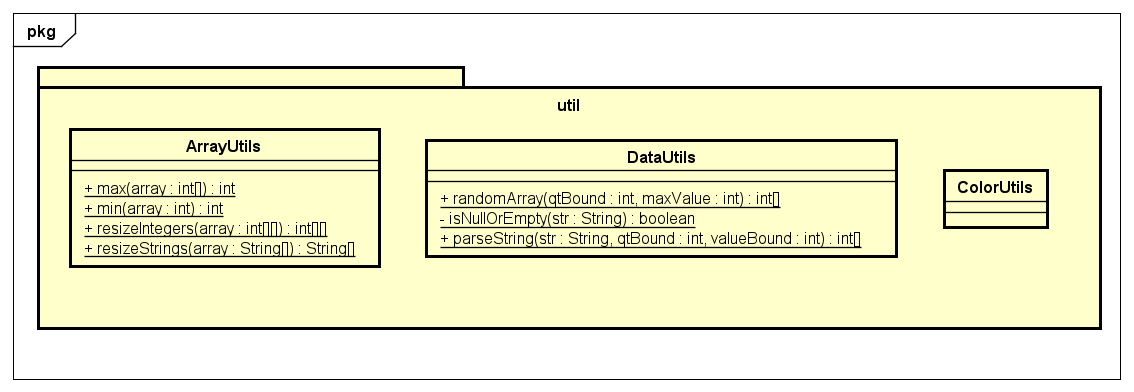


Figure Util Package Class Diagrams

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class Name | Class Usage | Attribute | Method | OOP Technique |
| DataUtils | Use to create Array from the user input |  | **+randomArray(qtBound, maxValue):** return a randomized array with max number and max value of array are qtyBound and maxValue respectedly  **+isNullOrEmpty(str):**  return true if str is null or only “,”; return false otherwise  **+parseString(str, qtBound, valueBound):** return a customized array from the input after check its validation |  |
| ArrayUtils | Useful method for sorting algorithm |  | **+max(array):** return the maximum of array  **+min(array):** return the minimum of array  **+resizeInteger(array):** return array with doubled size  **+resizeString(array):** return array with doubled size |  |
| ColorUtils | Define color use in the project |  |  |  |

### Package exception

Figure Exception Package Class Diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class Name | Class Usage | Attribute | Method | OOP Technique |
| OutOfBoundException | An exception called if input array has length or value exceed the maximum of each Sorting Screen |  |  | Inherit from Exception  (Inheritance) |
| NullException | An exception called if input array is null or empty |  |  |
| DataTypeException | An exception called if the input array does not contain positive numbers |  |  |

# CONCLUSION

The application has written in Java Swing and has successfully satisfied all the project requirements in the class. It is runnable, which has been tested by our team and in our opinion, have a good visualization for 3 types of sorts: Merge Sort, Counting Sort and Radix Sort. However, Visualization is a form of expression; it can take numerous forms to convey the idea, yet it always provides a new outlook if it is implemented properly and maybe the project was successful in delivering the idea of visualization of sorting algorithm through a java application and providing a separate look into the algorithm apart from lines of code and syntaxes. The design is also shown with 5 main package, and the use of many OOP technique such as Inheritance, Polymorphism, Aggregation, Composition… in our range of knowledge. This project still has plenty of space for improvements which can be achieved through better UI design, accessibility, application logic, performance, and/or tools. Visualization is the means to communicate between algorithms and programmers. Now our aim is to develop the project to attract more people to learn and understand these abstract ideas of algorithms.