OOP Project Report

Members

Name	Student ID	Responsibilities
Tran Van Toan	20214932	- Algorithm (Shellsort) - Animation Implementation - Animation Control Implementation - Bug Fixing.
Nguyen Thi Trang	20194458	- Algorithm (Merge Sort) - Detail Package Class Diagram - Report - Presentation Design - GUI Design
Nguyen Quang Tri	20210860	- Project Design (Overall system design) - Algorithm (Abstract SortingAlgorithm Class) - Animation Design - GUI Design - Components Design - General Class Diagram
Dang Kieu Trinh	20214933	- Algorithm (Selection Sort) - Use Case Diagram, General Class Diagram, Detail package class diagram. - Components design. - Animation Design

Contribution

algorithm	
SortingAlgorithm.java	Nguyen Quang Tri
ShellSortAlgorithm.java	Tran Van Toan
SelectionSortAlgorithm.java	Dang Kieu Trinh
MergeSortAlgorithm.java	Nguyen Thi Trang
component	
ArrayGraphic.java	Nguyen Quang Tri
ButtonComponent.java	Nguyen Quang Tri, Dang Kieu Trinh, Tran Van Toan
CardComponent.java	Nguyen Quang Tri, Dang Kieu Trinh
InfoWindowComponent.java	Nguyen Quang Tri, Nguyen Thi Trang, Dang Kieu Trinh
LabelComponent.java	Nguyen Quang Tri, Nguyen Thi Trang
SliderBarComponent.java	Nguyen Quang Tri
TextAreaComponent.java	Nguyen Quang Tri
TextFieldComponent.java	Nguyen Quang Tri
ToggleMenuComponent.java	Nguyen Quang Tri
component.utils	
ArrayUtil.java	Nguyen Quang Tri, Tran Van Toan
Element.java	Nguyen Quang Tri
controller	
Controller.java	Nguyen Quang Tri
SortController.java	Tran Van Toan, Nguyen Quang Tri
listener	
ScreenListener.java	Nguyen Quang Tri
test.algorithm	
*	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang

test.component	
*	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
test.component.utils	
*	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
test.utils	
*	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
test.view	
*	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
view	
HomeScreen.java	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
MergeSortScreen.java	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
Screen.java	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
SelectionSortScreen.java	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
ShellSortScreen.java	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
SortScreen.java	Nguyen Quang Tri, Tran Van Toan, Dang Kieu Trinh, Nguyen Thi Trang
Report	
*	Dang Kieu Trinh, Nguyen Thi Trang
Diagrams	
UseCaseDiagram	Dang Kieu Trinh
GeneralClassDiagram	Dang Kieu Trinh, Tran Van Toan, Nguyen Quang Tri
DetailedClassDiagrams	Dang Kieu Trinh, Nguyen Thi Trang
Presentation	
*	Nguyen Thi Trang

Demo video	
*	Dang Kieu Trinh
Readme	
*	Dang Kieu Trinh

Project Overview

Description

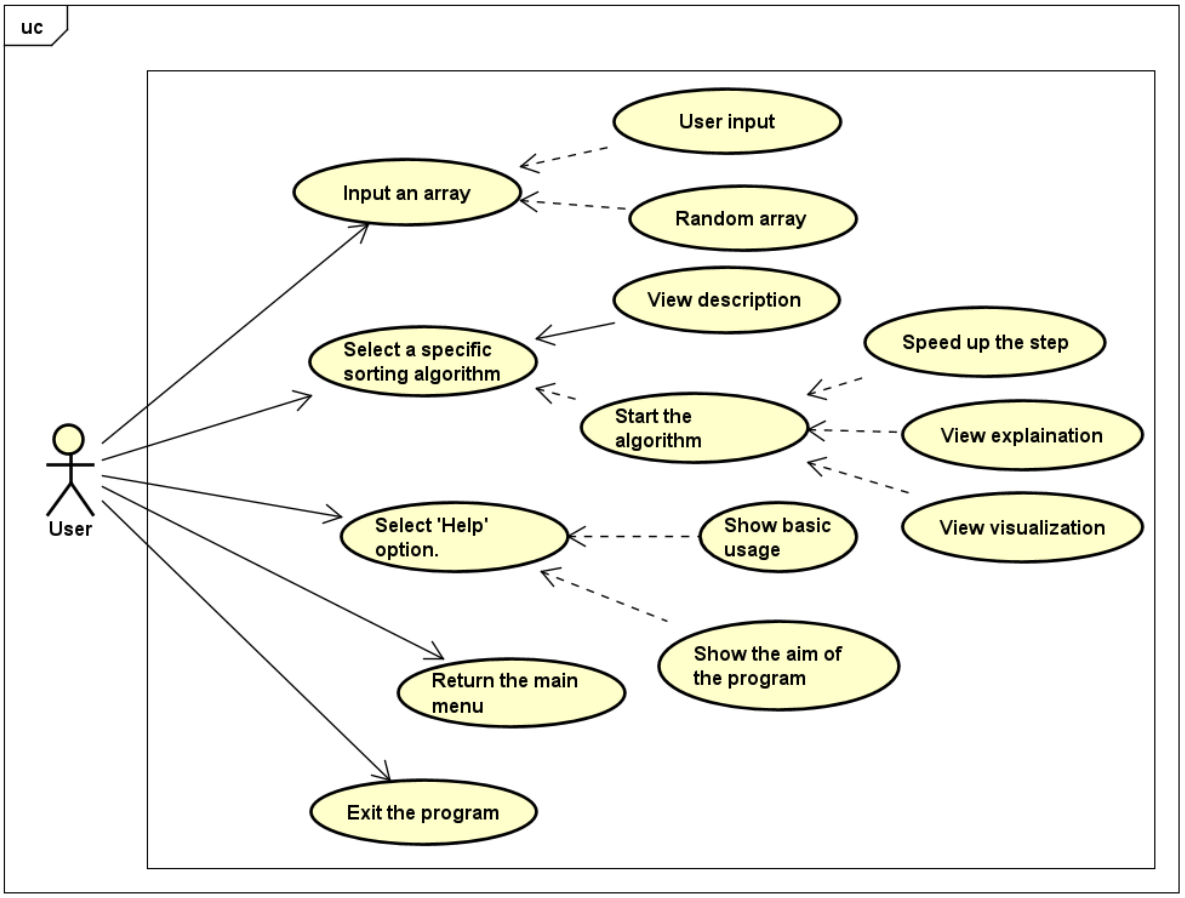
Array is the most basic structure of computer science. Most operations as well as other data structures are built and performed on arrays. This project's objective is to develop an application based on OOP in order to visualize three sorting algorithms on arrays: selection sort, merge sort and Shell sort.

Requirements

- On the main menu: title of the application, three types of sort algorithms for user to choose, help menu, quit
    - User must select a sort type in order to start the demonstration
    - Help menu show the basic usage and aim of the program
    - Quit exits the program. The application ask for confirmation before closing
- In the demonstration:
    - A button for creating the array: user can choose to randomly create an array or input an array for the program
    - A button for starting the algorithm with the program
    - A button for starting the algorithm with the created array. The application should show clearly each step of the sorting
    - A back button for user to return to main menu at any time
    - A bar Chart for visualizing the created array
    - First, previous, go, next, last button and process slider bar for user to control the sorting process

Explanation

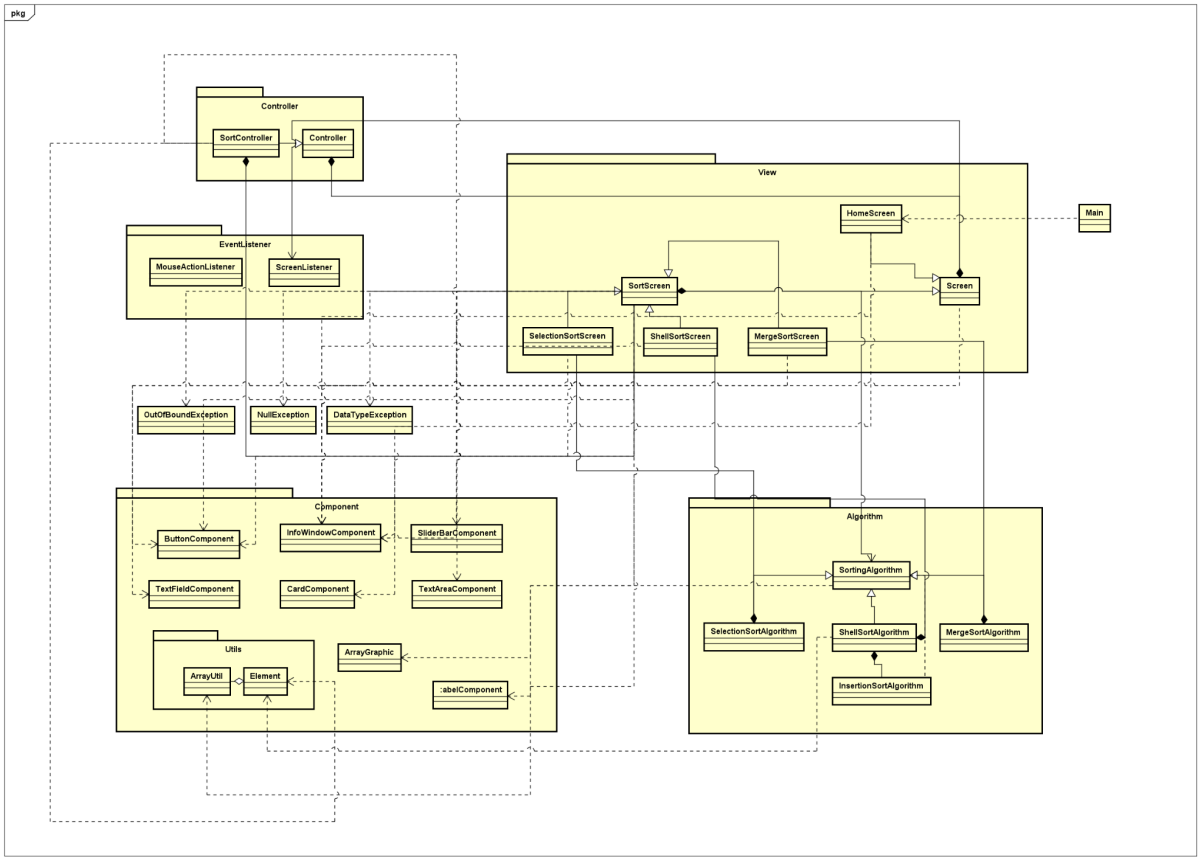
Use Case Diagram



Based on the project's objective and requirements, we have created a use-case diagram, which can be observed above. The user has the following options:
1. The user can select either to input an array or create a randomized array, with a maximum length of 30.
2. Once an array is chosen for the next step, the user can select a specific sorting algorithm to sort the array. After selecting the algorithm, the user can access the algorithm's description, initiate the sorting process, and manipulate the speed of each step or view explanations for each step.
3. If the user selects the 'Help' option, a detailed guide will be provided, explaining how to use the app and offering information about the project's objectives.

Throughout the app's usage, the user can opt to return to the main menu or exit the program at any time.

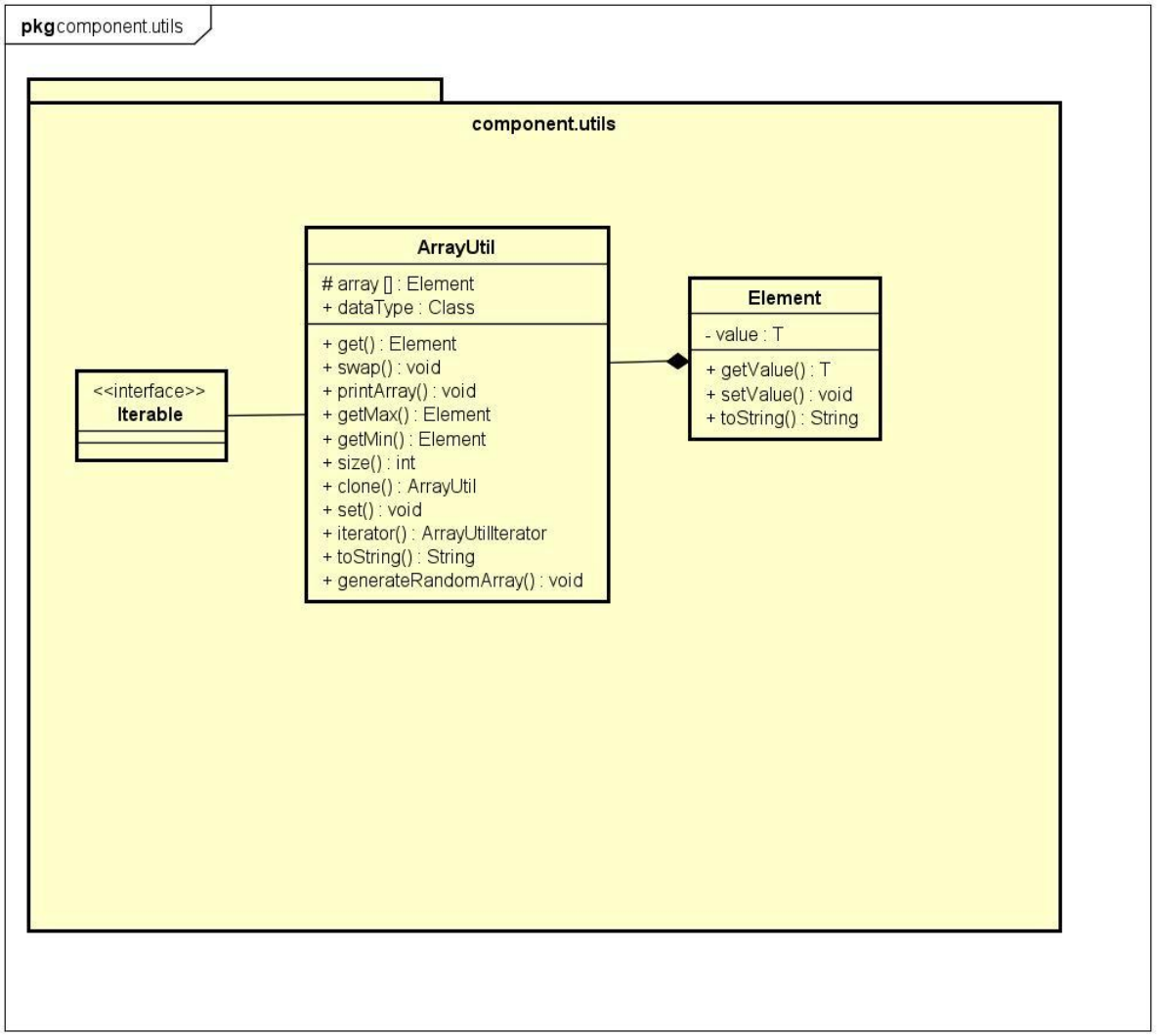
General class diagram



With the general class diagram, there are 6 main packages:
1. Package Components
2. Package Algorithm, which contains the three sorting algorithms: Merge Sort, Selection Sort and Shell Sort.
3. Package Listener
4. Package Controller
5. Package View

Detail class diagram of each package

1. Components.utils



This package - 'component.utils' - provides utility classes for working with generic elements and arrays.

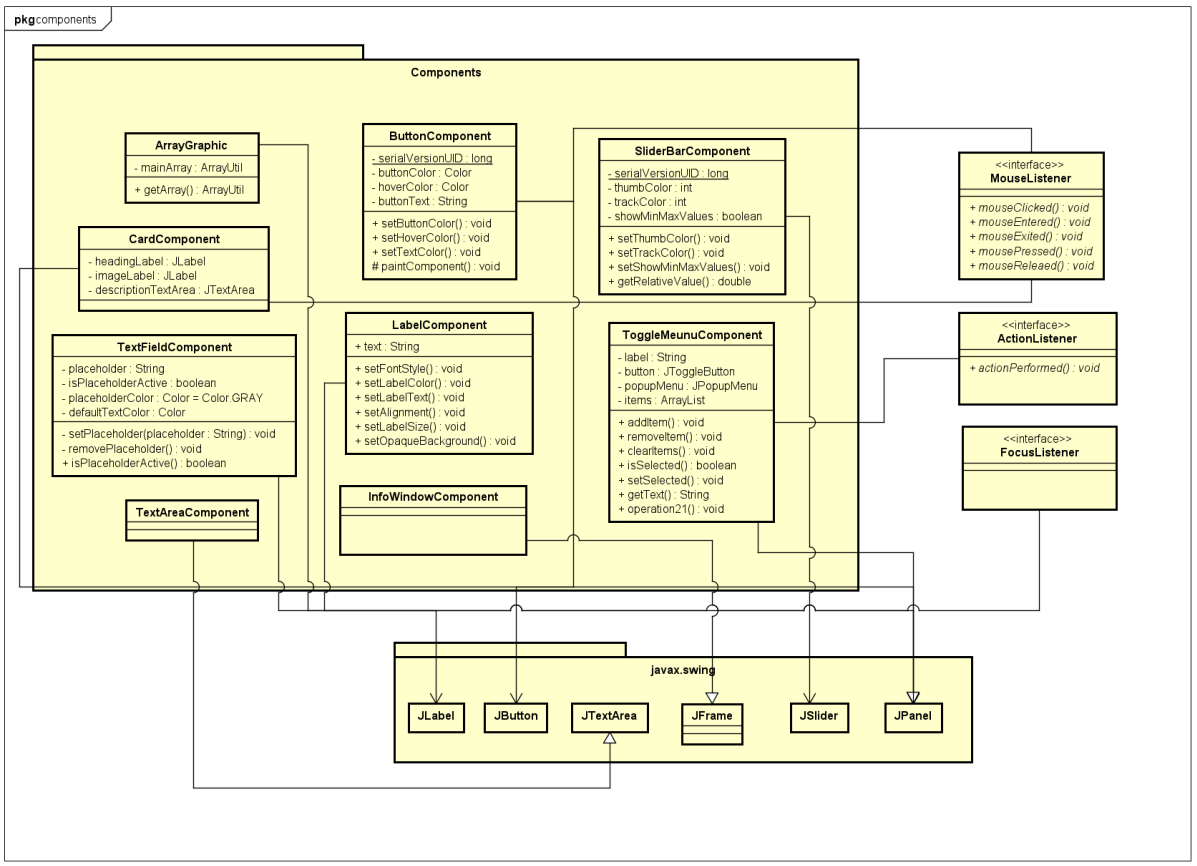
The `Element` class is a generic class that encapsulates a value of any type. It provides methods to access and modify the stored value. Additionally, it overrides the `toString()` method to provide a string representation of the value.

The `ArrayUtil` class is a utility class that operates on an array of `Element` objects. It implements the `Iterable` interface, allowing for iteration over the elements of the array. The class provides several useful methods, including:

Attributes explanation	Methods explanation
`array`: It is an array of type `Element<T>`, where `T` is a generic type parameter representing the type of elements stored in the array.	- `swap(int i, int j)`: Swaps the elements at indices `i` and `j` in the array.
`dataType`: This attribute is of type `Class<T>` and represents the data type of the elements stored in the `array`.	- `printArray()`: Prints the elements of the array to the console.
	- `getMax()`: Returns the element with the maximum value in the array.
	- `getMin()`: Returns the element with the minimum value in the array.
	- `size()`: Returns the size of the array.
	- `get(int index)`: Returns the element at the specified index in the array.
	- `clone()`: Creates a shallow copy of the `ArrayUtil` object.
	- `set(int index, Element<T> element)`: Replaces the element at the specified index with a new element.
	- `generateRandomArray()`: Generates a random array of elements of the same type as the elements in the original array.

The `ArrayUtil` class also provides an internal class `ArrayUtilIterator` that implements the `Iterator` interface. This iterator allows for easy traversal of the elements in the array.

2. Components



The Components package contains several classes that provide various graphical components for user interfaces.

2.1 ArrayGraphic

Attributes	Method
- mainArray: an ArrayUtil object	+ getArray(): a method to retrieve the array

The ArrayGraphic class extends JPanel and represents a graphical component that visualizes an array. This component can be used to display and manipulate array data in a graphical manner.

2.2 ButtonComponent

Attributes	Methods

‘buttonColor’: Represents the background color of the button.	setButtonColor: Sets the background color of the button
‘hoverColor’: Represents the background color of the button when the mouse is hovering over it.	setHoverColor: Sets the background color of the button when the mouse is hovering over it.
‘textColor’: Represents the color of the button's text.	setTextColor: Sets the color of the button's text.
‘buttonText’: Stores the text to be displayed on the button.	paintComponent: Overrides the paintComponent method to customize the rendering of the button. It handles the drawing of the button's background, text, and other visual aspects based on the current state (pressed, hovered, or normal).

The ButtonComponent class extends JButton and represents a customizable button component. The appearance of the button changes dynamically based on user interactions such as mouse presses and hovers.

2.3 CardComponent

Attributes	Methods
headingLabel: Represents the JLabel component used for displaying the heading text.	CardComponent: The constructor of the CardComponent class. It takes parameters such as the heading text, an Image object for the image (or null if no image), and the description text. It sets up the initial appearance and content of the card component.
imageLabel: Represents the JLabel component used for displaying the image (if available).	
descriptionTextArea: Represents the JTextArea component used for displaying the description text.	

The CardComponent class extends JPanel and represents a card-like component with a heading, an optional image, and a description. It is designed to display content in a visually appealing manner. The component supports mouse events and can be customized with different heading texts, images, and descriptions.

2.4 LabelComponent

Attributes	Methods
	LabelComponent: The constructor of the LabelComponent class. It takes a String parameter representing the text to be displayed on the label. It sets up the initial appearance and content of the label component.
	setFontStyle: Sets the font style for the label's text.
	setLabelColor: Sets the color of the label's text.
	setLabelText: Updates the text displayed on the label.
	setAlignment: Sets the horizontal and vertical alignment of the label's text within the label component
	setLabelSize: Sets the preferred size (width and height) of the label component.
	setOpaqueBackground: Sets the background color of the label component and makes it opaque.

The LabelComponent class extends JLabel and represents a simple label component that displays text. It allows you to set the text, font style, text color, alignment, and size. This component is useful for displaying descriptive or informative text in a user interface.

2.5 SliderBarComponents

Attributes	Methods

thumbColor: Represents the color of the thumb (the draggable handle) on the slider bar.	SliderBarComponent: The constructor of the SliderBarComponent class. It takes parameters for the minimum value, maximum value, and initial value of the slider. It sets up the initial appearance and behavior of the slider component.
trackColor: Represents the color of the track (the bar representing the range) on the slider bar.	setThumbColor: Sets the color of the thumb on the slider.
showMinMaxValues: A boolean value indicating whether to display the minimum and maximum values of the slider.	setTrackColor: Sets the color of the track on the slider.
	setShowMinMaxValues: Sets whether to display the minimum and maximum values of the slider.
	getRelativeValue(): Returns the relative value of the slider, which is a value between 0 and 1 representing the position of the slider's thumb relative to the range.
	paintComponent: Overrides the paintComponent method to customize the rendering of the slider. It handles the drawing of the thumb and track based on the current appearance settings.

The SliderBarComponent class extends JSlider and represents a slider bar component that allows users to select a value within a specified range. It provides customization options for the thumb color and track color. The appearance of the slider changes based on the selected value.

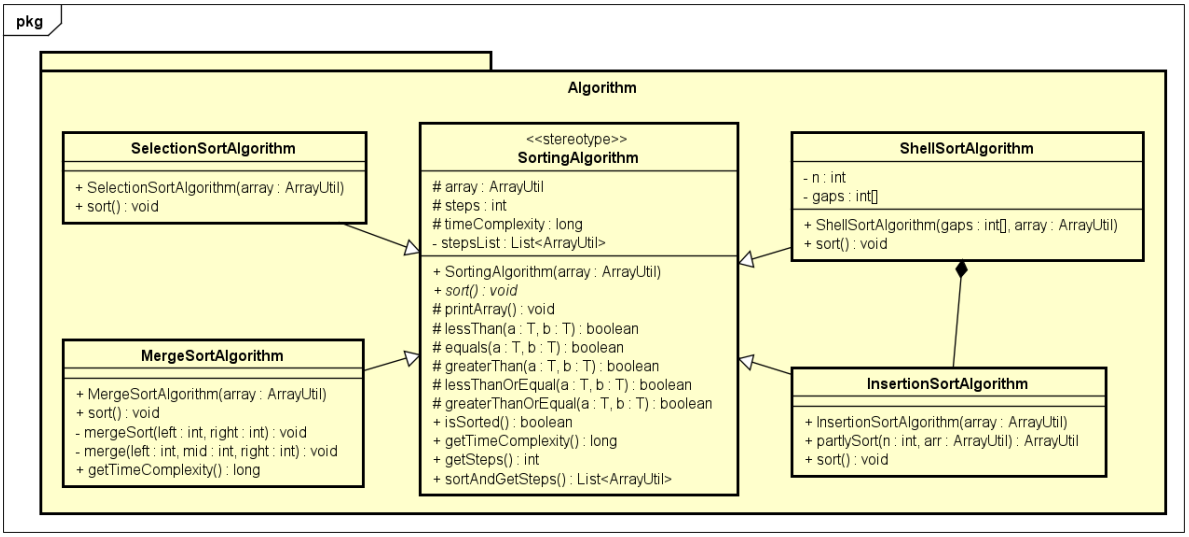
2.6 ToggleMenuComponent

Attributes	Methods
label: Represents the label or text displayed on the toggle button.	ToggleMenuComponent: The constructor of the ToggleMenuComponent class. It takes a

	String parameter representing the label or text to be displayed on the toggle button. It sets up the initial appearance and behavior of the toggle menu component.
button: Represents the JToggleButton component that serves as the toggle button.	addItem: Adds an item to the drop-down menu. The parameter item is a String representing the text of the item.
popupMenu: Represents the JPopupMenu component that displays the drop-down menu.	removeItem: Removes an item from the drop-down menu. The parameter item is a String representing the text of the item to be removed.
items: Represents an ArrayList that stores the items in the drop-down menu.	clearItems(): Clears all the items from the drop-down menu.
	isSelected(): Returns a boolean value indicating whether the toggle button is selected (toggled on).
	setSelected: Sets the selection state of the toggle button. The parameter selected is a boolean value indicating whether the button should be selected.
	getText(): Returns the text currently displayed on the toggle button.

The ToggleMenuComponent class extends JPanel and represents a toggle menu component with a drop-down menu. It consists of a toggle button and a popup menu that displays a list of items. Users can select an item from the menu by clicking on it. The component supports adding, removing, and clearing menu items.

3. Package algorithm



The `algorithms` package provides a collection of sorting algorithms that can be used to sort arrays of elements. The package includes an abstract class, `SortingAlgorithm`, which serves as a base for implementing specific sorting algorithms. Additionally, it offers several subclasses that provide concrete implementations of popular sorting techniques.

- Class SortingAlgorithm: an abstract class, generalization of four sort classes

Attributes	Methods
# array: is an instance of the ArrayUtil<T> and is the array to be sorted # steps: an integer counter to keep track of the steps taken by the sorting algorithm # timeComplexity: a long variable to record the time complexity of the sorting algorithm - stepsList: list that stores the array at each step in the sorting process	+ SortingAlgortihm(array: ArrayUtil): constructor method and it initializes array + *sort()*:abstract method. When this class is extended, sort() method must be overridden with the actual sorting algorithm # printArray(): print array # lessThan(a:T,b:T): compares two Comparable items a and b and return true if a is less than b # equals(a:T,b:T): compares two Comparable items a and b and return true if a is equal to b # greaterThan(a:T,b:T):compares two Comparable items a and b and return true if a is greater than b # lessThanOrEqual(a:T,b:T):compares two Comparable items a and b and return true if a is less than or equal to b

	# greaterThanOrEqual(a:T, b:T): compares two Comparable items a and b and return true if a is greater than or equal to b + isSorted(): checks whether array is sorted + getTimeComplexity(): return time complexity of sort algorithm + getSteps(): return number of steps of the sorting process + sortAndGetSteps(); return list of arrays in steps in the sorting process

- The MergeSortAlgorithm class extends the SortingAlgorithm class to implement merge sort algorithm

Attributes	Methods
	+ MergeSortAlgorithm(array: ArrayUtil): constructor method. This passes the array to the superclass (SortingAlgorithm<T>) constructor + sort() : override method to sort array by merge sort algorithm. It calls the mergeSort method with the first and last index of array - mergeSort(left: int, right: int): a recursive function that sorts elements in the underlying array. - merge(left: int, mid: int, right: int): to merge two sorted sub-arrays + getTimeComplexity(): overridden method returns the time complexity of the merge sort algorithm

- The SelectionSortAlgorithm class extends the SortingAlgorithm class. It implements selection sort algorithm

Attributes	Methods
	+ SelectionSortAlgorithm(array: Array Util): constructor method.This passes the array to the superclass (SortingAlgorithm<T>) constructor + sort() : override method to sort array by selection sort algorithm

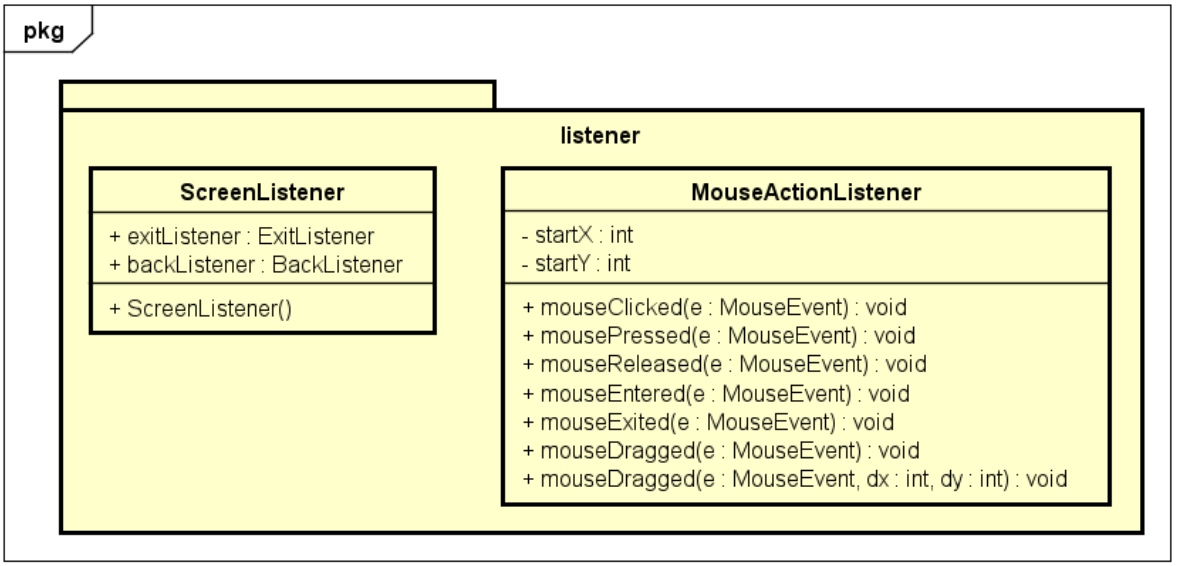
- Class InsertionSortAlgorithm: extends the SortingAlgorithm class. It implements the insertion sort algorithm recursively.

Attributes	Methods
	+ InsertionSortAlgorithm(array: ArrayUtil): constructor method.This passes the array to the superclass (SortingAlgorithm<T>) constructor + partlySort(n:int, arr: ArrayUtil): sorts the array partially by recursively sorting smaller subarrays and inserting the approximate elements in the correct position. + sort(): call partlySort() method to sort the entire array

- Class ShellSortAlgorithm: extends the SortingAlgorithm class. Its implements the shell sort algorithm

Attributes	Methods
- n: represents the size of the array to be sorted - gaps: an array of integers that determine the gaps used in the algorithm	+ ShellSortAlgorithm(array:ArrayUtil) : the constructor initializes the gaps and array variables, and calculates the gaps using the insertion sort algorithm. The InsertionSortAlgorithm is instantiated to sort the gaps array and the sorted gaps are then sorted in the gaps variable in reverse order. + sort(): implements the Shell Sort algorithm. It uses a nested loop to iterate over the gaps and performs insertion sort on subarrays of the main array.

4. Package listener



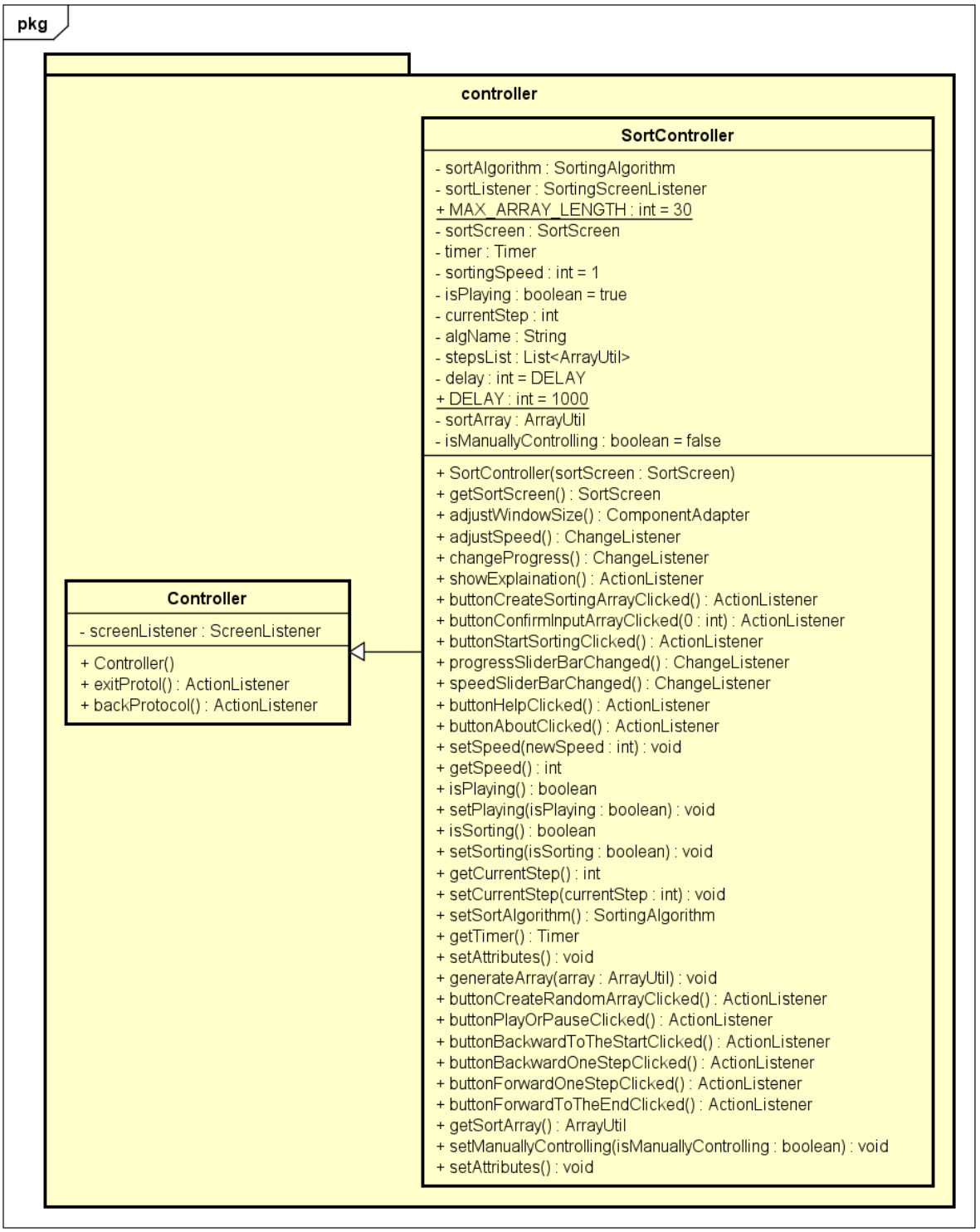
- Class MouseActionListener: extends MouseAdapter class

Attributes	Methods
- startX: variable to keep track of the X coordinate values at which the mouse was initially pressed - startY: variable to keep track of the Y coordinate values at which the mouse was initially pressed	+ mouseClicked(e: MouseEvent): invoked when the mouse button has been clicked on a component + mousePressed (e: MouseEvent): invoked when the mouse button has been pressed on a component + mouseReleased(e: MouseEvent): invoked when the mouse button has been released on a component + mouseEntered(e: MouseEvent): invoked when the mouse enters a component + mouseExited(e: MouseEvent): invoked when the mouse exits a component + mouseDragged(e: MouseEvent, dx: int, dy: int) + mouseDragged(e: MouseEvent): overridden from MouseMotionListener. Invoked when a mouse button is pressed on a component and then dragged. It calculates the distance moved (dx,dy) since the last mouse drag event and calls the custom mouseDragged(e: MouseEvent, dx: int, dy: int) method with these distances. After the call, it updates the start position to the current position

- The ScreenListener class is defined with instance of ExitListener and BackListener classes to handle actions performed on 'Exit' and 'Back' buttons:
    - The ExitListener class implements the ActionListener interface to handle 'Exit' button click events. In the actionPerformed method, a confirmation dialog is displayed. If the user chooses "OK", the program's window is disposed of, effectively closing the program.
    - The BackListener class also implements the ActionListener interface to handle 'Back' button click events. In the actionPerformed method, a new instance of the HomeScreen class is created and made visible. Then, the program's window is disposed of.

Attributes	Methods
+ exitListener: an instance of the ExitListener class + backListener: an instance of the BackListener class	+ ScreenListener(): constructor method to initialize the exitListener and backListener variable

5. Package Controller



- The Controller class acts as an intermediary between the screen and the ScreenListener

Attributes	Methods

- screenListener: an instance of ScreenListener class	+ Controller(): the constructor initializes a new ScreenListener object + exitProtol(): handle actions related to ‘Exit’ button + backProtocol(): handle actions related to ‘Back’ button

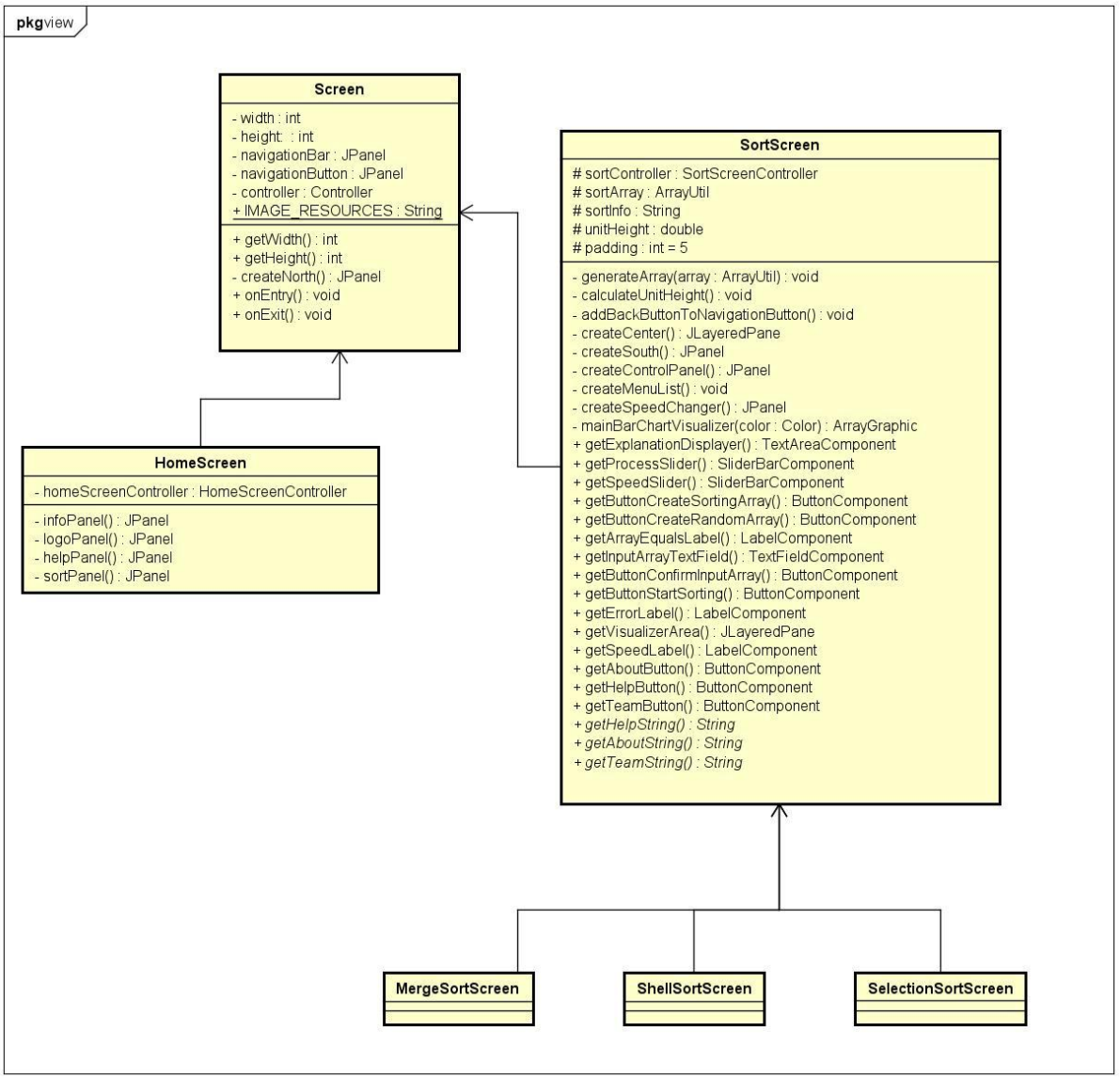
- The SortController class inherits from Controller class and acts as intermediary between SortScreen and the underlying logic or data components of the application

Attributes	Methods
- sortAlgorithm: an instance of SortingAlgorithm - sortListener: an instance of SortListener + MAX_ARRAY_LENGTH : the maximum length of an array is 300 - sortScreen: an instance of SortScreen - timer: an instance of Timer - sortingSpeed: an integer attribute to store the sorting speed - isPlaying: return true if the sorting animation is currently playing - isSorting: return true if the sorting process is currently ongoing - currentStep: represents the current step in the sorting process - algName: name of the algorithm - stepsList: List of current array in each step of the sorting process + DELAY = 1000 - delay: delay between steps of the animation. - sortArray: the array to be sorted - isManuallyControlling: to know if the user is manually controlling the sorting	+ SortController(sortScreen: SortScreen): the constructor method + getSortScreen(): returns the SortScreen associated with this SortController + setAttributes(): set ‘algName’ attribute to the name of sort algorithm + adjustWindowSize(): returns a ComponentAdapter object that adjusts the window size of the SortScreen based on certain conditions + adjustSpeed(): returns a ChangeListener object that adjusts the sorting speed based on the value of a JSlider component + changeProgress(): return an ChangeListener object that handle changes in the progress of the sorting process + showExplaination(): toggles the visibility of an explanation display based on button clicks + buttonCreateSortingArrayClicked(): handles button clicks for creating a sorting array by inputting array + buttonConfirmInputArrayClicked(): handles button clicks for confirming an input array +generateArray(array: ArrayUtil): generate a new random array if sortArray is null, otherwise set sortArray to the provided array + buttonCreateRandomArrayClicked(): handle ‘Random’ button to create an radom array + buttonStartSortingClicked(): handles button clicks for starting the sorting process + progressSliderBarChanged(): handles changes in the process slider bar + speedSliderBarChanged(): handles changes

animation, or letting it automatically run.	in the speed slider bar + buttonPlayOrPauseClicked(): handles play/pause button + buttonBackwardToTheStartClicked(): handles backward button to go to the start of the sorting process + buttonBackwardOneStepClicked(): handles backward button to go back one step in the sorting process + buttonForwardOneStepClicked(): handles forward button to go forward one step in the sorting process + buttonForwardToTheEndClicked(): handles forward button to go to the end of the sorting process + buttonHelpClicked(): handles ‘help’ button clicks for displaying a help window + buttonAboutClicked(): handles ‘about’ button clicks for display an about window + buttonTeamClicked(): handles button clicks for display a team window + setSpeed(): sets the sorting speed to the specified value + getSpeed(): return the current sorting speed + isPlaying(): checks whether the sorting animation is currently playing + setPlaying(): set the playing state and manually control of the sorting animation + isSorting(): checks whether the sorting process is currently ongoing + setSorting(): sets the sorting state + getCurrentStep(): returns the current step in the sorting process + setCurrentStep(): returns the current step in the sorting process + setSortAlgorithm(): sets the sorting algorithm to be used + getSortAlgorithm(): return the currently sorting algorithm + getTimer() + getSortArray(): retrieve the current array + setManuallyControlling()

6. Package view

The View package plays a crucial role in the application's user interface, providing different screens for users to navigate and interact with. The Screen class serves as the base class, while HomeScreen and SortingScreen subclasses provide specialized screens. The subclasses of SortingScreen further refine the functionality for specific sorting algorithms.



6.1 Class Screen: abstract class, generalization of HomeScreen and SortScreen classes

Attributes	Methods
width: Represents the width of the main screen.	Screen(int width, int height): The constructor of the Screen class. It sets up the initial appearance and behavior of the screen, including the

	navigation bar and buttons.
height: Represents the height of the main screen.	Screen(): An overloaded constructor of the Screen class that sets default values for the width and height of the main screen.
navigationBar: Represents the panel at the top of the screen containing the navigation elements.	getWidth(): Returns the width of the main screen.
navigationButton: Represents the panel within the navigation bar containing navigation buttons.	getHeight(): Returns the height of the main screen.
controller: Represents an instance of the Controller class.	createNorth(): A private helper method that creates and returns a panel representing the navigation bar at the top of the screen. It sets up the navigation elements such as menus, menu items, and buttons.
IMAGE_RESOURCES: Represents the directory path to the image resources used in the application.	

The Screen class serves as the main entry point for your application's user interface. It creates the main screen with a specified width and height.

6.1.1 HomeScreen

Attributes	Methods
homeScreenController: An instance of the HomeScreenController class, which handles the logic and actions related to the home screen.	`infoPanel()`: creates and returns a panel for displaying information about the application
	logoPanel(): creates and returns a panel for displaying the application logo and name
	`helpPanel()`: creates and returns a panel for displaying help and about buttons
	`sortPanel()`: creates and returns a panel for displaying the sorting algorithms

6.1.2 SortScreen

6.1.2.1 Attributes

Name	Explaination
sortController	An instance of the `SortController` class, which handles the logic and actions related to sorting algorithms.
sortArray	An instance of the `ArrayUtil` class, which represents the array being sorted.
sortInfo	A string that holds the information or welcome message displayed on the sorting screen.
unitHeight	A double value representing the height of each unit in the graphical representation of the sorting array.
padding	An int variable represents the amount of padding or space to be added around the bars in the bar chart visualization. The value of padding is set to 5 which means that there will be a padding of 5 pixels added around each bar in the visualization.

6.1.2.2 Methods

Methods	Explanation
SortScreen()	Initializes the sorting screen by setting up the layout and components
generateArray(ArrayUtil array)	A method for generating or setting the sorting array. If the `sortArray` attribute is null, it generates a new `ArrayUtil` object with a random array. Otherwise, it sets the `sortArray` attribute to the provided `array` parameter.

calculateUnitHeight()	Calculate the unit height based on the maximum value in the sorting array. It sets the `unitHeight` attribute accordingly.
addBackButtonToNavigationButton()	Adds a back button to the navigation button panel.
createCenter()	Creates and returns the center panel of the sorting screen.
createSouth()	Creates and returns the south panel of the sorting screen.
createControlPanel()	Creates and returns the control panel for controlling the sorting animation
createMenuList()	Creates and adds menu buttons to the visualizer area
createSpeedChanger()	creates and returns the speed changer panel.
createInfoPanel()	creates and returns the information panel.
mainBarChartVisualizer(Color color)	creates and returns the main bar chart visualizer component.

6.1.2.3 Abstract methods

`getHelpString()`, `getAboutString()`, and `getTeamString()` are abstract methods that should be implemented in subclasses. They are used to provide specific help, about, and team information related to each sorting algorithm.

- Class MergeSortScreen: represent for screen of merge sort

Attributes	Methods

- Class SelectionSortScreen: represent for screen of selection sort

Attributes	Methods

- Class ShellSortScreen: represent for screen of shell sort

Attributes	Methods

OOP Techniques

- Inheritance:
  - In the package algorithm, three types of sort algorithms inherit from the SortingAlgorithm class.
  - In the package controller, HomeScreenController and SortScreenController classes inherit from the Controller class.
  - Most of our GUI components inherit from its respective parent class in the JSwing module.
- Abstraction:
  - We use abstract class SortingAlgorithm to manage MergeSortAlgorithm, SelectionSortAlgorithm, InsertionSortAlgorithm and ShellSortAlgorithm classes.
  - Screen is the abstraction of HomeScreen, SortScreen. SortScreen is also the abstraction of MergeSortScreen, ShellSortScreen, SelectionSortScreen.
- Aggregation:
  - We use aggregation between Screen/Controller classes and component's and component.util's classes
- Polymorphism
  - We use polymorphism for upcasting elements type like Integer, Double,.. to our generalized Element and vice versa.
- Composition
  - We use composition between ArrayUtil and Element.

References

- https://docs.oracle.com/en/java/
- https://docs.oracle.com/javase%2F7%2Fdocs%2Fapi%2F%2F/javax/swing/package-summary.html

Ideas:

- VisualGO: https://visualgo.net/en
- Algorithm Visualizer: https://algorithm-visualizer.org/