OOP MINI-PROJECT REPORT

GROUP: 8, DS-AI

1. ASSIGNMENT OF MEMBERS

Pham Quang Hieu - 20194432	**Nguyen Vu Thien Trang – 20194459**	**Nguyen Van Thanh Tung - 20190090**
MainScreen.java MainScreenController.java MergeSort.java SortController.java HelpMenu.java helpText.txt	SelectionSort.java SortController.java MainScreen.fxml SortScreen.fxml ElementShape.java Record demo video Design presentation Slides	ShellSort.java SortAlgorithm.java Displayable.java SortController.java ElementShape.java SquareShape.java BarShape.java MainScreenController.java Check, fix code, feedback for other tasks of partners

2. MINI-PROJECT DESCRIPTION

2.1 PROJECT OVERVIEW

The mission of our project is to build an application which visualizes three sorting algorithms, namely selection sort, merge sort and shell sort.

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

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

2. The array's size used for visualization has to be in between 5 and 8.

2.2 DESIGN REQUIREMENTS.

- On the main menu: title of the application, 3 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 should ask for confirmation before closing.

- In the demonstration:
+) A button for creating the array: 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 have to show clearly each step of the sorting
+) A back button for user to return to main menu at any time

- For a better understanding and visualization, we have added two more feature in our application:
+) Reset, back, next and skip button: instead of letting the application automatically run the visualization user can manually switch between steps of the process to see clearly what happened.
+) 1 more bonus visualize form (bar form within which each array's element is displayed as a column in a bar chart), in the two algorithms Shell sort and Selection sort to help the user track the process more easily.

2.3 USE CASE DIAGRAM EXPLANATION

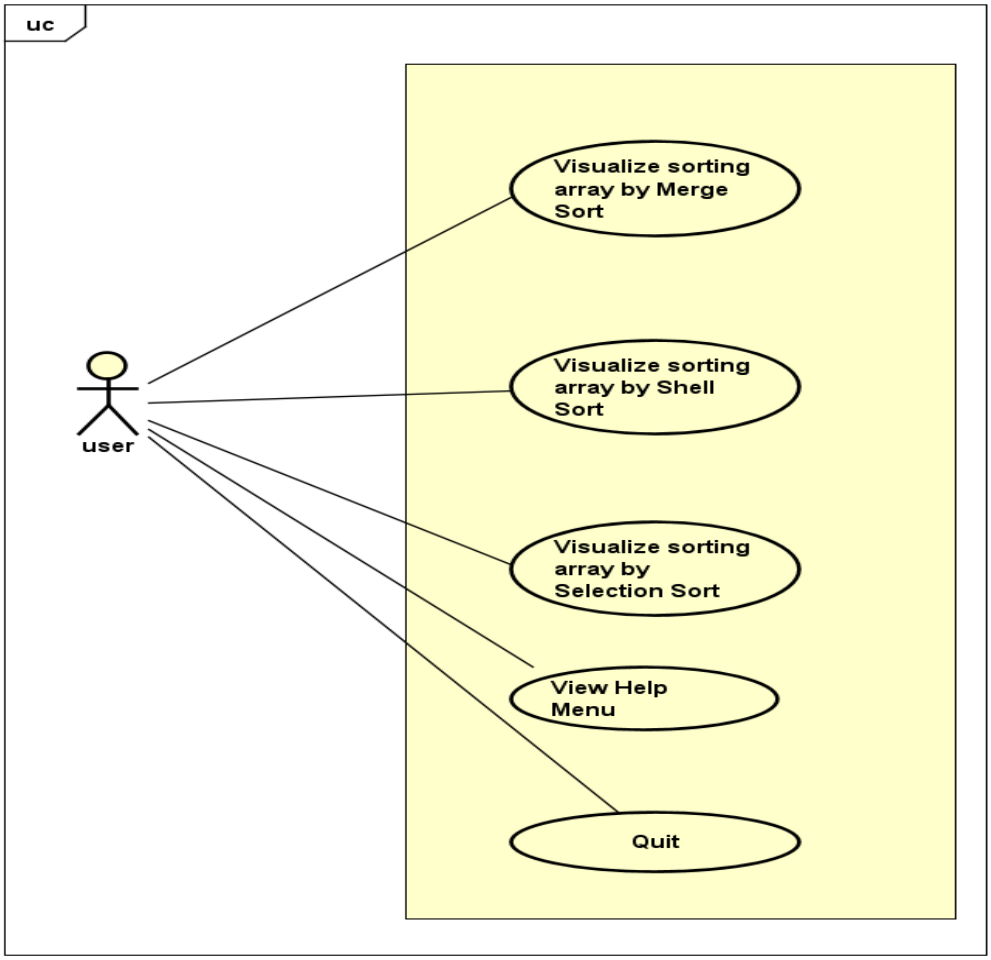

Figure 1. Use Case diagram

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

To be more specific:

- In the first 3 visualize use case, the application will:

1. Take all user command from the GUI such as: whether he want to customize/randomize his array, which array size he want to create, which form of visualization does he want
2. Notice the user if there is anything wrong with his 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 he wants to do the visualization manually or automatically.
5. Run/Reset the visualization based on user command.

- View help menu use case: Show the user manual and application restriction to the user under.
- Quit use case: First ask the user whether he wants to leave/ continue staying in the app. Then based on his decision, close/remain the current window.

3. OOP DESIGN IDEA EXPLANATION

3.1 GENERAL DESIGN

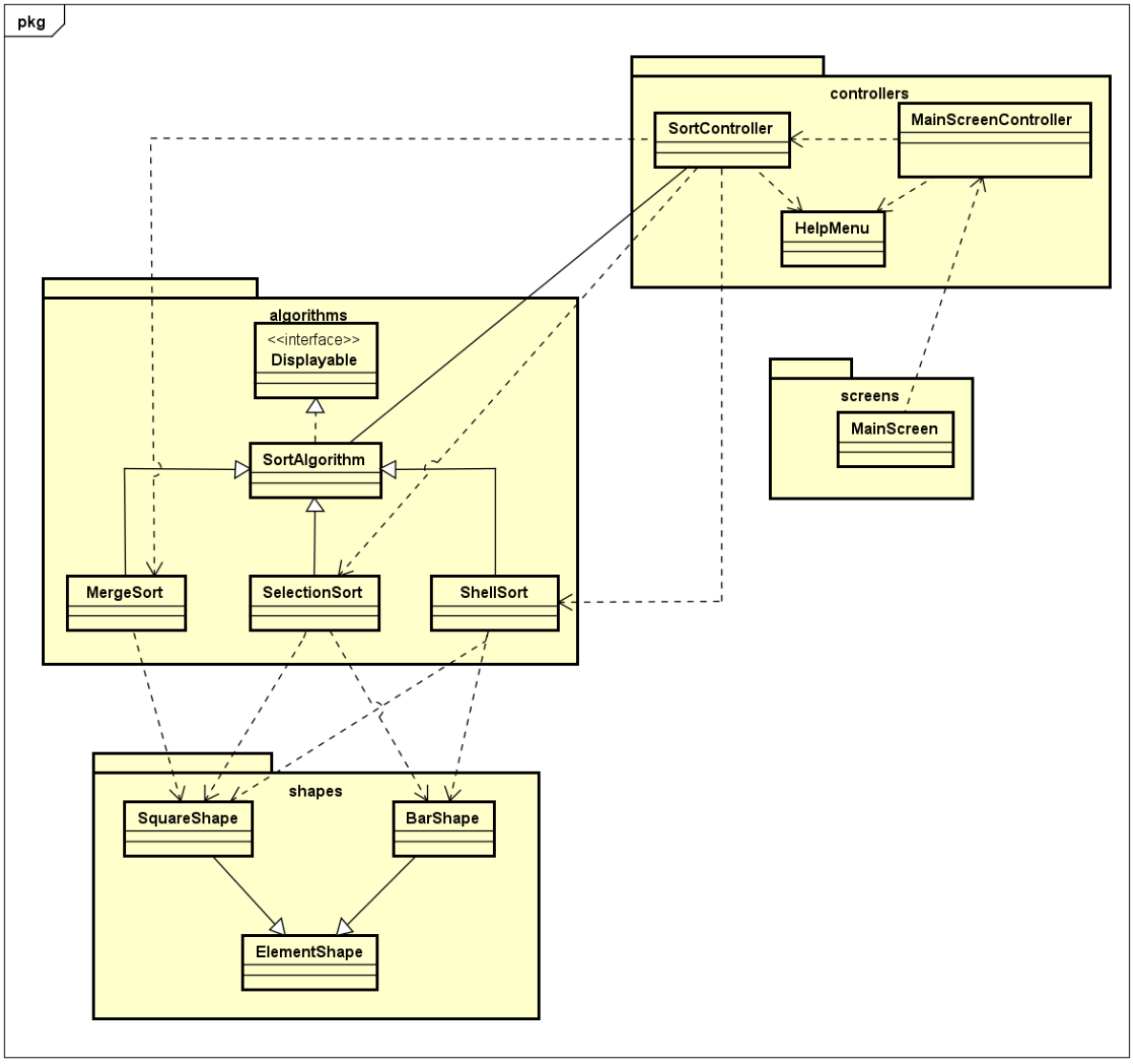

Figure 2. General class diagram.

Packages:

+ Package “algorithms”: store 3 sorting algorithms for visualizing, and an abstract class “SortAlgorithm” to store common operations, attributes between sorting algorithms.

+ Package “controllers”: store all the screen controllers.
+ Package “screens”: store the main screen class of the application
+ Package “shapes”: store the abstract class “ElementShape” and 2 derived shapes, namely SquareShape and BarShape.

3.2 PACKAGE DETAILS

3.2.1 SCREENS PACKAGE

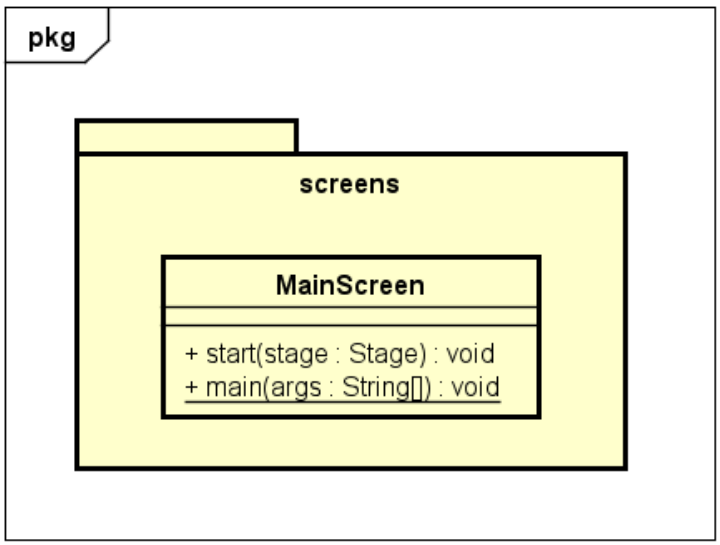

Figure 3. screens package

This package is used for managing the main screen of the application.

For asking the user before exit we have to override the method setOnCloseRequest of the stage used for the main screen.

3.3.2 CONTROLLERS PACKAGE

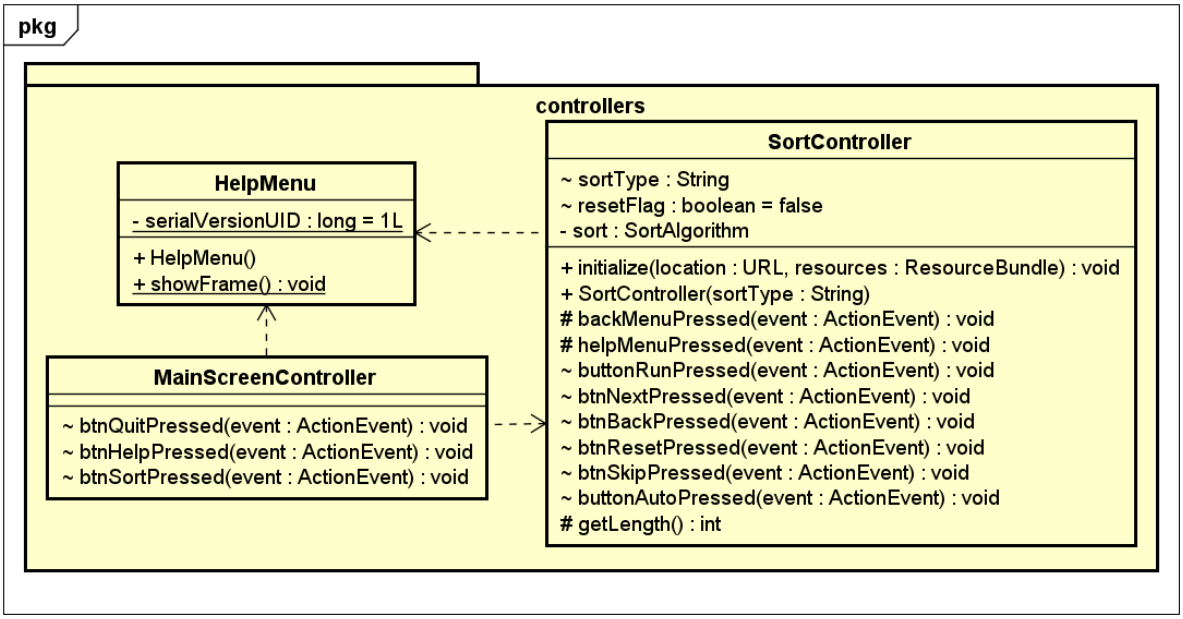

Figure 4. controllers package

MainScreenController:
This controller is used for managing the main screen of the application, where the user can choose a type of algorithm for visualization, view help menu, or quit the app.

+ To ask before the user quit the app, I have to get the current stage,
use the method setOnCloseRequest(), and then pass a new instance of class EventHandler with an overridden “handle” method to show a YES_NO Swing dialog box.

+ btnSortPressed(): switch the current scene to sort screen, then create a new instance of SortController with the input parameter “sortType”

HelpMenu: implemented in JavaSwing to show the application’s user manual. There is a static method for creating a new instance of this class whenever needed.

SortController: This controller is created for managing all the JavaFX components and their corresponding action that we created in the fxml file, because the UI for all sort type visualizers is the same.

Attributes:

+ String sortType: take the name of the sort algorithm which the user wants to visualize from the main screen.
+ boolean resetFlag: used to reset and stop the "Auto mode" when the user clicks the reset button.
+ SortAlgorithm sort: this variable used for upcasting the corresponding Sort algorithm to perform implemented visualization action.

Methods:

+ getLength(): a separated method used for taking user choice for array's length from the UI. This will be called inside "buttonRunPressed".
+ buttonRunPressed(): get all the user input from the UI, handle input exceptions if the user chooses "customize" mode. Create value for the variable "sort" with desired algorithm constructor based on "sortType".
+ This part is where we apply polymorphism technique. All specific sort algorithms are upcasted to the abstract class SortAlgorithm which implements the interface "Displayable" containing all required abstract functions for visualization tasks.
+ Those abstract functions' implementation will be called in the btnNextPressed, btnSkipPressed, btnBackPressed and btnAutoPressed.

3.3.3 ALGORITHM PACKAGE

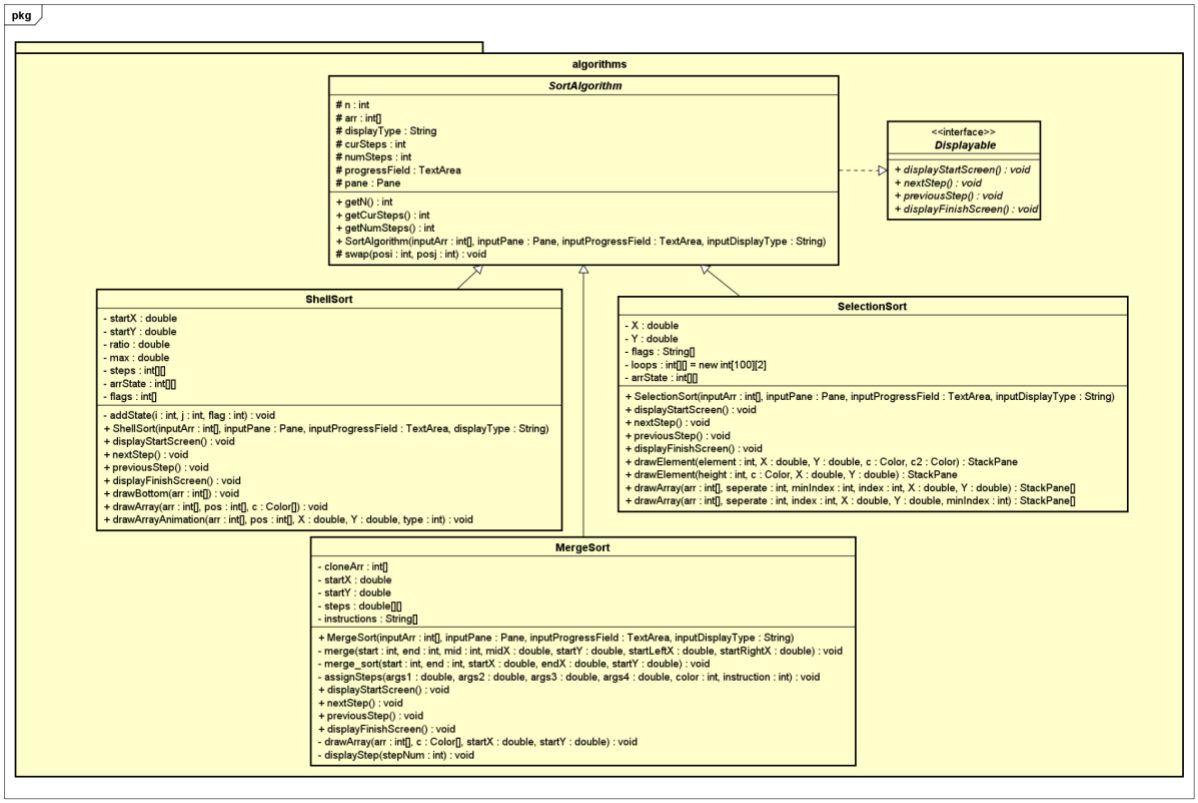

Figure 5. algorithm package

Main idea: First, we sort the array beforehand, then store the corresponding stage of the sort algorithm. Finally, we will use those stored stages to display on the screen. For every algorithm, we have to implement the 4 abstract methods from the interface Displayable to do the visualizing task.

- Displayable:
    + displayStartScreen: show the start screen.
    + nextStep: show the next visualization step.
    + previousStep: show the previous visualization step.
    + displayFinishScreen: show the screen when the visualization finishes.

SortAlgorithm.java	
This class is created for storing common characteristics, operations between the three sort algorithms.	Attributes: int n: size of original array int[] arr: original array

	Pane pane: area displaying array TextArea progressField: text box displaying progress explanation String displayType: shape display type (Nodes or Bars) int curSteps: current step int numSteps: total number of steps

ShellSort.java	
Attributes	Methods
n: size of input array startX, startY: coordinate to illustrate array flags[]: each element indicates a corresponding user's instruction. ratio: the ratio to illustrate barform fit with size of the window.	addState(int i, int j, int flag): +) i, j: index of the two array's elements being taken into comparison. +) flag: id of user's instruction. drawArray(arr[], pos[], c1, c2): illustrate the state of array arr[] when selecting elements at pos[] with color filled by c1, color of text is c2 drawArrayAnimation(arr[], pos[], double X, double Y, type): illustrate the animation when we process the sorting step at the index pos[], depending on which form of visualization is chosen (node or bar). drawBottom(): draw the number label under each bar shape, for a better visualization.

MergeSort.java	
Attributes	Methods
int[] cloneArr: a copy without reference of the original array. int[][] steps: array to encode all the steps of the visualization process. double startX: left limit of the display area. double startY: Y-coordinate for the first row of elements. String[] instructions: store predefined user's instructions for each step.	merge_sort() and merge() function: used for sorting the array. assignSteps(): an encoding function integrated in merge_sort() and merge() function to encode each step of visualization then store that into the "steps" array. drawArray(): visualize the array at a specific position on the screen. displayStep(): visualize a specific step of visualization. This function will be called inside the 4 abstract methods when needed.
SelectionSort.java	
Attributes	Methods
double X, Y: coordinate String[] flags: progress status at each step StackPane[][]: staticNodes, staticBars: node-shape, bar-shape array at each step int[][] loops = new int[100][2]: store {array status, index of smallest element} at swap steps int[][] arrState: store array status at each step.	drawElement(), drawArray(): visualize elements and arrays drawElement(): return an ElementShape. drawArray(): return an array of ElementShape. SelectionSort(): sort the original array (integer array). During the sorting process, store the shaped-array, integer-array, progress status.

3.3.4 SHAPES PACKAGE

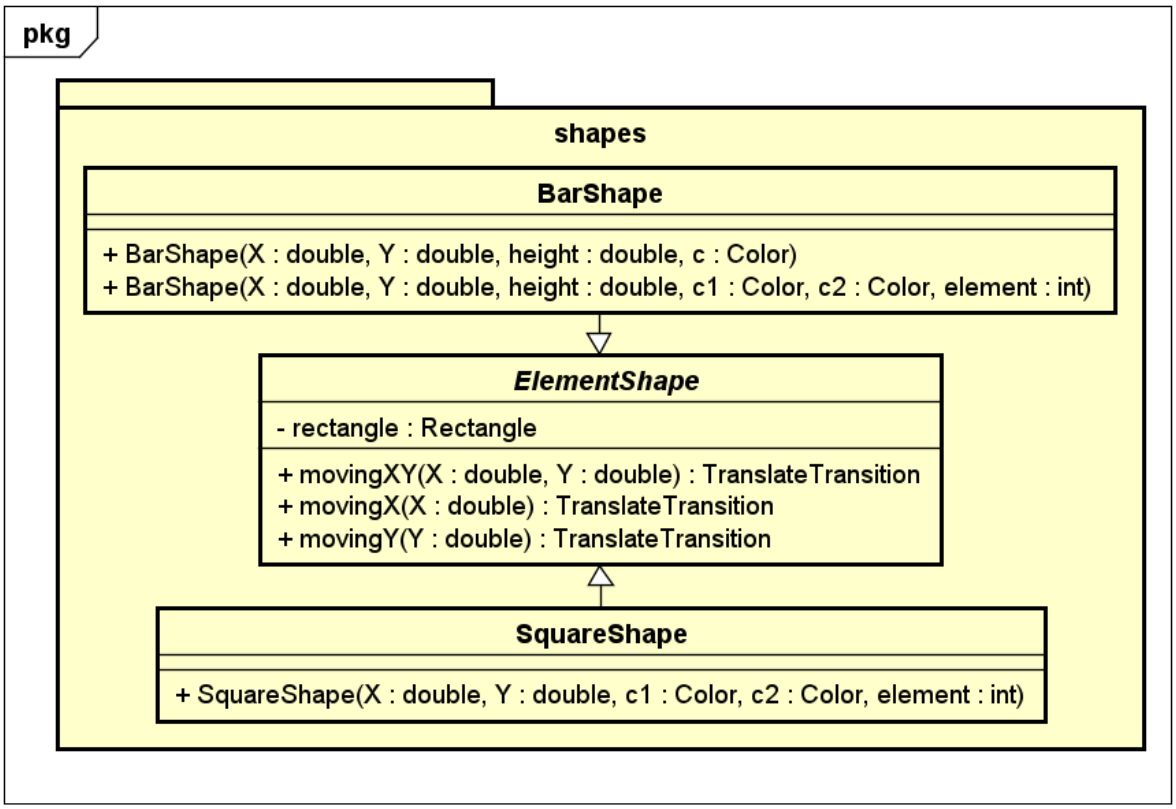

Figure 6. shapes package

Our customized Shape for representing array elements in this application is ElementShape, which directly inherits from the StackPane class of JavaFX.

- ElementShape: Abstract class for representing array elements in this application is ElementShape, which directly inherits from the StackPane class of JavaFX (in order to stack the text label on top in the node form representation).
- SquareShape: class for visualizing the node form.
- BarShape: class for visualizing the bar form.

Method:

- movingX(X): return TranslateTransition object that can move the stackpane along the coordinate X by X unit in 500ms.
- movingY(Y): return TranslateTransition object that can move the stackpane along the coordinate Y by Y unit in 500ms.
- movingXY(X, Y): composition movement of movingX then movingY

TABLE OF CONTENTS