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	SelectionSort.java	ShellSort.java
MainScreenController.java	SelectionSortController.java	ShellSortController.java
MergeSortController.java	MainScreen.fxml	ElementShape.java (20%)
SortScreenController.java	SortScreen.fxml	MainScreen.java
HelpMenu.java	ElementShape.java (60%)	Array.java
helpText.txt	Write and fix the report	Check, fix code, give feedback for tasks of partners
Array.java	Record demo video	Contributor to the report (30%).
ElementShape.java (20%)	Design presentation Slides	Report refactor.
Draw class diagrams,	References: for the auto transition mode, I use the idea of chriszq/VisualSortingAlgorithms.	Contributor to the slide.
Write part 2 of the report, colab with others in other parts.		

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

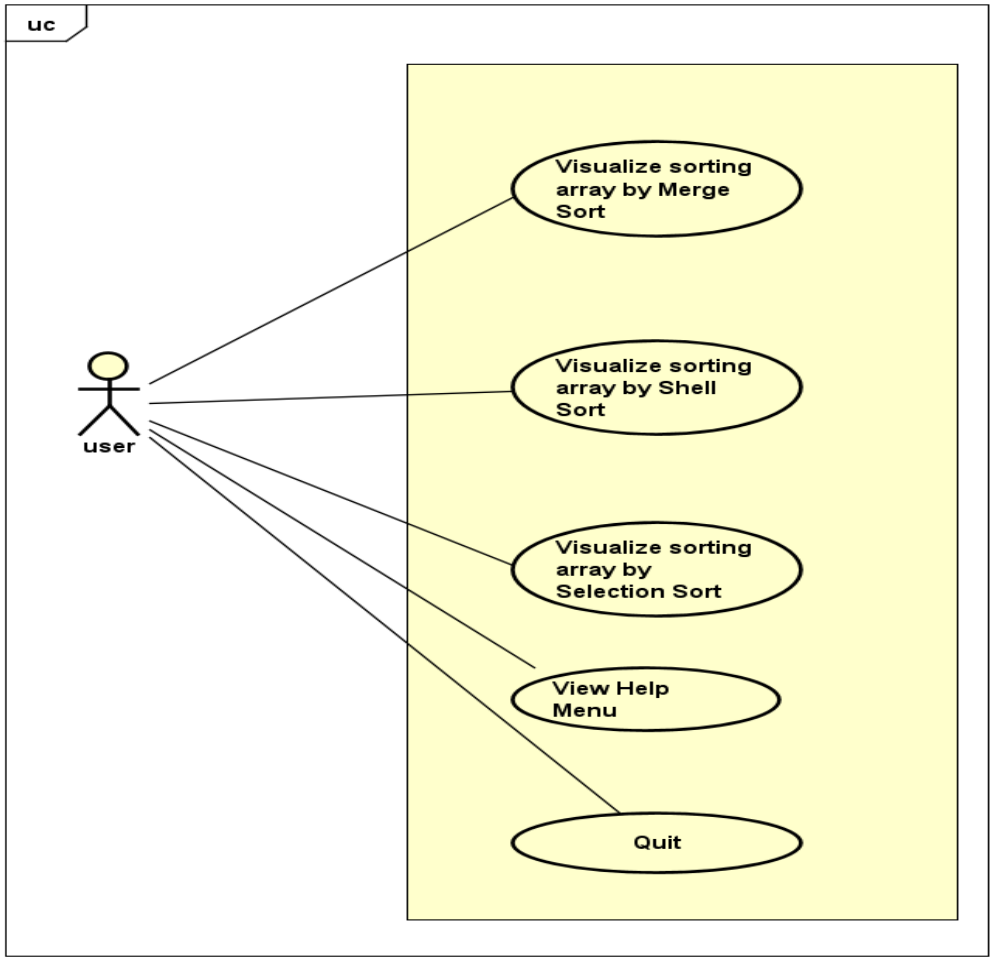

Figure 1. Use Case diagram

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

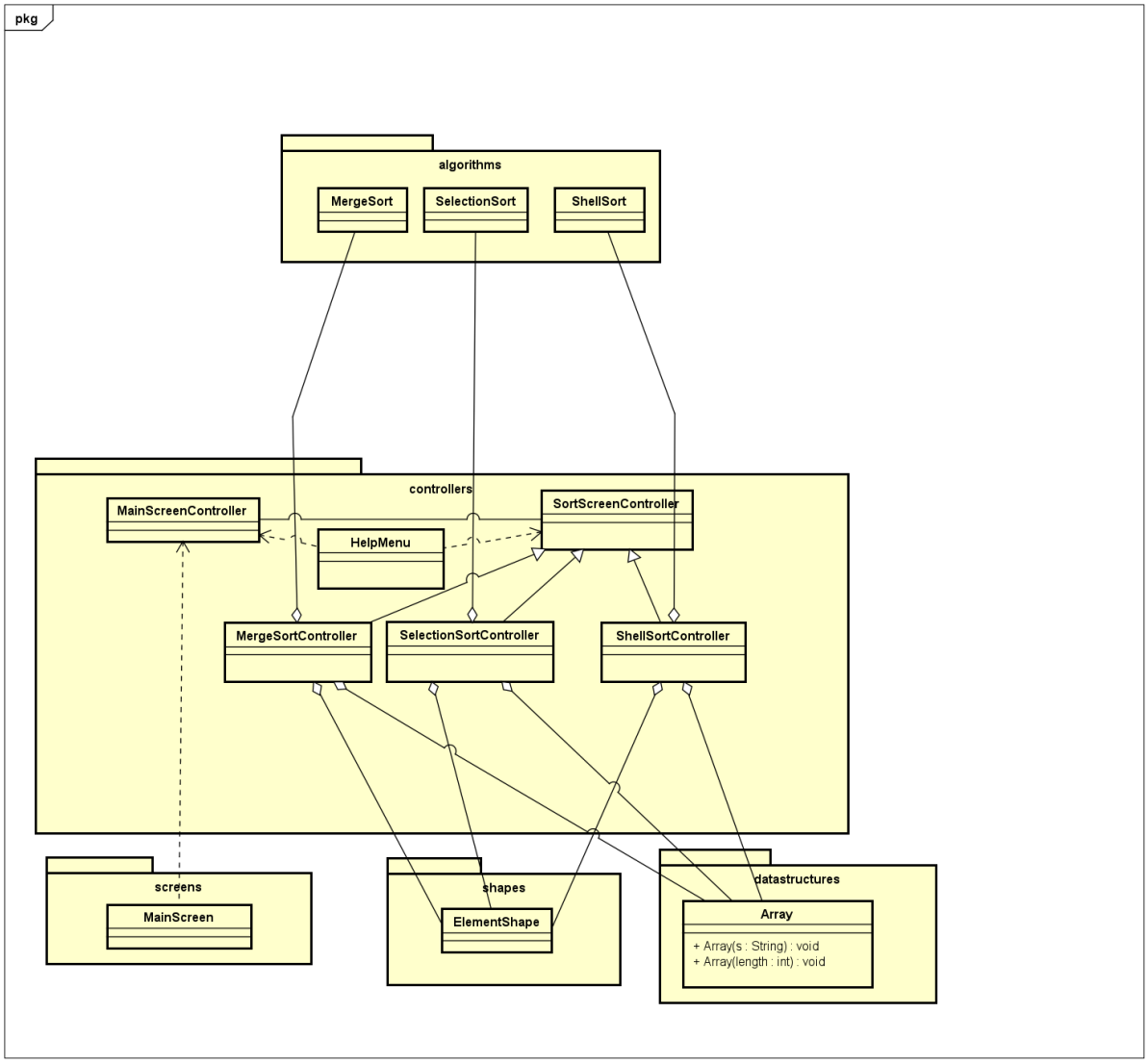

Figure 2. General class diagram.

Packages:

+ Package “datastructures”: store the main data structure (customized array) used throughout this project.

+ Package “algorithms”: store 3 sorting algorithms for visualizing.
+ Package “controllers”: store all the screen controllers.
+ Package “main screen”: store the main screen class of the application
+ Package “shapes”: store the class “ElementShape” which is used for visualization in the GUI.

3.2 PACKAGE DETAILS

3.2.1 DATASTRUCTURES PACKAGE

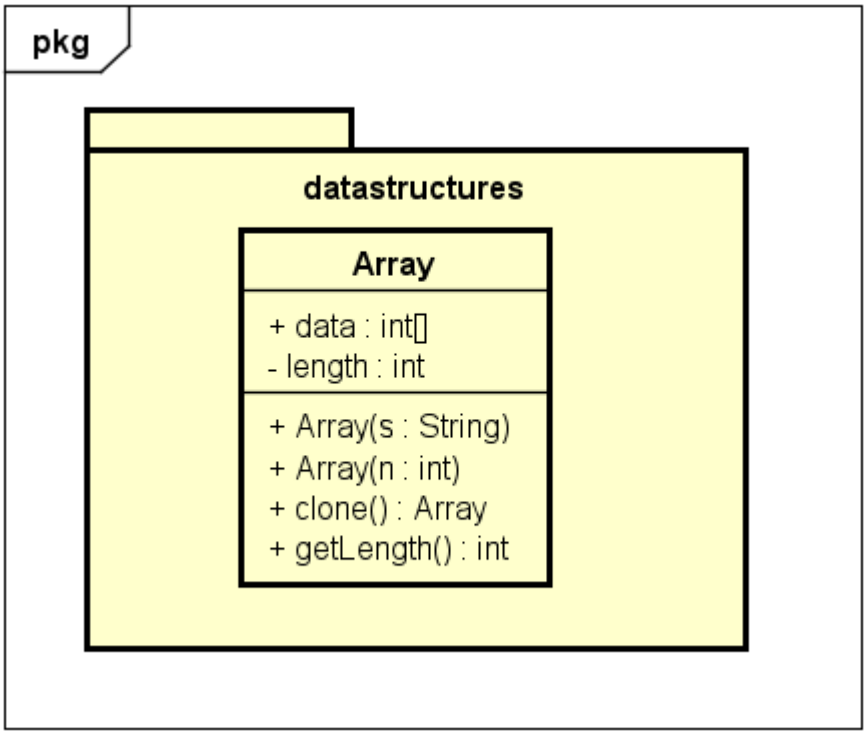

Figure 3. datastructures package

Class Array	
Attribute explanation	Method explanation
data: an usual integer array. length: data's length.	Array(s: String): take the user input when they choose customize mode. This constructor is designed to handle & throw user IO exceptions. clone(): A method to clone the Array data structure without reference. getLength(): return the length attribute.

3.2.2 ALGORITHMS PACKAGE

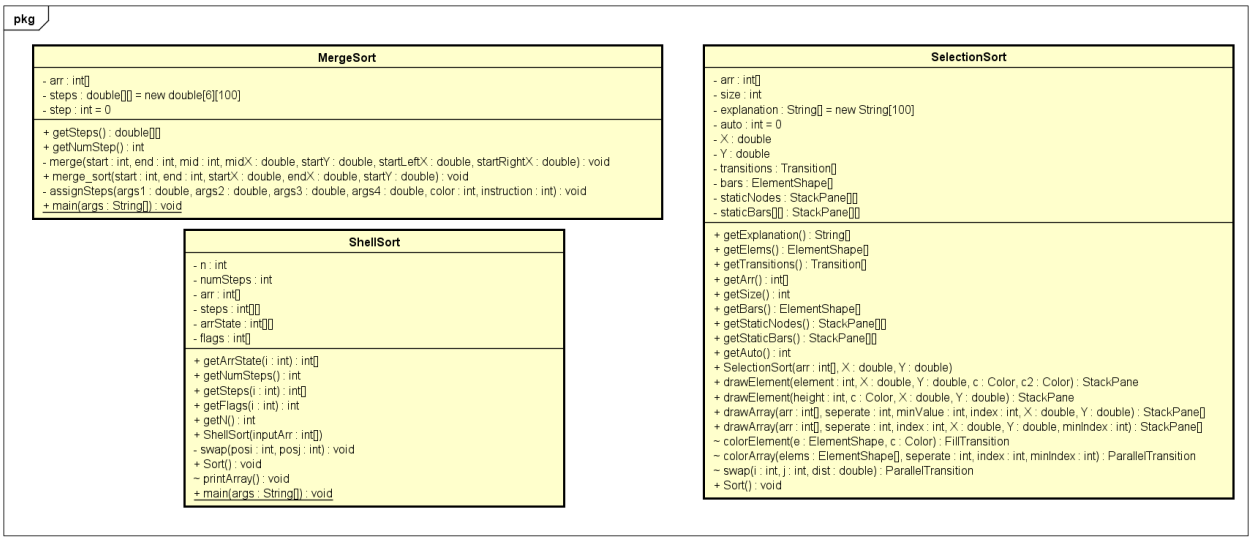

Figure 4. algorithms package

Main idea: Because in our application, we provide both automatic and manual visualizing mode, we decided to sort the array beforehand, then encode the process of visualization into a 2D array named "steps" for easier manipulation in the corresponding controller class. Each column in the array "steps" corresponds to a single step.

Class SelectionSort	
Attributes explanation:	

int[] arr: original array int size: size of array Transition[] transitions: transitions at each step ElementShape[] bars: ElementShape array corresponding to original integer array StackPane[][] staticNodes: Array Node shape at each step StackPane[][] staticBars: Array Node shape at each step String[] explanation: Explanation at each step int auto = 0: number of steps double X: coordinate double Y: coordinate	drawElement(), drawArray(), colorElement(), colorArray(): return the shape of element, array at each step: drawElement(): return an ElementShape. drawArray(): return an array of ElementShape. colorElement(): return a transition which fills the color to an ElementShape. colorArray(): return a parallel transition of filling colors to elements of current array. swap(): swap two elements, return a translation transition. Sort(): sort the original array (integer array) and the encoded array (ElementShape array). At the same time, store the array's shapes, transitions, explanations at each step.

Class MergeSort	
Attributes explanation:	
int[] arr: the original array. int[][] steps: array to encode all the steps of the visualization process. int step: total number of steps of the visualization process.	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. getSteps(): return the array “steps” when called, provided that the merge_sort function has been called before. getNumStep(): return the attribute step.

Class ShellSort	
Attributes explanation:	
n: size of array arr: original array numSteps: number of steps in sorting process steps[][]: steps[i] denotes pair of positions of elements to compare in step i arrState[][]: arrState[i] denotes the state of array after step i flags[]: flag[i] denotes the type of step in the process	getNumSteps(): return number of steps getSteps(i): return selected compare positions getFlags(i): return type of step getN(): return array size Sort(): process the shell sort algorithm printArray(): print array state Constructor: ShellSort(int []): create ShellSort object by passing corresponding array.

3.3.3 CONTROLLERS PACKAGE

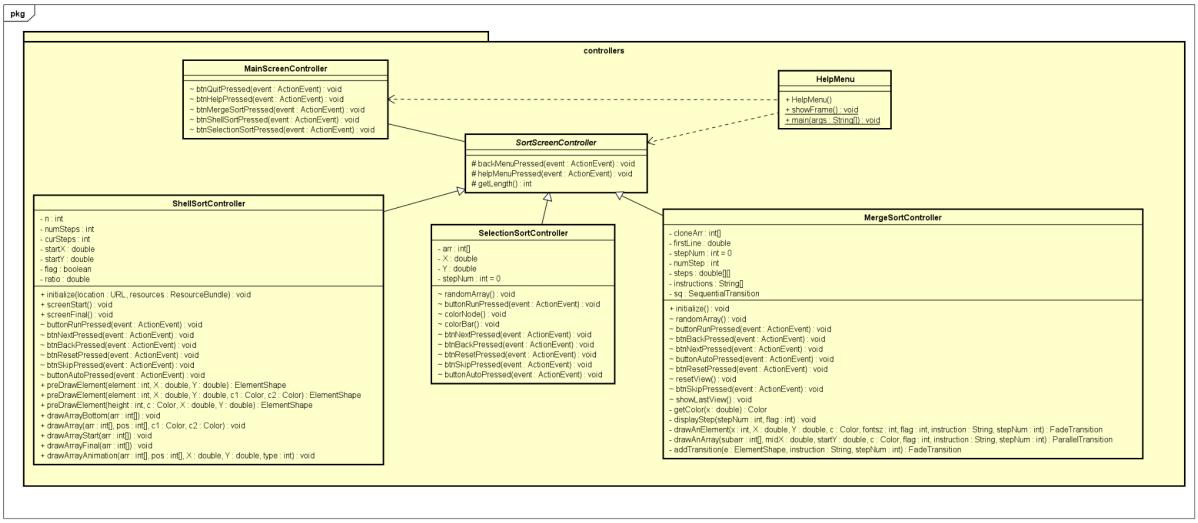

Figure 5. controllers package

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.

- SortScreenController: This controller is created for managing all the JavaFX components that we created in the fxml file, because the UI for all sort type visualizer is the same. There is a method getLength() to return the chosen length option from the user.
  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.
- 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.
- SelectionSortController:
+ Attributes:
- arr int[]: integer array
- double X, double Y: coordinate
- int stepNum: number of steps processed
+ randomArray(): randomize an array.
+ colorNode() and colorBar(): display notations of colors for node shape and bar shape.

MergeSortController:

- Attributes:
+ int [] cloneArr : a copy of the original array (before being sorted)
+ double firstLine: starting Y-coordinate of the visualization
+ int stepNum: a variable to track the current step of visualization
+ int numStep: total number of steps of the visualization
+ String[] instructions: an array to store instructions to describe the instant step of visualization for the user.
  Since the instruction has been encoded in the "steps" array, this array serves as a decode array to translate from a step to a string.
+ SequentialTransition sq: to display all the visualization steps sequentially.
- Method:
+ getColor(): decode the information about the color of each ElementShape on the screen.
+ drawAnElement(): draw an ElementShape onto the display area.
+ drawAnArray(): draw an Array onto the display area.

+ addTransition(): add transition effect to an ElementShape
- Working mechanism: supposed that the user has input the array, by randomize mode or customize mode.
+ In the auto visualization mode: The SequentialTransition (sq) will be called to play. To prevent unwanted bugs during this process, I temporarily disable the back, the next and skip buttons which are shown in the UI as "<, >, >>" correspondingly. After that, I override the method "handle" of the class EventHandler and use the method setOnFinised of the SequentialTransition.
+ In the manual visualization mode: When the user clicks on the ">" next button, The application shows the current step, and then increases the "stepNum" by one until the "stepNum" equals to the total "numStep".

ShellSortController:

- Working mechanism: same as Merge Sort.
- Attributes:
+ n: size of input array
+ numSteps: number of steps in sort process
+ startX, startY: coordinate to illustrate array
+ flag: true if user chooses nodeForm, false if user chooses bar Form
+ ratio: the ratio to illustrate barform fit with size of the window.
- Methods:
+ screenStart(): illustrate state of window before sort process.
+ screenFinal(): illustrate the state of window after sort process.
+ 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 pos[],
depend on the type of sorting step.

3.3.4 SCREENS PACKAGE:

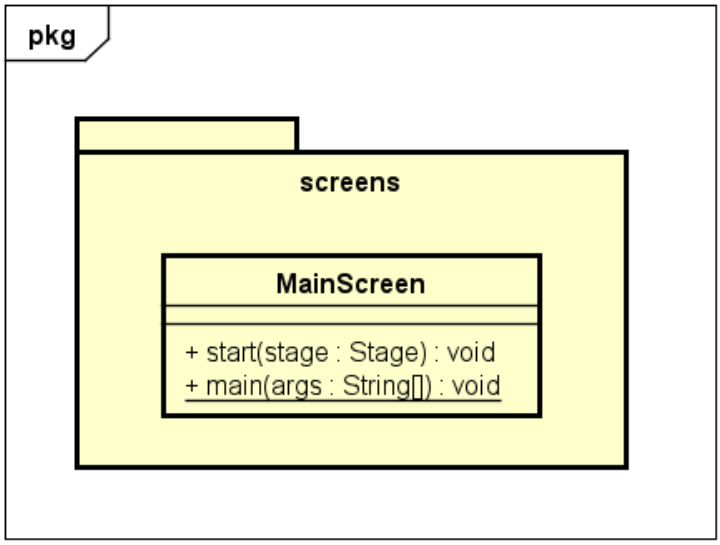

Figure 6. screens package

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

The mechanism for asking the user before quitting is the same as that of the SortScreenController class.

3.3.5 SHAPES PACKAGE

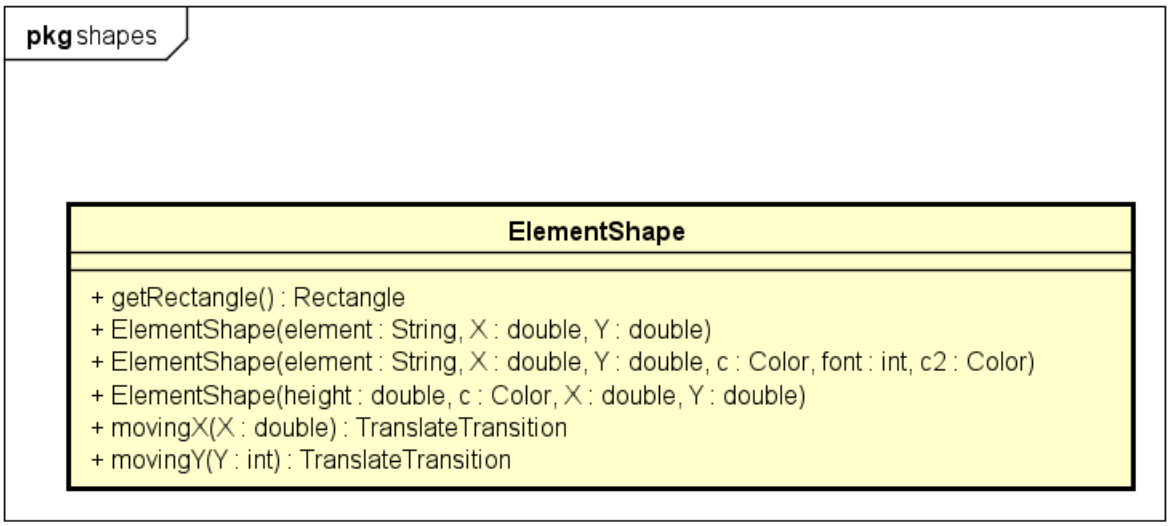

Figure 7. shapes package

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

There are 3 main constructor in this class, representing 3 different shape types which will be used in this app.

There are two types of TranslateTransition function which will be used in shell sort and selection sort visualization process.

Constructor:

- ElementShape(element, X, Y): return object as a stack of a rectangle with the label element inside at coordinate (X,Y) in the window.
- ElemenShape(element, X, Y, c, font, c2): return object as a stack of a rectangle with label element inside at coordinate (X,Y) in the window. Can customize the color and font of the rectangle, label.
- ElementShape(height, c, X, Y): return object as a bar with customized height, color at coordinate (X,Y)

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.

TABLE OF CONTENTS