Vũ Minh Dũng

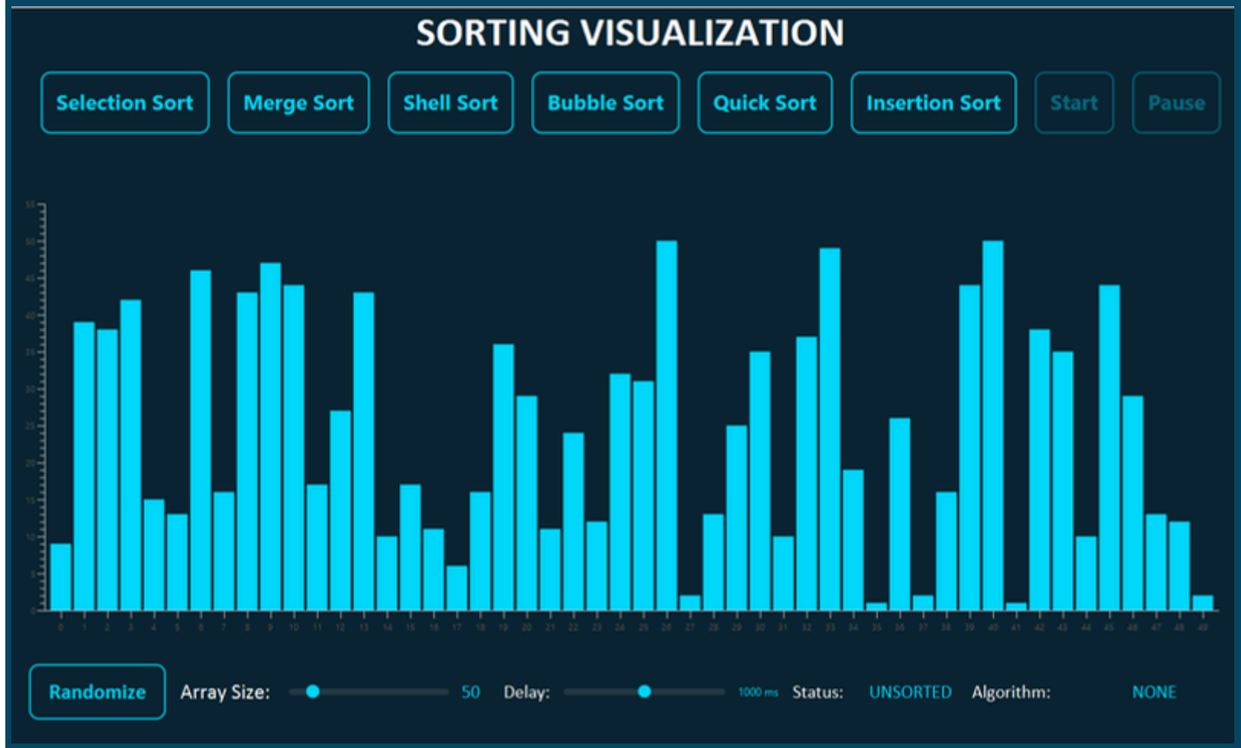
Object-Oriented Programming

Sorting Algorithms Visualizer

Problem Description

1. What is sorting visualizer?

A sorting visualizer is a tool or application that visually demonstrates how different sorting algorithms operate by animating the process of sorting. It's primarily used for educational purposes to help students and developers understand the mechanics and efficiency of various sorting algorithms.

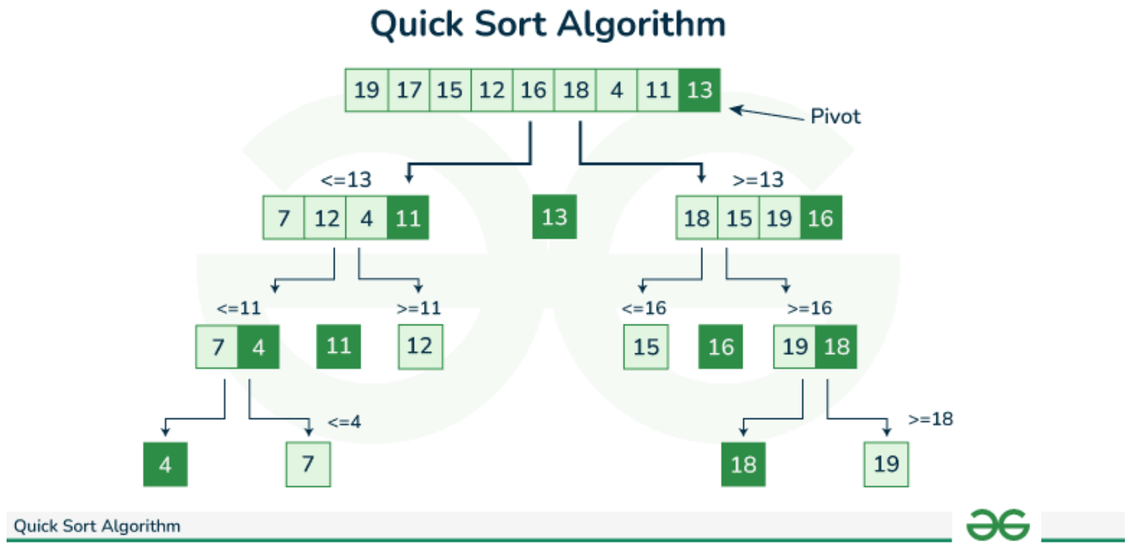


Sorting algorithms

- ## Quick sort

Description: Quick Sort is another divide-and-conquer algorithm that selects a pivot element, partitions the list into two sublists (elements smaller than the pivot and elements larger than the pivot), recursively sorts these sublists, and finally combines them.

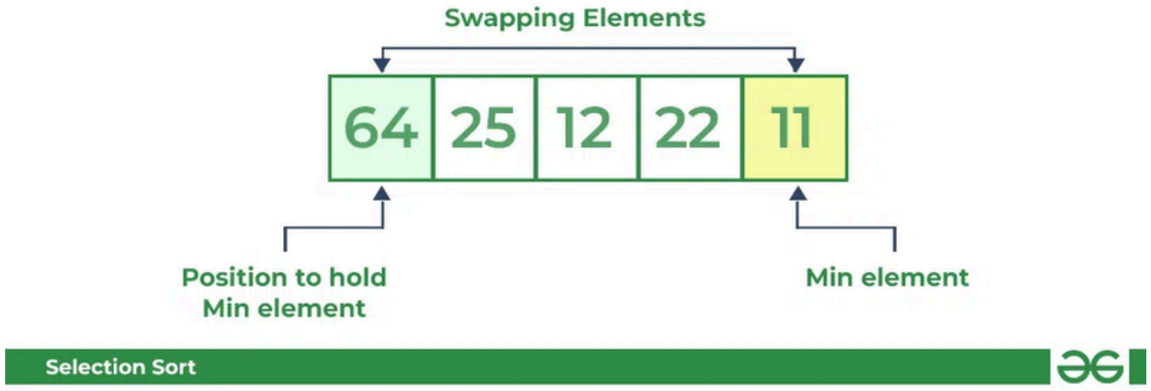
Characteristics: Efficient for large datasets, in-place (requires only a small auxiliary stack), not stable by default (can be implemented as stable with additional effort).



Sorting algorithms

- ## Selection sort

Description: Selection Sort divides the list into a sorted and an unsorted region. It repeatedly selects the smallest (or largest, depending on sorting order) element from the unsorted region and swaps it with the leftmost unsorted element.
Characteristics: Simple implementation, minimal data movement (good for memory usage), but inefficient for large datasets due to its quadratic time complexity.

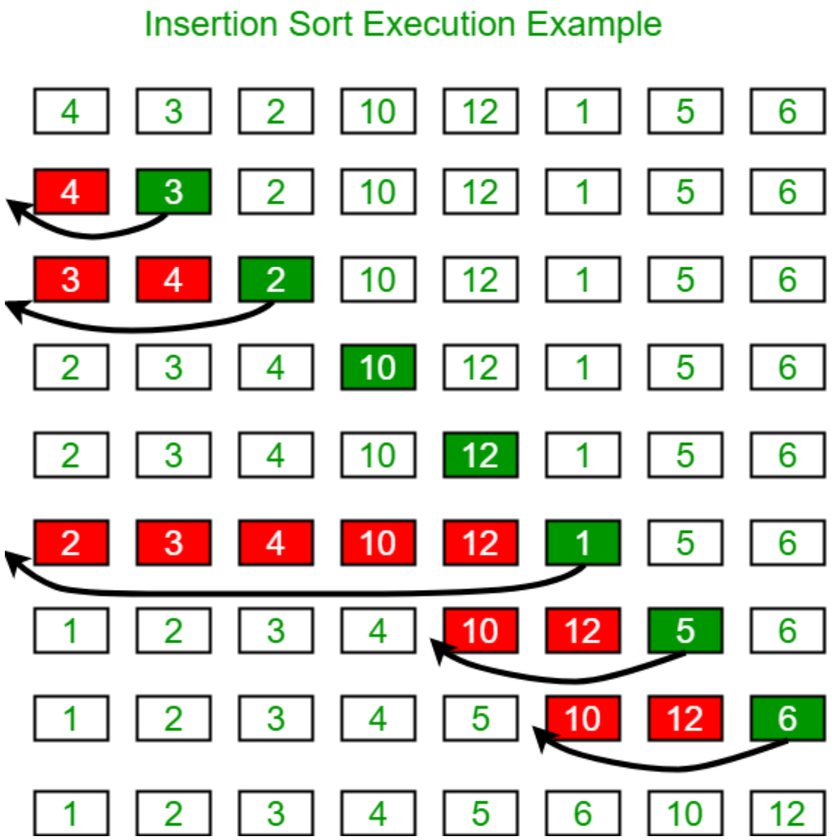


Sorting algorithms

- ## Insertion sort

Description: Insertion Sort builds the sorted list one element at a time by repeatedly taking the next element from the unsorted part and inserting it into its correct position in the sorted part. It shifts the larger elements one position to the right to make room for the inserted element..

Characteristics: Efficient for small datasets or nearly sorted data, stable (preserves the relative order of equal elements), and in-place (requires constant extra space).

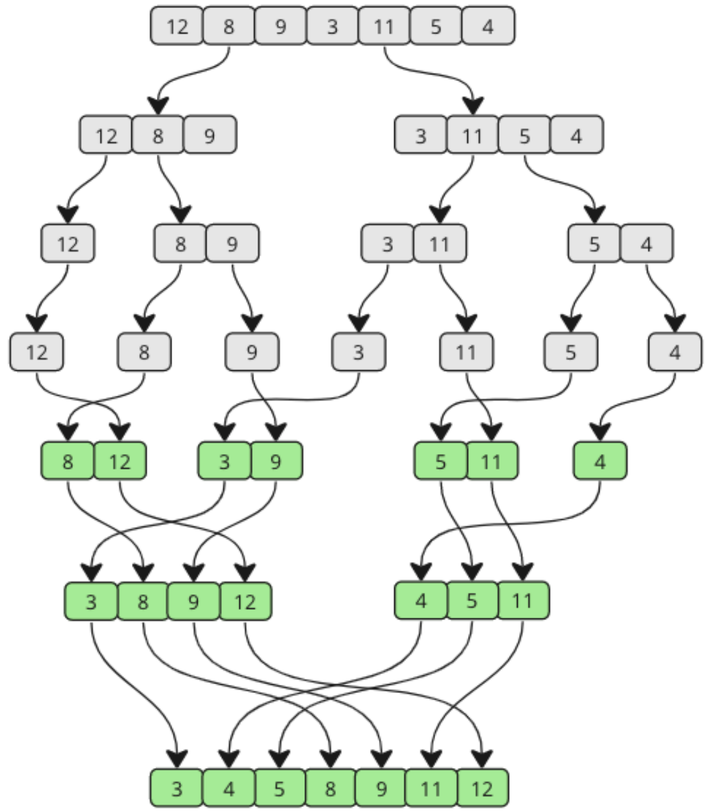


Sorting algorithms

- ## Merge sort

Description: Merge Sort is a divide-and-conquer algorithm that divides the list into smaller sublists, recursively sorts them, and then merges them back into the original list in sorted order. It uses additional space proportional to the size of the input list.

Characteristics: Stable, efficient for large datasets due to its O(nlogn) time complexity, but requires additional space for merging.

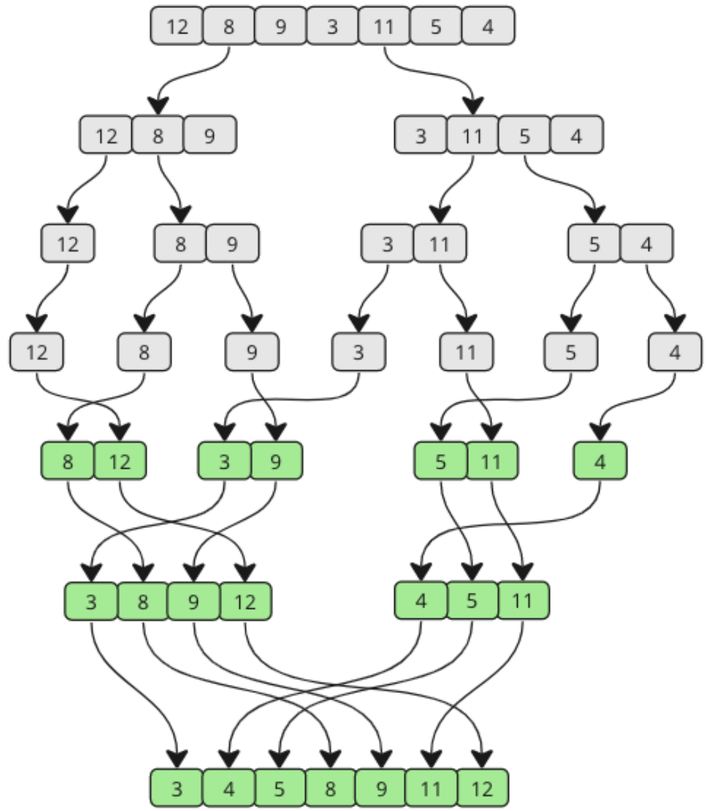


Sorting algorithms

- **Shell sort**

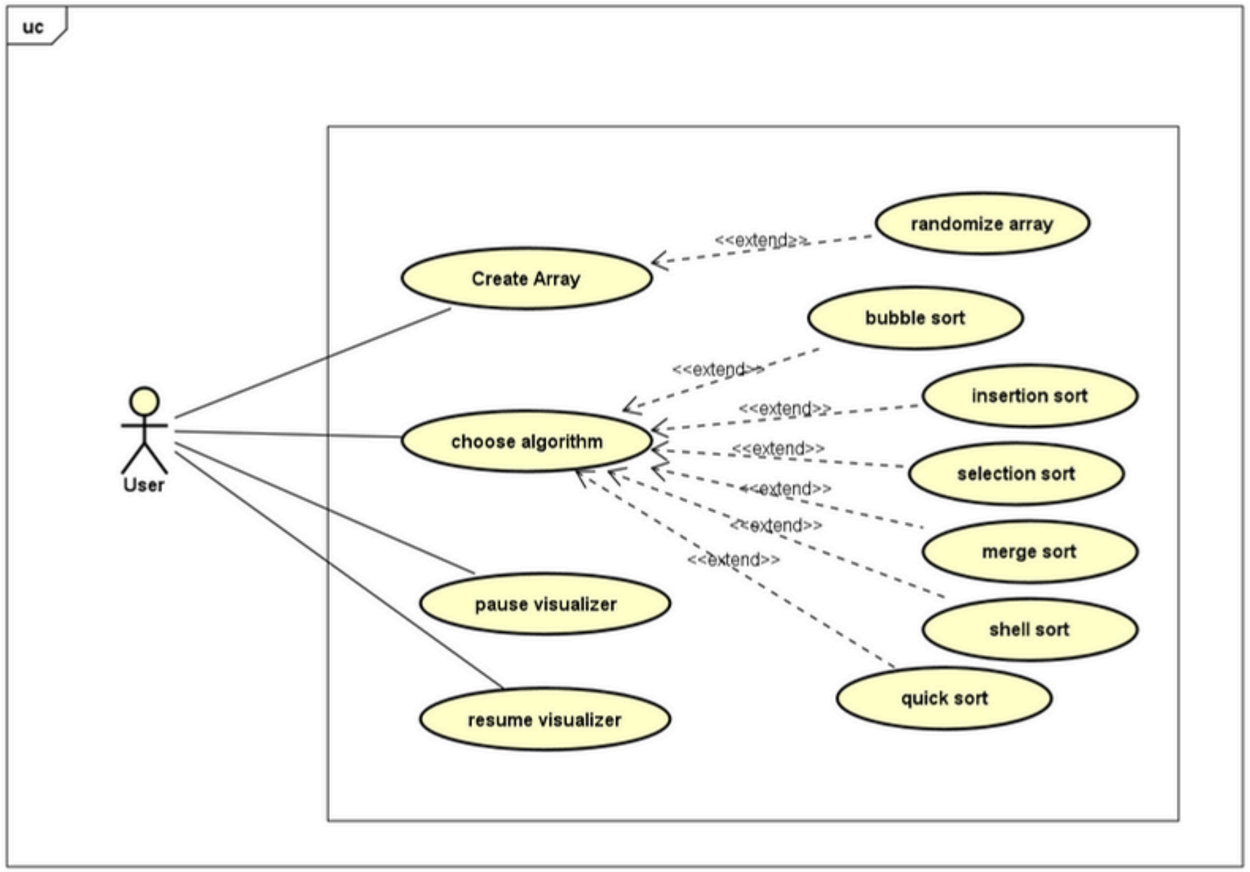
Description: Shell Sort is an extension of Insertion Sort where elements are compared with elements distant from each other (using a gap), rather than adjacent. The gap decreases with each pass until it becomes 1, which then makes the list almost sorted.

Characteristics: Generally better performance than Insertion Sort for medium-sized datasets, not stable, and its efficiency depends heavily on the choice of gap sequence.

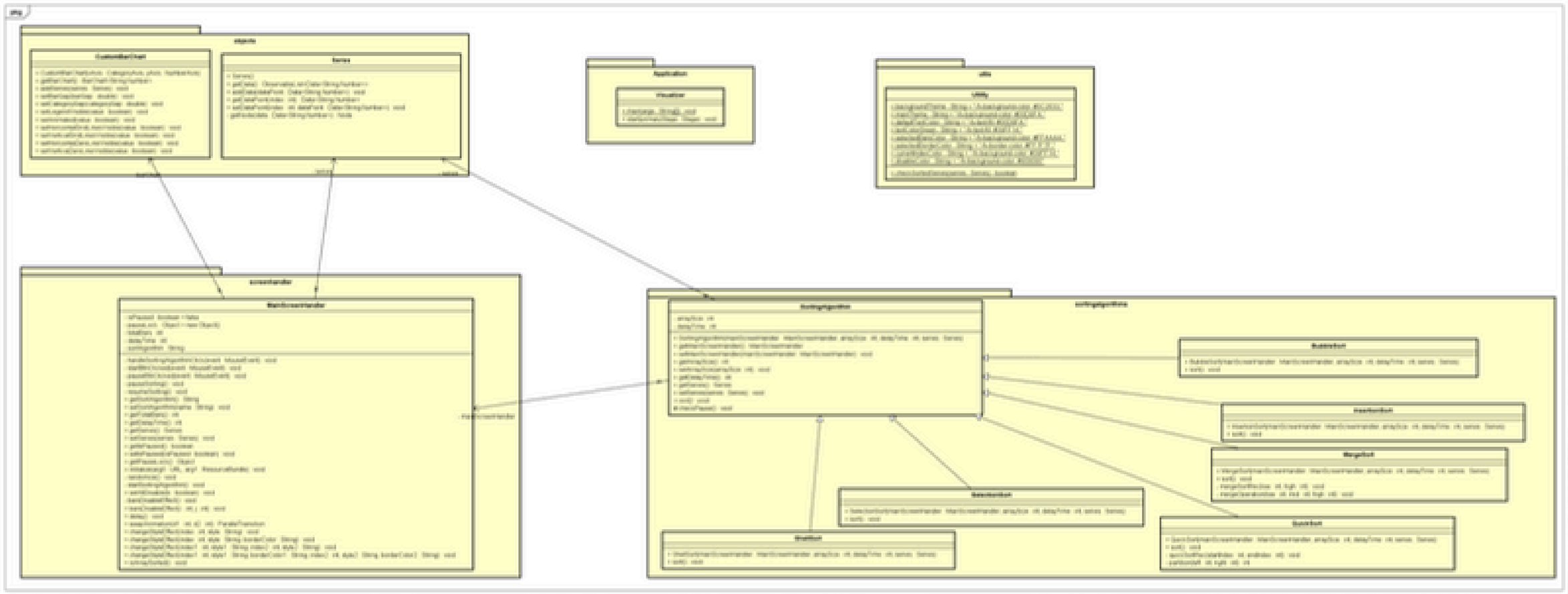


Use case diagram

- create array: allow user to create an randomize array with selected size for visualizer
- choose algorithm: allow user to choose which algorithm for visualizing
- pause visualizer: when the visualizer is running algorithms, the user can pause to see the current step
- pause visualizer: when the visualizer is running algorithms, the user can pause to see the current step



General Class Diagram



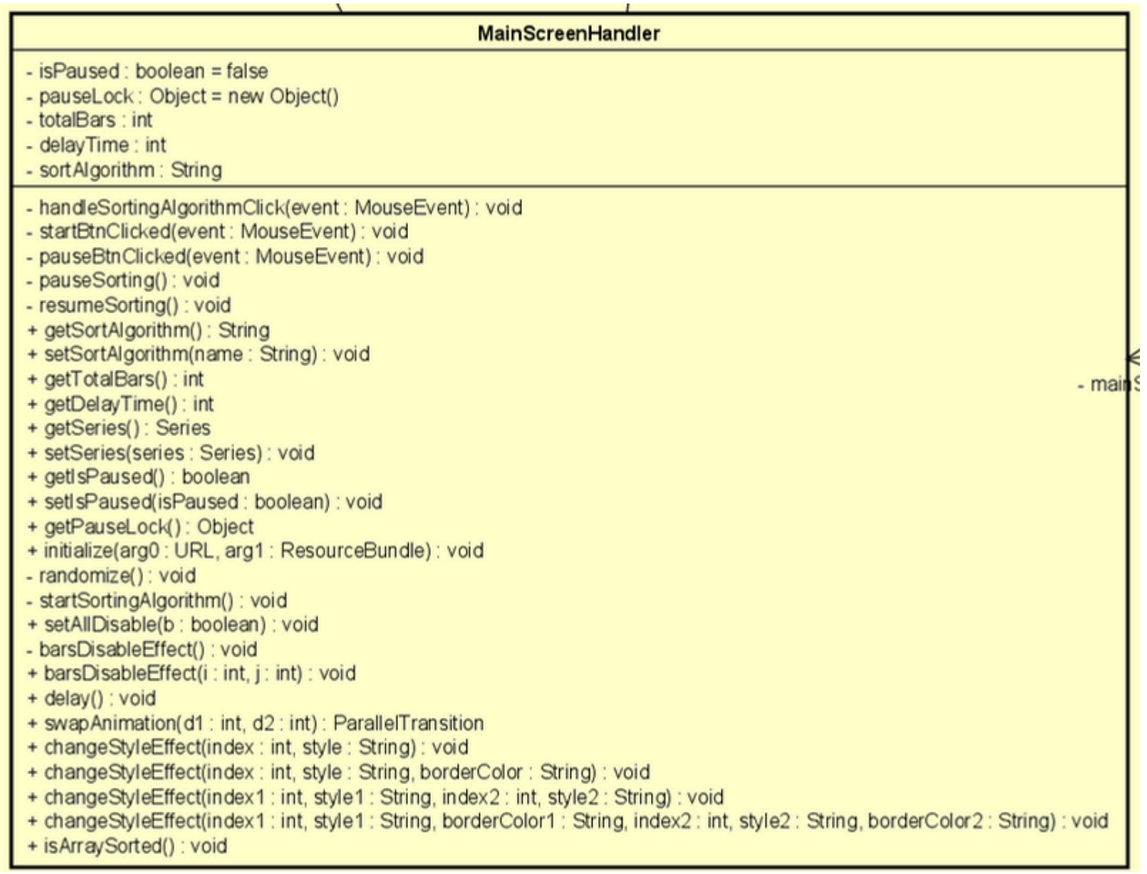
OOP Technical

FEELING

- **Encapsulation**

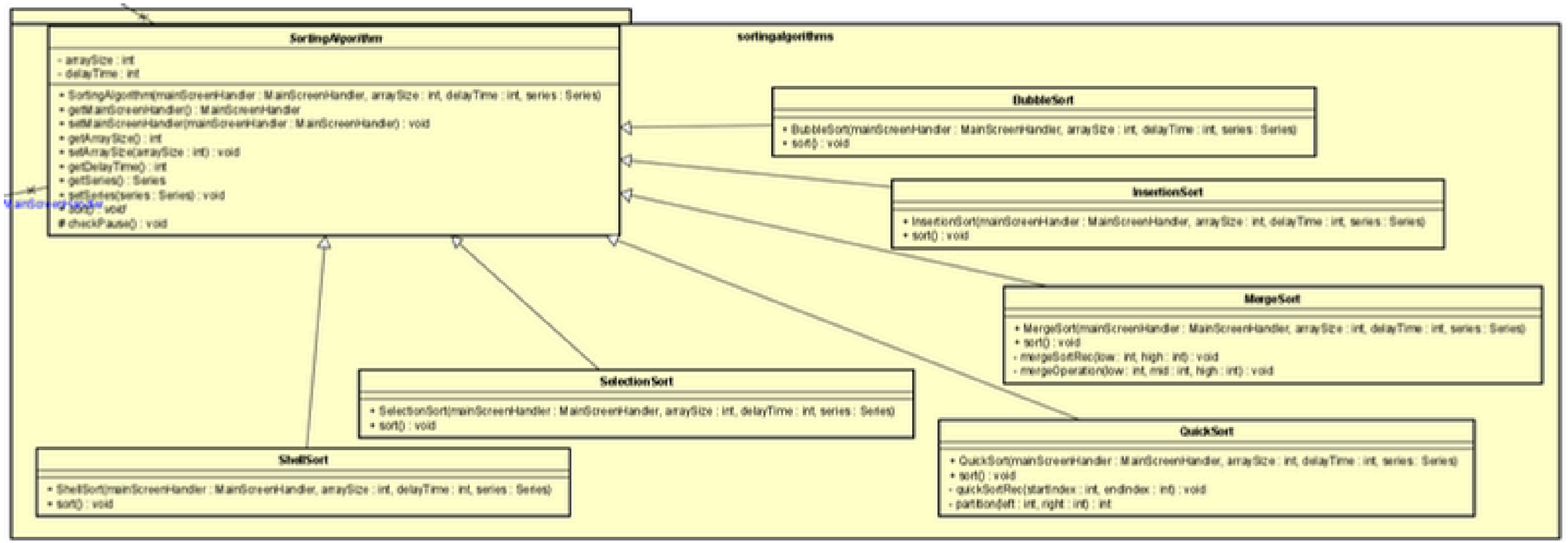
Characteristic: Encapsulation refers to the bundling of data (attributes) and methods (functions) that operate on the data into a single unit called a class. It hides the internal state of an object from the outside world and only exposes a public interface to interact with it.

Significance: Protects an object's internal state from accidental modification, improves code modularity, and promotes reusability



OOP Technical

- **Inheritance**

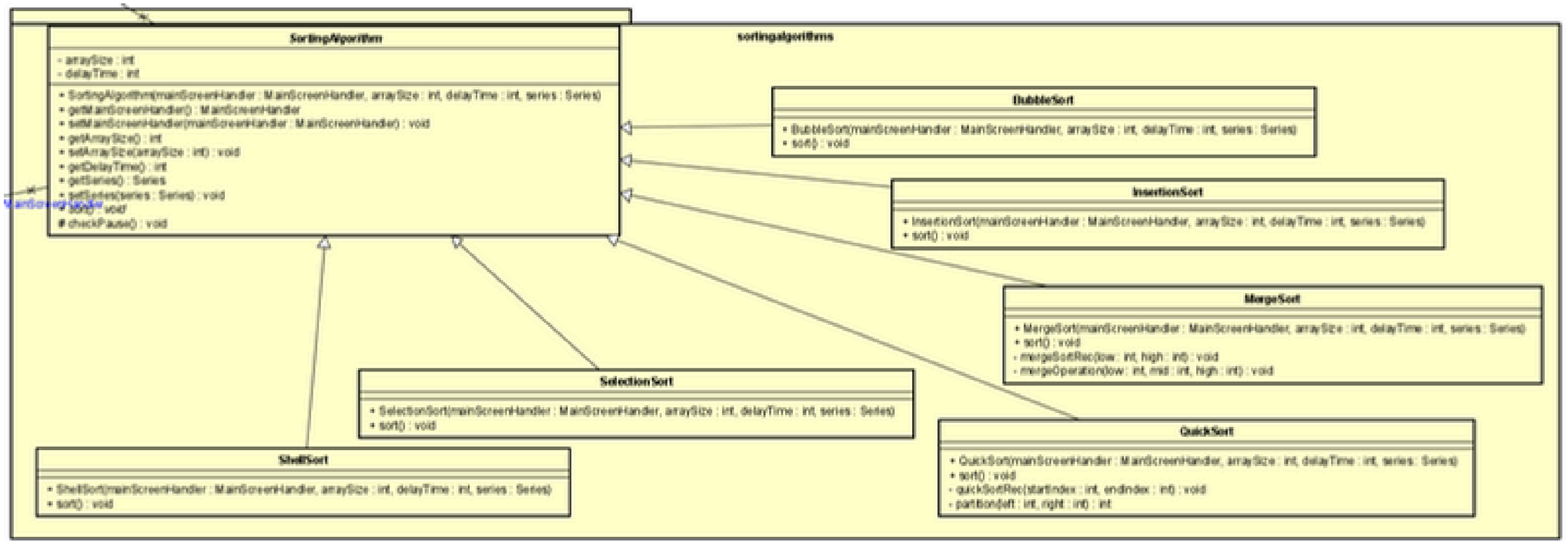


Characteristic: Inheritance allows a class (subclass or derived class) to inherit attributes and methods from another class (superclass or base class). It supports the concept of hierarchical classification.

Significance: Reduces redundancy in code by allowing derived classes to reuse the code of existing classes, facilitates the modeling of real-world relationships, and enhances code organization and readability.

OOP Technical

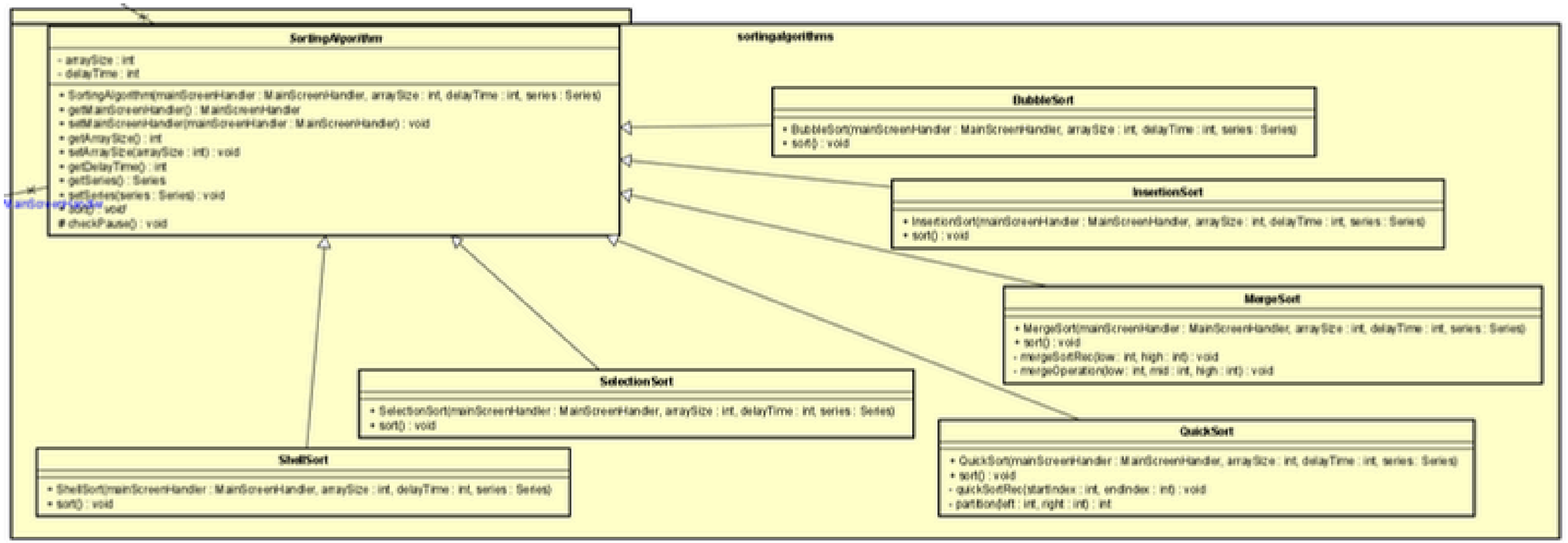
- **Polymorphism**



Characteristic: Polymorphism means the ability of different objects to respond to the same message (method call) in different ways. It allows methods to be written to manipulate objects of a base class to act on objects of its derived classes.

OOP Technical

- **Abstraction**



Significance: Simplifies complex systems by hiding unnecessary details and exposing only relevant information, enhances security by focusing on what an object does instead of how it does it, and supports code reusability and maintenance.

Thanks for your attention

For questions and comments:

Phone

0366032936

Email

vm.dung.learning@gmail.com
dung.vm205179@sis.hust.edu.vn