Purpose?
Decide which of four sort algorithm implementations to include in their library
Questions?
Which implementations run faster (for which kind of inputs)?

Dependent Variables:

- **Execution Time**
- **Number of Swaps** performed by each algorithm

Independent Variables:

- **Sorting Algorithm**
- **Data type**(String, Int, Float, Double)
- **Data Distribution**(is the array almost already sorted, or is in reverse should do more work to sort it)
- **Array size**
- **Input size**(are the number or the strings very long or very short)

Confounding factors:

- **Hardware Differences:** Differences in the hardware used for testing (e.g., CPU speed, RAM)
- **System Performance**: Variability in CPU load, memory availability, and other system activities at the time of execution could affect timing results
- **Measurement Techniques**: How execution time is measured (e.g., using `System.nanoTime()` vs. `System.currentTimeMillis()`) can lead to different levels of accuracy
- **Different type of Java(JDK)** used

Can we randomize something?

- **Array Content:** When generating the arrays to be sorted, you can randomize the values. This means creating random arrays with varying values and distributions
- **Array Sizes:** Instead of testing a fixed set of array sizes, you can randomly select sizes within a predefined range for each experiment

Considerations:

- Warm up the machine/machines before conducting the experiment:
- For calculating the average time should we use the mean or the median?
  - Our data is normally distributed, mean is better
  - Our Data is NOT normally distributed, there are some elements that skew the results(e.g if we do the inserting of the elements in the array with randomization, some values might be very big and others very small), in this case median is better.
  - Display the result by plotting some graphs, to make it look cooler