

DAY 28:

ASSIGNMENT 4:

Task 4: Strategy

Develop a Context class that can use different SortingStrategy algorithms interchangeably to sort a collection of numbers

Answer:

```
import java.util.List;

// SortingStrategy interface
interface SortingStrategy {
    void sort(List<Integer> numbers);
}

// InsertionSort algorithm implementation
class InsertionSort implements SortingStrategy {
    @Override
    public void sort(List<Integer> numbers) {
        for (int i = 1; i < numbers.size(); i++) {
            int key = numbers.get(i);
            int j = i - 1;
            while (j >= 0 && numbers.get(j) > key) {
                numbers.set(j + 1, numbers.get(j));
                j = j - 1;
            }
            numbers.set(j + 1, key);
        }
    }
}
```

```

// MergeSort algorithm implementation

class MergeSort implements SortingStrategy {

    @Override

    public void sort(List<Integer> numbers) {

        mergeSort(numbers, 0, numbers.size() - 1);

    }


    private void mergeSort(List<Integer> numbers, int left, int right) {

        if (left < right) {

            int mid = (left + right) / 2;

            mergeSort(numbers, left, mid);

            mergeSort(numbers, mid + 1, right);

            merge(numbers, left, mid, right);

        }

    }


    private void merge(List<Integer> numbers, int left, int mid, int right) {

        int n1 = mid - left + 1;

        int n2 = right - mid;

        int[] L = new int[n1];

        int[] R = new int[n2];

        for (int i = 0; i < n1; i++)

            L[i] = numbers.get(left + i);

        for (int j = 0; j < n2; j++)

            R[j] = numbers.get(mid + 1 + j);

        int i = 0, j = 0;

        int k = left;

        while (i < n1 && j < n2) {

```

```

        if (L[i] <= R[j]) {
            numbers.set(k, L[i]);
            i++;
        } else {
            numbers.set(k, R[j]);
            j++;
        }
        k++;
    }

    while (i < n1) {
        numbers.set(k, L[i]);
        i++;
        k++;
    }

    while (j < n2) {
        numbers.set(k, R[j]);
        j++;
        k++;
    }
}

// Context class
class SortingContext {
    private SortingStrategy strategy;

    public SortingContext(SortingStrategy strategy) {
        this.strategy = strategy;
    }
}

```

```

public void setStrategy(SortingStrategy strategy) {
    this.strategy = strategy;
}

public void sort(List<Integer> numbers) {
    strategy.sort(numbers);
}
}

public class Main {
    public static void main(String[] args) {
        // Example usage
        List<Integer> numbers = List.of(5, 2, 9, 1, 6);

        // Sorting using InsertionSort
        SortingContext insertionSortContext = new SortingContext(new InsertionSort());
        insertionSortContext.sort(numbers);
        System.out.println("Sorted using InsertionSort: " + numbers);

        // Sorting using MergeSort
        SortingContext mergeSortContext = new SortingContext(new MergeSort());
        mergeSortContext.sort(numbers);
        System.out.println("Sorted using MergeSort: " + numbers);
    }
}

```