import java.util.Arrays;

// Strategy interface

interface SortStrategy {

void sort(int[] data);

}

// Concrete strategy: Bubble Sort

class BubbleSort implements SortStrategy {

public void sort(int[] data) {

int n = data.length;

for (int i = 0; i < n-1; i++) {

for (int j = 0; j < n-i-1; j++) {

if (data[j] > data[j+1]) {

int temp = data[j];

data[j] = data[j+1];

data[j+1] = temp;

}

}

}

}

}

// Concrete strategy: Merge Sort

class MergeSort implements SortStrategy {

public void sort(int[] data) {

if (data.length < 2) return;

int mid = data.length / 2;

int[] left = Arrays.copyOfRange(data, 0, mid);

int[] right = Arrays.copyOfRange(data, mid, data.length);

sort(left);

sort(right);

merge(data, left, right);

}

private void merge(int[] data, int[] left, int[] right) {

int i = 0, j = 0, k = 0;

while (i < left.length && j < right.length) {

if (left[i] <= right[j]) {

data[k++] = left[i++];

} else {

data[k++] = right[j++];

}

}

while (i < left.length) {

data[k++] = left[i++];

}

while (j < right.length) {

data[k++] = right[j++];

}

}

}

// Context

class Sorter {

private SortStrategy strategy;

public Sorter(SortStrategy strategy) {

this.strategy = strategy;

}

public void setStrategy(SortStrategy strategy) {

this.strategy = strategy;

}

public void sortArray(int[] data) {

strategy.sort(data);

}

}

// Client code to test Strategy Pattern

public class SortingApp {

public static void main(String[] args) {

int[] arr = {64, 25, 12, 22, 11};

Sorter sorter = new Sorter(new BubbleSort());

sorter.sortArray(arr);

System.out.println("Bubble Sorted array: " + Arrays.toString(arr));

sorter.setStrategy(new MergeSort());

sorter.sortArray(arr);

System.out.println("Merge Sorted array: " + Arrays.toString(arr));

}

}