package DesignPatterns;

import java.util.Arrays;

interface Strategy {

    void sort(int[] arr);

}

class BSort implements Strategy {

    public void sort(int[] arr) {

        for (int i = 0; i < arr.length - 1; i++) {

            boolean sorted = false;

            for (int j = 0; j < arr.length - 1 - i; j++) {

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

                    int temp = arr[j];

                    arr[j] = arr[j + 1];

                    arr[j + 1] = temp;

                    sorted = true;

                }

            }

            if (!sorted) break;

        }

    }

}

class ISort implements Strategy {

    public void sort(int[] arr) {

        for (int i = 1; i < arr.length; i++) {

            int key = arr[i];

            int j = i - 1;

            while (j >= 0 && arr[j] > key) {

                arr[j + 1] = arr[j];

                j = j - 1;

            }

            arr[j + 1] = key;

        }

    }

}

class Context {

    private Strategy strategy;

    public void setStrategy(Strategy strategy) {

        this.strategy = strategy;

    }

    public void executeStrategy(int[] arr) {

        strategy.sort(arr);

    }

}

public class StrategyPattern {

    public static void main(String[] args) {

        int[] arr1 = { 5, 4, 3, 2, 1 };

        int[] arr2 = { 5, 4, 3, 2, 1 };

        Context context1 = new Context();

        context1.setStrategy(new BSort());

        context1.executeStrategy(arr1);

        Context context2 = new Context();

        context2.setStrategy(new ISort());

        context2.executeStrategy(arr2);

        System.out.println(Arrays.toString(arr1));

        System.out.println(Arrays.toString(arr2));

    }

}

**Circuit Breaker Pattern:**

Used in distributed systems. Used to handle faults and failures.

**Prototype Pattern:**

//prototype pattern

interface Prototype {

    Prototype clone();

}

class ConcretePrototype implements Prototype {

    private int value;

    ConcretePrototype(int value){

        this.value = value;

    }

    //define setter and getter methods as needed

    public Prototype clone(){

        return new ConcretePrototype(this.value);

    }

}

public class Client {

    public static void main(String args[]){

        ConcretePrototype concreteProtoype = new ConcretePrototype(10);

        ConcretePrototype clonedPrototype = (ConcretePrototype) concreteProtoype.clone();

    }

}