Names:

Findlay Brown

Course Name:

Principles of Software Design

Lab Section:

B02

Course Code:

ENSF 480

Assignment Number:

Lab-5

Submission Date:

27/10/2023

# Exercise A & B

DemoStrategyPattern.java

import java.util.ArrayList;

import java.util.Random;

/\*\*

 \* Main file that runs and tests MyVector and sorting classes

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>DemoStrategyPattern.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class DemoStrategyPattern {

    public static *void* main(String[] *args*) {

        // Create an object of MyVector<Double> with capacity of 50 elements

        MyVector<Double> v1 = new MyVector<Double> (50);

        // Create a Random object to generate values between 0

        Random rand = new Random();

        // adding 5 randomly generated numbers into MyVector object v1

        for(*int* i = 4; i >=0; i--) {

            Item<Double> item;

            item = new Item<Double> (Double.valueOf(rand.nextDouble()\*100));

            v1.add(item);

        }

        System.out.println("---- TEST BUBBLE SORT ----");

        // displaying original data in MyVector v1

        System.out.println("The original values in v1 object are:");

        v1.display();

        // choose algorithm bubble sort as a strategy to sort object v1

        v1.setSortStrategy(new BubbleSorter<Double>());

        // perform algorithm bubble sort to v1

        v1.performSort();

        System.out.println("\nThe values in MyVector object v1 after performing BoubleSorter is:");

        v1.display();

        // create a MyVector<Integer> object V2

        MyVector<Integer> v2 = new MyVector<Integer> (50);

        // populate v2 with 5 randomly generated numbers

        for(*int* i = 4; i >=0; i--) {

            Item<Integer> item;

            item = new Item<Integer> (Integer.valueOf(rand.nextInt(50)));

            v2.add(item);

        }

        System.out.println("---- TEST INSERTION SORT ----");

        System.out.println("\nThe original values in v2 object are:");

        v2.display();

        v2.setSortStrategy(new InsertionSorter<Integer>());;

        v2.performSort();

        System.out.println("\nThe values in MyVector object v2 after performing InsertionSorter is:");

        v2.display();

        // create a MyVector<Integer> object V3

        MyVector<Integer> v3 = new MyVector<Integer> (5);

        // populate v3 with 5 randomly generated numbers

        for(*int* i = 4; i >=0; i--) {

            Item<Integer> item;

            item = new Item<Integer> (Integer.valueOf(rand.nextInt(50)));

            v3.add(item);

        }

        System.out.println("---- TEST SELECTION SORT ----");

        System.out.println("\nThe original values in v3 object are:");

        v3.display();

        v3.setSortStrategy(new SelectionSorter<Integer>());;

        v3.performSort();

        System.out.println("\nThe values in MyVector object v3 after performing SelectionSorter is:");

        v3.display();

        System.out.println("---- TEST COPY ARRAYLIST CTOR ----");

        ArrayList<Item<Integer>> arr4 = new ArrayList<Item<Integer>>(5);

        // populate v4 with 5 randomly generated numbers

        for(*int* i = 4; i >=0; i--) {

            Item<Integer> item;

            item = new Item<Integer> (Integer.valueOf(rand.nextInt(50)));

            arr4.add(item);

        }

        System.out.println("\nThe original values in arr4 object are:");

        for(Item<Integer> i : arr4){

            System.out.print(i.getItem() + " ");

        }

        System.out.println();

        // create a MyVector<Integer> object V4

        MyVector<Integer> v4 = new MyVector<Integer>(arr4);

        System.out.println("\nThe original values in v4 object are:");

        v4.display();

        // Change the values in the Original Arraylist

        arr4.set(0, new Item<Integer>(1));

        arr4.set(1, new Item<Integer>(1));

        arr4.set(2, new Item<Integer>(1));

        arr4.set(3, new Item<Integer>(1));

        arr4.set(4, new Item<Integer>(1));

        System.out.println("\nThe new values in arr4 object are:");

        for(Item<Integer> i : arr4){

            System.out.print(i.getItem() + " ");

        }

        System.out.println();

        System.out.println("\nThe new values in v4 object are:");

        v4.display();

    }

}

Item.java

/\*\*

 \* Item class

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>Item.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

class Item <E extends Number & Comparable<E>>{

    private E item;

    public Item(E *value*) {

        item = *value*;

    }

    public *void* setItem(E *value*){

        item = *value*;

    }

    public E getItem(){

        return item;

    }

}

MyVector.java

import java.util.ArrayList;

/\*\*

 \* MyVector class

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>MyVector.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class MyVector<E extends Number & Comparable<E>> {

    private ArrayList<Item<E>> storageM;

    private Sorter<E> sorter;

    public MyVector(*int* *size*){

        this.storageM = new ArrayList<Item<E>>(*size*);

    }

    public MyVector(ArrayList<Item<E>> *arr*){

        this.storageM = new ArrayList<Item<E>>(*arr*.size()+1);

        this.storageM.addAll(*arr*);

    }

    public *void* add(Item<E> *value*){

        this.storageM.add(*value*);

    }

    public *void* setSortStrategy(Sorter<E> *s*){

        this.sorter = *s*;

    }

    public *void* performSort(){

        this.sorter.sort(this.storageM);

    }

    public *void* display(){

        for(Item<E> i : storageM){

            System.out.print(i.getItem() + " ");

        }

        System.out.println();

    }

}

Sorter.java

import java.util.ArrayList;

/\*\*

 \* Sorter interface

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>Sorter.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public interface Sorter<E extends Number & Comparable<E>>{

    public *void* sort(ArrayList<Item<E>> *arr*);

    /\*\*

     \* Helper Method to compare two elements of an ArrayList

     \* @param *arr*       ArrayList of Item<E> object

     \* @param *index1*    index of the first item

     \* @param *index2*    index of the second item

     \* @return the boolean result of the comparison arr[index1] > arr[index2]

     \*/

    default public *boolean* compare(ArrayList<Item<E>> *arr*, *int* *index1*, *int* *index2*){

        return *arr*.get(*index1*).getItem().compareTo(*arr*.get(*index2*).getItem()) > 0;

    }

    /\*\*

     \* Helper method to compare two Item<E> objects

     \* @param *o1* first Item<E> object

     \* @param *o2* second Item<E> object

     \* @return the boolean result of the comparison o1 > o2

     \*/

    default public *boolean* compare(Item<E> *o1*, Item<E> *o2*){

        return *o1*.getItem().compareTo(*o2*.getItem()) > 0;

    }

    /\*\*

     \* Helper method to swap two elements of an ArrayList

     \* @param *arr*       ArrayList of Item<E> object

     \* @param *index1*    index of the first item

     \* @param *index2*    index of the second item

     \*/

    default public *void* swap(ArrayList<Item<E>> *arr*, *int* *index1*, *int* *index2*){

        Item<E> temp = *arr*.get(*index1*);

*arr*.set(*index1*, *arr*.get(*index2*));

*arr*.set(*index2*, temp);

    }

}

BubbleSorter.java

import java.util.ArrayList;

/\*\*

 \* Bubble Sorter class

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>BubbleSorter.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class BubbleSorter<E extends Number & Comparable<E>> implements Sorter<E> {

    public *void* sort(ArrayList<Item<E>> *arr*){

*boolean* sorted;

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

            sorted = true;

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

                if(compare(*arr*, j, j+1)){

                    swap(*arr*, j, j+1);

                    sorted = false;

                }

            }

            if(sorted == true)

                break;

        }

    }

}

InsertionSorter.java

import java.util.ArrayList;

/\*\*

 \* Insertion Sorter class

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>InsertionSort.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class InsertionSorter<E extends Number & Comparable<E>> implements Sorter<E>{

    public *void* sort(ArrayList<Item<E>> *arr*){

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

            Item<E> keyItem = *arr*.get(i);

*int* index = i - 1;

            while (index >= 0 && compare(*arr*.get(index), keyItem)) {

*arr*.set(index+1, *arr*.get(index));

                index--;

            }

*arr*.set(index+1, keyItem);

        }

    }

}

SelectionSorter.java

import java.util.ArrayList;

/\*\*

 \* Selection Sorter class

 \* <p>ENSF 480 - Lab 5 Exercise A and B

 \* <p>SelectionSorter.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class SelectionSorter<E extends Number & Comparable<E>> implements Sorter<E> {

    public *void* sort(ArrayList<Item<E>> *arr*){

*int* swapIndex = 0;

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

*int* minIndex = i;

            for(*int* j = i; j < *arr*.size(); j++){

                if(compare(*arr*, minIndex, j)){

                    minIndex = j;

                }

            }

            swap(*arr*,swapIndex,minIndex);

        }

    }

}

# Exercise C

ObserverPatternController.java

/\*\*

 \* Main file that runs and tests code following the observer design pattern

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>ObserverPatternController.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class ObserverPatternController {

    public static *void* main(String []*s*) {

*double* [] arr = {10, 20, 33, 44, 50, 30, 60, 70, 80, 10, 11, 23, 34, 55};

        System.out.println("Creating object mydata with an empty list -- no data:");

        DoubleArrayListSubject mydata = new DoubleArrayListSubject();

        System.out.println("Expected to print: Empty List ...");

        mydata.display();

        mydata.populate(arr);

        System.out.println("mydata object is populated with: 10, 20, 33, 44, 50, 30, 60, 70, 80, 10, 11, 23, 34, 55 ");

        System.out.print("Now, creating three observer objects: ht, vt, and hl ");

        System.out.println("\nwhich are immediately notified of existing data with different views.");

        ThreeColumnTable\_Observer ht = new ThreeColumnTable\_Observer(mydata);

        FiveRowsTable\_Observer vt = new FiveRowsTable\_Observer(mydata);

        OneRow\_Observer hl = new OneRow\_Observer(mydata);

        System.out.println("\n\nChanging the third value from 33, to 66 -- (All views must show this change):");

        mydata.setData(66.0, 2);

        System.out.println("\n\nAdding a new value to the end of the list -- (All views must show this change)");

        mydata.addData(1000.0);

        System.out.println("\n\nNow removing two observers from the list:");

        mydata.remove(ht);

        mydata.remove(vt);

        System.out.println("Only the remained observer (One Row ), is notified.");

        mydata.addData(2000.0);

        System.out.println("\n\nNow removing the last observer from the list:");

        mydata.remove(hl);

        System.out.println("\nAdding a new value the end of the list:");

        mydata.addData(3000.0);

        System.out.println("Since there is no observer -- nothing is displayed ...");

        System.out.print("\nNow, creating a new Three-Column observer that will be notified of existing data:");

        ht = new ThreeColumnTable\_Observer(mydata);

    }

}

Subject.java

/\*\*

 \* Subject interface

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>Subject.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public interface Subject {

    public *void* register(Observer *o*);

    public *void* remove(Observer *o*);

    public *void* notifyObserver();

}

DoubleArrayListSubject.java

import java.util.ArrayList;

/\*\*

 \* DoubleArrayListSubject class

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>DoubleArrayListSubject.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class DoubleArrayListSubject implements Subject{

    private ArrayList<Observer> observers;

    private ArrayList<Double> data;

    public DoubleArrayListSubject(){

        this.observers = new ArrayList<Observer>();

        this.data = new ArrayList<Double>();

    }

    public *void* addData(Double *d*){

        this.data.add(*d*);

    }

    public *void* setData(*int* *index*, Double *data*){

        this.data.set(*index*, *data*);

    }

    public *void* setData(Double *data*, *int* *index*){

        this.data.set(*index*, *data*);

        notifyObserver();

    }

    public ArrayList<Double> getData(){

        return this.data;

    }

    public *void* populate(*double*[] *newData*){

        this.data = new ArrayList<Double>(*newData*.length);

        for (Double d : *newData*) {

            this.data.add(d);

        }

    }

    public *void* register(Observer *o*){

        this.observers.add(*o*);

*o*.update(this.data);

    }

    public *void* remove(Observer *o*){

        this.observers.remove(*o*);

    }

    public *void* notifyObserver(){

        observers.forEach(*o* *->* *o*.update(data));

    }

    public *void* display(){

        if(data.size() != 0){

            data.forEach(*x* *->* System.out.print(*x* + " "));

            System.out.println();

        }

        else{

            System.out.println("Empty List...");

        }

    }

}

Observer.java

import java.util.ArrayList;

/\*\*

 \* Observer interface

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>Observer.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public interface Observer {

    public *void* update(ArrayList<Double> *newArrayList*);

}

OneRow\_Observer.java

import java.util.ArrayList;

/\*\*

 \* OneRow\_Observer class

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>OneRow\_Observer.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class OneRow\_Observer implements Observer{

    private ArrayList<Double> arrayList;

    private DoubleArrayListSubject data;

    public OneRow\_Observer(DoubleArrayListSubject *data*){

        this.data = *data*;

        this.data.register(this);

    }

    @*Override*

    public *void* update(ArrayList<Double> *newArrayList*){

        this.arrayList = *newArrayList*;

        display();

    }

    public *void* display(){

        System.out.println("\nNotification to One-Row Observer: Data Changed:");

        for (Double data : this.arrayList) {

            System.out.print(data + "  ");

        }

        System.out.println();

    }

}

ThreeColumnTable\_Observer.java

import java.util.ArrayList;

/\*\*

 \* ThreeColumnTable\_Observer class

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>ThreeColumnTable\_Observer.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class ThreeColumnTable\_Observer implements Observer{

    private ArrayList<Double> arrayList;

    private DoubleArrayListSubject data;

    public ThreeColumnTable\_Observer(DoubleArrayListSubject *data*){

        this.data = *data*;

        this.data.register(this);

    }

    @*Override*

    public *void* update(ArrayList<Double> *newArrayList*){

        this.arrayList = *newArrayList*;

        display();

    }

    public *void* display(){

*int* i = 0;

        System.out.println("\nNotification to Three-Column Table Observer: Data Changed:");

        for (Double data : this.arrayList) {

            System.out.print(data + "\t");

            i++;

            if(i == 3){

                System.out.println();

                i = 0;

            }

        }

        System.out.println();

    }

}

FiveRowsTable\_Observer.java

import java.util.ArrayList;

/\*\*

 \* FiveRowsTable\_Observer class

 \* <p>ENSF 480 - Lab 5 Exercise C

 \* <p>FiveRowsTable\_Observer.java

 \* @author  Findlay Brown

 \* @version 1.0

 \* @since   27-10-2023

 \*/

public class FiveRowsTable\_Observer implements Observer{

    private ArrayList<Double> arrayList;

    private DoubleArrayListSubject data;

    public FiveRowsTable\_Observer(DoubleArrayListSubject *data*){

        this.data = *data*;

        this.data.register(this);

    }

    @*Override*

    public *void* update(ArrayList<Double> *newArrayList*){

        this.arrayList = *newArrayList*;

        display();

    }

    public *void* display(){

        System.out.println("\nNotification to Five-Rows Table Observer: Data Changed:");

        for(*int* i=0; i<5;i++){

            for(*int* j=i; j <= this.arrayList.size()-1; j = j + 5){

                System.out.print(this.arrayList.get(j)+"\t");

            }

            System.out.println();

        }

    }

}