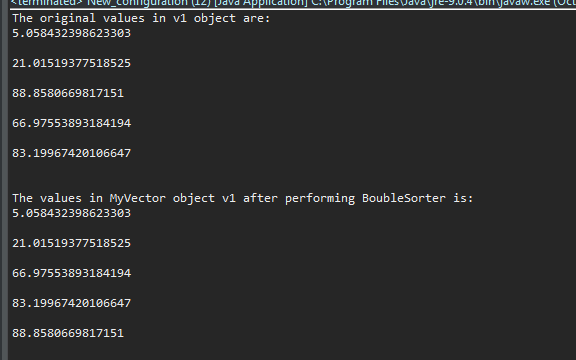
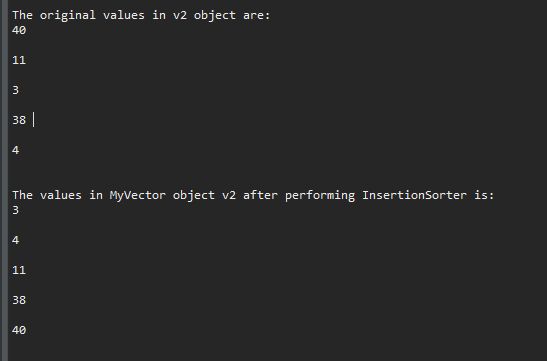
Name: Muhammad Hassan

UCID: 30032437

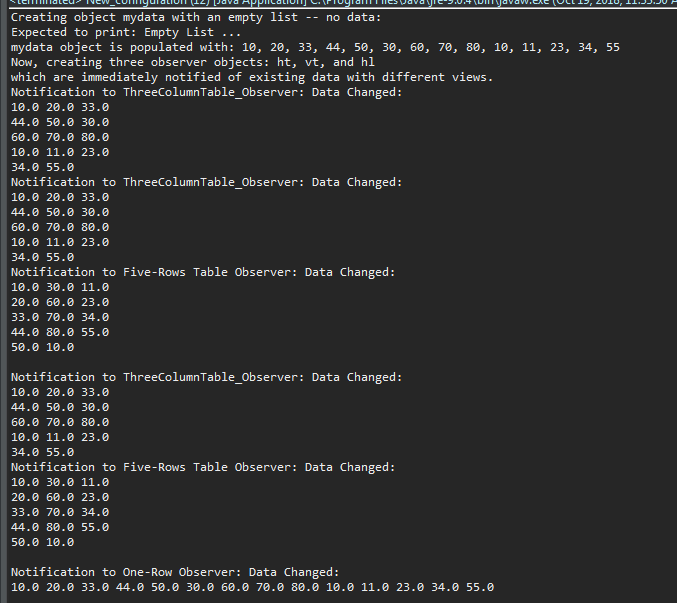
Lab report: ENSF 480 lab 5

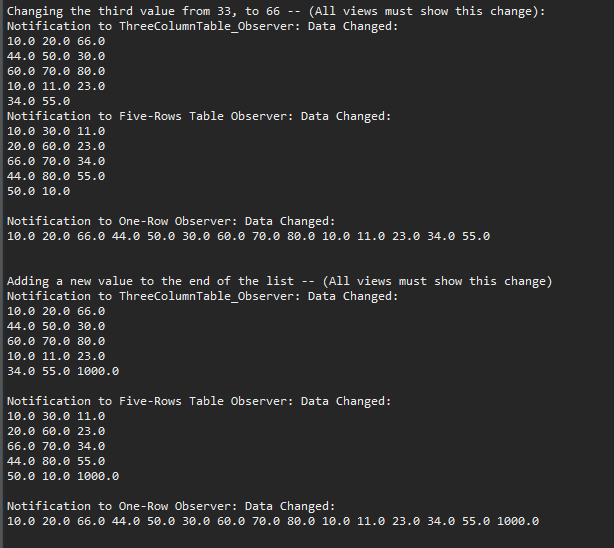
Ex A: Output

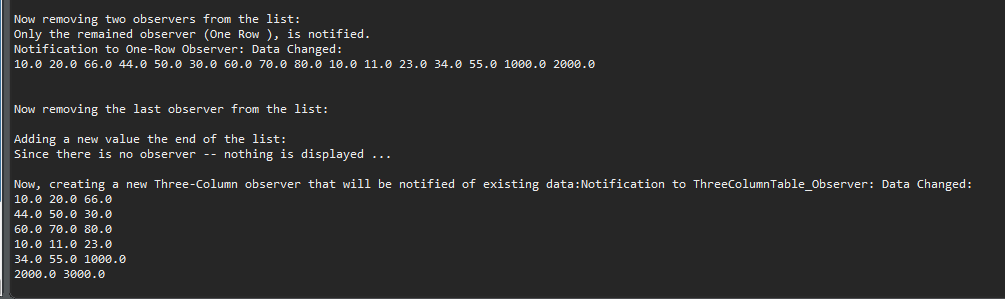




Ex C: Output







BubbleSorter.java

**import** java.util.ArrayList;

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

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

**int** **n** = arr.size();

**int** **i**, **j**;

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

//Last i elements are already in place

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

**if** (arr.get(j).compareTo(arr.get(j+1)) > 0) {

//swapping

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

arr.set(j, arr.get(j+1));

arr.set(j+1, temp);

}

}

}

DemostrategyPattern.java

/\* ENSF 480 - Lab 5 - Exercise A and B

\* M. Moussavi, October 2018

\*/

**import** java.util.Random;

**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);

}

// 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("\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();

}

}

DoubleArrayListSubject.java

**import** java.util.\*;

**public** **class** **DoubleArrayListSubject** **implements** Subject{

**private** **ArrayList**<Double> data;

**private** **ArrayList**<Observer> list;

**static** **int** *id* = 0;

//constructor

**DoubleArrayListSubject**(){

data = **new** ArrayList<Double>();

list = **new** ArrayList<Observer>();

}

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

data.add(d);

notifyAllObservers();

}

**public** **void** **setData**(**Double** d, **int** observerID){

data.set(observerID, d);

notifyAllObservers();

}

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

**for**(**int** **i** = 0; i < d.length; i++)

data.add(d[i]);

notifyAllObservers();

}

***@Override***

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

// **TODO** Auto-generated method stub

list.add(o);

*id*++;

notifyAllObservers();

}

***@Override***

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

// **TODO** Auto-generated method stub

list.remove(list.indexOf(o));

}

***@Override***

**public** **void** **notifyAllObservers**() {

// **TODO** Auto-generated method stub

**for**(Observer **o**: list)

o.update(data);

}

***@Override***

**public** **void** **display**() {

// **TODO** Auto-generated method stub

**for**(**int** **i**=0;i<data.size();i++)

**System**.***out***.print(data.get(i) + " " + "\n");

}

}

FiveRowTableObserver

**import** java.util.\*;

**public** **class** **FiveRowsTable\_Observer** **implements** Observer{

**private** **int** idOfObserver;

**private** **ArrayList**<Double> data;

**private** Subject dList;

//ctor

**FiveRowsTable\_Observer**(Subject s)

{

dList = s;

idOfObserver = ++**DoubleArrayListSubject**.*id*;

s.registerObserver(**this**);

}

***@Override***

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

// **TODO** Auto-generated method stub

data=arr;

display();

}

**public** **void** **display**(){

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

**int** **k**=0;

**for**(**int** **i**=0;i<data.size();i++) {

**if**(data.size() == k) **break**;

**for**(**int** **j**=0;j<data.size();j++) {

**if**((j-i)>=0 && (j-i)%5 == 0) {

**System**.***out***.print(data.get(j) + " ");

k++;

}

}

**System**.***out***.println();

}

**System**.***out***.println();

}

}

InsertionSorter

**import** java.util.\*;

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

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

**int** **n**=arr.size();

**int** **i**, **j**;

**Item**<**E**> **key**;

**for** (i = 1; i < n; i++)

{

key = arr.get(i);

j = i-1;

/\* Move elements of arr[0..i-1], that are

greater than key, to one position ahead

of their current position \*/

**while** (j >= 0 && arr.get(j).compareTo(key) > 0)

{

arr.set(j+1,arr.get(j));

j = j-1;

}

arr.set(j+1, key);

}

}

}

Item.java

/\* ENSF 480 - Lab 5 Exercise A and B

\* M. Moussavi, October 2018

\*/

**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;

}

**public** **int** **compareTo**(**Item**<**E**> a)

{

**if**(item.compareTo(a.item) > 0) **return** 1;

**else** **if**(item.compareTo(a.item) < 0) **return** -1;

**else** **return** 0;

}

}

MyVector.java

**import** java.util.\*;

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

**private** **ArrayList**<**Item**<**E**>> storageM;

**private** Sorter<**E**> sorter;

**public** **MyVector**(**int** n) {

storageM=**new** ArrayList<**Item**<**E**>>(n);

}

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

storageM=arr;

}

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

//System.out.print("adding value: "+ value);

storageM.add(value);

}

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

sorter = s;

}

**public** **void** **performSort**() {

sorter.performSort(storageM);

}

**public** **void** **display**() {

**for**(**int** **i** = 0; i < storageM.size(); i++)

**System**.***out***.println(storageM.get(i).getItem() + " " + "\n");

}

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

//

//

// }

}

Observer.java

**import** java.util.ArrayList;

**public** **interface** Observer {

**abstract** **public** **void** **update**(**ArrayList**<Double> arr);

}

OneRowObserver.java

**import** java.util.\*;

**public** **class** **OneRow\_Observer** **implements** Observer{

**private** **int** idOfObserver;

**private** **ArrayList**<Double> data;

**private** Subject dList;

//ctor

**OneRow\_Observer**(Subject s)

{

dList = s;

idOfObserver = ++**DoubleArrayListSubject**.*id*;

s.registerObserver(**this**);

}

***@Override***

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

// **TODO** Auto-generated method stub

data=arr;

display();

}

**public** **void** **display**(){

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

**for**(**int** **i**=0;i<data.size();i++)

**System**.***out***.print(data.get(i) + " ");

**System**.***out***.println();

}

}

SelectionSorter.java

**import** java.util.ArrayList;

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

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

**int** **n** = arr.size();

**int** **i**, **j**, **min\_idx**;

// One by one move boundary of unsorted subarray

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

{

// Find the minimum element in unsorted array

min\_idx = i;

**for** (j = i+1; j < n; j++)

**if** (arr.get(j).compareTo(arr.get(min\_idx)) < 0)

min\_idx = j;

// Swap the found minimum element with the first element

**Item**<**E**> **temp** = arr.get(min\_idx);

arr.set(min\_idx, arr.get(i));

arr.set(i,temp);

}

}

}

Subject.java

**public** **interface** Subject {

**abstract** **void** **registerObserver**(Observer o);

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

**abstract** **void** **notifyAllObservers**();

**abstract** **void** **display**();

}

ThreeColumnObserver

**import** java.util.\*;

**public** **class** **ThreeColumnTable\_Observer** **implements** Observer{

**private** **int** idOfObserver;

**private** **ArrayList**<Double> data;

**private** Subject dList;

//ctor

**ThreeColumnTable\_Observer**(Subject s)

{

dList = s;

idOfObserver = ++**DoubleArrayListSubject**.*id*;

s.registerObserver(**this**);

}

***@Override***

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

// **TODO** Auto-generated method stub

data=arr;

display();

}

**public** **void** **display**(){

**System**.***out***.println("Notification to ThreeColumnTable\_Observer: Data Changed: ");

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

**System**.***out***.print(data.get(i-1) + " ");

**if**(i%3 == 0) **System**.***out***.println();

}

**System**.***out***.println();

}

}

Sorter.java

**import** java.util.\*;

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

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

}