Lab 05 Report **Completed by: Dominic Choi 30109955**

Exercise A:
MyVector.java

```
package exABpackage;
import java.util.ArrayList;

public class MyVector<E extends Number & Comparable<E>> {
    private ArrayList<Item<E>> storageM;
    private Sorter<E> sorter;

    public MyVector(int n) {
        storageM = new ArrayList<>(n);
    }

    public MyVector(ArrayList<Item<E>> arr) {
        storageM = new ArrayList<>(arr);
    }

    public void add(Item<E> value) {
        storageM.add(value);
    }

    public void setSortStrategy(Sorter<E> s) {
        sorter = s;
    }

    public void performSort() {
        if (sorter != null) {
            sorter.sort(storageM);
        }
    }

    public void display() {
        for (int i = 0; i < storageM.size(); i++) {
            System.out.print(storageM.get(i).getItem());
            if (i < storageM.size() - 1) {
                System.out.print(" ");
```

```
            }
        }
        System.out.println();
    }
}
```

Sorter.java

```
package exABpackage;
import java.util.List;

public interface Sorter<E extends Number & Comparable<E>> {
    void sort(List<Item<E>> list);
}
```

InsertionSorter.java

```
package exABpackage;

import java.util.List;

public class InsertionSorter<E extends Number & Comparable<E>> implements
Sorter<E> {
    @Override
    public void sort(List<Item<E>> list) {
        int n = list.size();
        for (int i = 1; i < n; i++) {
            Item<E> key = list.get(i);
            int j = i - 1;
            while (j >= 0 && list.get(j).getItem().doubleValue() >
key.getItem().doubleValue()) {
                list.set(j + 1, list.get(j));
                j = j - 1;
            }
            list.set(j + 1, key);
        }
    }
}
```

BubbleSorter.java

```java
package exABpackage;

import java.util.List;

public class BubbleSorter<E extends Number & Comparable<E>> implements
Sorter<E> {
    @Override
    public void sort(List<Item<E>> list) {
        int n = list.size();
        boolean swapped;
        do {
            swapped = false;
            for (int i = 1; i < n; i++) {
                if (list.get(i - 1).getItem().doubleValue() >
list.get(i).getItem().doubleValue()) {
                    Item<E> temp = list.get(i - 1);
                    list.set(i - 1, list.get(i));
                    list.set(i, temp);
                    swapped = true;
                }
            }
        } while (swapped);
    }
}
```

Exercise B:
SelectionSorter.java

```java
package exABpackage;

import java.util.List;

public class SelectionSorter<E extends Number & Comparable<E>> implements
Sorter<E> {
    @Override
    public void sort(List<Item<E>> list) {
        int n = list.size();
        for (int i = 0; i < n - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < n; j++) {
                if
(list.get(j).getItem().compareTo(list.get(minIndex).getItem()) < 0) {
                    minIndex = j;
                }
            }
            if (minIndex != i) {
                Item<E> temp = list.get(i);
                list.set(i, list.get(minIndex));
                list.set(minIndex, temp);
            }
        }
    }
}
```

Output for A and B:

```
PS D:\Schoolwork\2023-4 Fall\ENSF 480\ENSF-480-Labs>  d:; cd 'd:\Schoolwork\2023-4 Fall\ENSF 480\ENSF-480-Labs'; & 'C:\Program Files\Java\jdk1.8.0_361\bin\java.exe' '-cp' 'C:\Users\dominic.KCHOI\AppData\Roaming\Code\User\workspaceStorage\f89ec19d9c63bf57d47fbe4a703b9a6c\redhat.java\jdt_ws\ENSF-480-Labs_421e831c\bin' 'DemoStrategyPattern'
The original values in v1 object are:
20.70538794923881 70.29613736051206 32.22804461682668 23.187525732022507 97.33922997996518
The original values in v3 object are:
20.70538794923881 70.29613736051206 32.22804461682668 23.187525732022507 97.33922997996518

The values in MyVector object v1 after performing BoubleSorter is:
20.70538794923881 23.187525732022507 32.22804461682668 70.29613736051206 97.33922997996518

The values in MyVector object v3 after performing SelectionSorter is:
20.70538794923881 23.187525732022507 32.22804461682668 70.29613736051206 97.33922997996518

The original values in v2 object are:
21 6 33 13 26

The values in MyVector object v2 after performing InsertionSorter is:
6 13 21 26 33
```

Exercise C:
DoubleArrayListSubject.java

```java
package exCpackage;

import java.util.ArrayList;

public class DoubleArrayListSubject implements Subject {
    public ArrayList<Double> data;
    private ArrayList<Observer> observers;

    public DoubleArrayListSubject() {
        observers = new ArrayList<Observer>();
        data = new ArrayList<Double>();
    }

    // displays current content of data
    public void display() {
        if (data.isEmpty()) {
            System.out.println("Empty List...");
        } else {
            System.out.print("mydata object is populated with: ");
            for (int i = 0; i < data.size(); i++) {
                System.out.print(data.get(i));
                if (i < data.size() - 1) {
                    System.out.print(", ");
                }
            }
            System.out.println();
        }
    }

    public ArrayList<Double> getData() {
        return data;
    }

    // notifies all observers and updates
    public void notifyObserver() {
        for (int i = 0; i < observers.size(); i++) {
```

```
            Observer o = observers.get(i);
            o.update(data);
        }
    }

    // add double to data arraylist
    public void addData(double value) {
        Double newDouble = Double.valueOf(value);
        data.add(newDouble);
        notifyObserver();
    }

    // set double at index in arraylist
    public void setData(double value, int index) {
        Double newDouble = Double.valueOf(value);
        data.set(index, newDouble);
        notifyObserver();
    }

    // populate data arraylist with array of doubles
    public void populate(double[] arr) {
        for (double value : arr) {
            data.add(Double.valueOf(value));
        }
        notifyObserver();
    }

    // remove an observer
    public void remove(Observer observer) {
        observers.remove(observer);
    }

    // add an observer
    public void add(Observer observer) {
        observers.add(observer);
        observer.update(data);
    }
}
```

FiveRowsTable_Observer

```
package exCpackage;

import java.util.ArrayList;

public class FiveRowsTable_Observer implements Observer{

    private Subject subject;
    private ArrayList<Double> data;

    public FiveRowsTable_Observer(Subject s) {
        this.subject = s;
        s.add(this);
    }

    public void update(ArrayList<Double> list) {
        this.data = list;
        display();
    }

    public void display() {
        System.out.println();
        System.out.println("Notification to Five-Rows Table Observer: Data
Changed:");

        String[] strings = new String[5];
        int counter = 0;
        for (int i = 0; i < data.size(); i++) {
            if (strings[counter] == null){
                strings[counter] = data.get(i) + " ";
            } else {
                strings[counter] += (data.get(i) + " ");

            }

            counter++;
            if(counter == 5){
                counter = 0;
```

```
            }
        }

        for(int i = 0; i < 5; i++){
            System.out.println(strings[i]);
        }

    }
}
```

Observer.java

```
package exCpackage;

import java.util.ArrayList;

public interface Observer {
    void update(ArrayList<Double> list);
}
```

OneRow_Observer.java

```
package exCpackage;

import java.util.ArrayList;

public class OneRow_Observer implements Observer {

    private Subject subject;
    private ArrayList<Double> data;

    public OneRow_Observer(Subject s) {
        this.subject = s;
        s.add(this);
    }

    public void update(ArrayList<Double> list) {
        this.data = list;
        display();
    }
```

```
    public void display() {
        System.out.println();
        System.out.println("Notification to One-Row Observer: Data Changed:");
        for (int i = 0; i < data.size(); i++) {
            System.out.print(data.get(i));
            if (i < data.size() - 1) {
                System.out.print(" ");
            }
        }
        System.out.println();
    }
}
```

Subject.java

```
package exCpackage;

import java.util.ArrayList;

interface Subject {
    ArrayList<Double> data = null;
    void addData(double data);
    void setData(double data, int index);
    void populate(double[] arr);
    void add(Observer observer);
    void remove(Observer observer);
}
```

ThreeColumnTable_Observer.java

```
package exCpackage;

import java.util.ArrayList;

public class ThreeColumnTable_Observer implements Observer{
    private Subject subject;
    private ArrayList<Double> data;

    public ThreeColumnTable_Observer(Subject s) {
```

```
        this.subject = s;
        s.add(this);
    }

    public void update(ArrayList<Double> list) {
        this.data = list;
        display();
    }

    public void display() {
        System.out.println();
        System.out.println("Notification to Three-Column Table Observer: Data
Changed:");
        int counter = 0;
        for (int i = 0; i < data.size(); i++) {
            if (counter == 3) {
                System.out.println();
                counter = 0;
            }
            System.out.print(data.get(i));
            counter++;
            if (i < data.size() - 1) {
                System.out.print(" ");
            }
        }
        System.out.println();

    }
}
```

Output:

```
PS D:\Schoolwork\2023-4 Fall\ENSF 480\ENSF-480-Labs>  & 'C:\Program Files\Java\jdk1.8.0_3
e4a703b9a6c\redhat.java\jdt_ws\ENSF-480-Labs_421e831c\bin' 'exCpackage.ObserverPatternCon
Creating object mydata with an empty list -- no data:
Expected to print: Empty List ...
Empty List...
mydata object is populated with: 10, 20, 33, 44, 50, 30, 60, 70, 80, 10, 11, 23, 34, 55
Now, creating three observer objects: ht, vt, and hl
which are immediately notified of existing data with different views.

Notification to Three-Column Table Observer: Data Changed:
10.0 20.0 33.0
44.0 50.0 30.0
60.0 70.0 80.0
10.0 11.0 23.0
34.0 55.0

Notification to Five-Rows Table Observer: Data Changed:
10.0 30.0 11.0
20.0 60.0 23.0
33.0 70.0 34.0
44.0 80.0 55.0
50.0 10.0

Notification to One-Row Observer: Data Changed:
10.0 20.0 33.0 44.0 50.0 30.0 60.0 70.0 80.0 10.0 11.0 23.0 34.0 55.0


Changing the third value from 33, to 66 -- (All views must show this change):

Notification to Three-Column Table Observer: Data Changed:
10.0 20.0 66.0
44.0 50.0 30.0
60.0 70.0 80.0
10.0 11.0 23.0
34.0 55.0

Notification to Five-Rows Table Observer: Data Changed:
10.0 30.0 11.0
20.0 60.0 23.0
66.0 70.0 34.0
44.0 80.0 55.0
50.0 10.0

Notification to One-Row Observer: Data Changed:
10.0 20.0 66.0 44.0 50.0 30.0 60.0 70.0 80.0 10.0 11.0 23.0 34.0 55.0


Adding a new value to the end of the list -- (All views must show this change)

Notification to Three-Column Table Observer: Data Changed:
10.0 20.0 66.0
44.0 50.0 30.0
60.0 70.0 80.0
10.0 11.0 23.0
34.0 55.0 1000.0

Notification to Five-Rows Table Observer: Data Changed:
10.0 30.0 11.0
20.0 60.0 23.0
66.0 70.0 34.0
44.0 80.0 55.0
50.0 10.0 1000.0

Notification to One-Row Observer: Data Changed:
10.0 20.0 66.0 44.0 50.0 30.0 60.0 70.0 80.0 10.0 11.0 23.0 34.0 55.0 1000.0


Now removing two observers from the list:
Only the remained observer (One Row ), is notified.

Notification to One-Row Observer: Data Changed:
10.0 20.0 66.0 44.0 50.0 30.0 60.0 70.0 80.0 10.0 11.0 23.0 34.0 55.0 1000.0 2000.0


Now removing the last observer from the list:

Adding a new value the end of the list:
Since there is no observer -- nothing is displayed ...

Now, creating a new Three-Column observer that will be notified of existing data:
Notification to Three-Column Table Observer: Data Changed:
10.0 20.0 66.0
44.0 50.0 30.0
60.0 70.0 80.0
10.0 11.0 23.0
34.0 55.0 1000.0
2000.0 3000.0
```