Course: ENSF614 – Fall 2025
Lab #: Lab 5
Instructor: Mahmood Moussavi
Student Name: John Zhou
Submission Date: Oct 20th, 2025

I have been keeping all the files in github. I hope by providing this github link will help you a little bit.
https://github.com/JZ-Zhou-UofC/ENSF-614-assignment-repo

Ex A & Ex B

```
/*
*
*
File Name: BubbleSorter.java
Assignment: Lab 5 Exercise A
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
import java.util.ArrayList;


public class BubbleSorter<E extends Number & Comparable<E>> implements Sorter<E> {
   @Override
   public void sort(ArrayList<Item<E>> items) {
      int size = items.size();
      for (int i = 0; i < size - 1; i++) {
         boolean swapped = false;
         for (int j = 0; j < size - i - 1; j++) {
            if (items.get(j).getItem().compareTo(items.get(j + 1).getItem()) > 0) {
               Item<E> temp = items.get(j);
               items.set(j, items.get(j + 1));
               items.set(j + 1, temp);
               swapped = true;
            }
         }
         if (!swapped) break;
      }
```

```
    }
}
```

```
/*
*
*
File Name: DemoStrategyPattern.java
Assignment: Lab 5 Exercise A and B
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
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();

MyVector<Float> v3 = new MyVector<Float> (50);

// populate v2 with 5 randomly generated numbers
for(int i = 4; i >=0; i--) {
```

```
        Item<Float> item;
        item = new Item<Float> (Float.valueOf(rand.nextInt(50)));
        v3.add(item);
        }

        System.out.println("\nThe original values in v3 object are:");
        v3.display();
        v3.setSortStrategy(new SelectionSorter<Float>());;
        v3.performSort();
        System.out.println("\nThe values in MyVector object v3 after performing SelectionSorter is:");
        v3.display();
    }


}
```

```
/*
*
*
File Name: InsertionSorter.java
Assignment: Lab 5  B
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
import java.util.ArrayList;

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

   @Override
   public void sort(ArrayList<Item<E>> items) {
      int size = items.size();
      for (int i = 1; i < size; i++) {
         Item<E> key = items.get(i);
         int j = i - 1;

         while (j >= 0 && items.get(j).getItem().compareTo(key.getItem()) > 0) {
            items.set(j + 1, items.get(j));
            j--;
         }
         items.set(j + 1, key);
      }
   }
}
```

```
/* ENSF 614 - Lab 5 Exercise A and B
 * M. Moussavi, October 2024
 *
 */

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;
   }
}
```

```
/*
*
*
File Name: MyVector.java
Assignment: Lab 5 Exercise A
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
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) {
      this.sorter = s;
   }
```

```
    public void performSort() {

      sorter.sort(storageM);
    }

    public void display() {
       for (Item<E> item : storageM) {
          System.out.print(item.getItem() + "  ");
       }
       System.out.println();
    }
}
```

```
/*
*
*
File Name: SelectionSorter.java
Assignment: Lab 5 Exercise B
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
import java.util.ArrayList;

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

   @Override
   public void sort(ArrayList<Item<E>> items) {
      int size = items.size();

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

         // Find the index of the minimum element in the unsorted portion
         for (int j = i + 1; j < size; j++) {
            if (items.get(j).getItem().compareTo(items.get(minIndex).getItem()) < 0) {
               minIndex = j;
            }
         }

         // Swap the found minimum with the first element of unsorted portion
         if (minIndex != i) {
            Item<E> temp = items.get(i);
```

```
                items.set(i, items.get(minIndex));
                items.set(minIndex, temp);
            }
        }
    }
}
```

```
/*
*
*
File Name: Sorter.java
Assignment: Lab 5 Exercise A
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
import java.util.ArrayList;

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

```
/*
*
*
File Name: SortStrategy.java
Assignment: Lab 5 Exercise A
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
public interface SortStrategy<E extends Number & Comparable<E>> {
   void sort(Item<E>[] var1, int var2);
}
```

```
john2@John-Laptop MINGW64 ~/Desktop/uofc/614/ENSF-614-assignment-repo/assignment5/EXAandB (main)
$ java DemoStrategyPattern
The original values in v1 object are:
89.49977493672083  59.57801050718117  21.061397212551093  92.92986805087772  5.128171531743309

The values in MyVector object v1 after performing BoubleSorter is:
5.128171531743309  21.061397212551093  59.57801050718117  89.49977493672083  92.92986805087772

The original values in v2 object are:
35  45  49  29  36

The values in MyVector object v2 after performing InsertionSorter is:
29  35  36  45  49

The original values in v3 object are:
32.0  15.0  49.0  2.0  8.0

The values in MyVector object v3 after performing SelectionSorter is:
2.0  8.0  15.0  32.0  49.0
```

Ex C

```
package EXC;

import java.util.ArrayList;

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

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

    @Override
    public void attach(Observer o) {
        observers.add(o);
        o.update(data);
    }

    @Override
    public void remove(Observer o) {
        observers.remove(o);
    }

    @Override
    public void notifyObservers() {
        for (Observer o : observers) {
```

```
        o.update(data);
    }
}

public void addData(Double value) {
    data.add(value);
    notifyObservers();
}

public void setData(Double value, int index) {
    if (index >= 0 && index < data.size()) {
        data.set(index, value);
        notifyObservers();
    } else {
        System.out.println("Index out of range.");
    }
}

public void populate(double[] arr) {
    data.clear();
    for (double v : arr) {
        data.add(v);
    }
    notifyObservers();
}

public void display() {
    if (data.isEmpty()) {
        System.out.println("Empty List ...");
```

```
        } else {


            System.out.println("default display");
        }
    }
}
```

```
/*
*
*
File Name: FiveRowsTable_Observer.java
Assignment: Lab 5 Exercise C
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
package EXC;

import java.util.ArrayList;

public class FiveRowsTable_Observer implements Observer {
   private Subject subject;

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

   @Override
   public void update(ArrayList<Double> data) {
      System.out.println("\nNotification to five-rows Table Observer: Data Changed:");
      display(data);
   }

   private void display(ArrayList<Double> data) {

      int numRows = 5;
```

```
        for (int row = 0; row < numRows; row++) {
            for (int index = row; index < data.size(); index += numRows) {
                System.out.printf("%-8.2f", data.get(index));
            }
            System.out.println();
        }
    }
}
```

```
/*
*
*
File Name: Observer.java
Assignment: Lab 5 Exercise C
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
package EXC;
import java.util.ArrayList;

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

```
package EXC;

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);
    }
}
```

```
/*
*
*
File Name: OneRow_Observer.java
Assignment: Lab 5 Exercise C
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
package EXC;

import java.util.ArrayList;

public class OneRow_Observer implements Observer {
   private Subject subject;

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

   @Override
   public void update(ArrayList<Double> data) {
         System.out.println("\nNotification to one-row Table Observer: Data Changed:");
      display(data);
   }

   private void display(ArrayList<Double> data) {

      for (int index=0; index<data.size();index++) {
```

```
            System.out.printf("%-8.2f", data.get(index));
        }
        System.out.println();
    }
}
```

```
/*
*
*
File Name: Subject.java
Assignment: Lab 5 Exercise C
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
package EXC;
public interface Subject {
   void attach(Observer o);
   void remove(Observer o);
   void notifyObservers();
}
```

```
/*
*
*
File Name: ThreeColumnTable_Observer.java
Assignment: Lab 5 Exercise C
*  Completed by: John Zhou
*  Submission Date: Oct 29th, 2025
*/
package EXC;

import java.util.ArrayList;

public class ThreeColumnTable_Observer implements Observer {
   private Subject subject;

   public ThreeColumnTable_Observer(Subject subject) {
      this.subject = subject;
      subject.attach(this);
   }

   @Override
   public void update(ArrayList<Double> data) {
      System.out.println("\nNotification to three-Column Table Observer: Data Changed:");
      display(data);
   }

   private void display(ArrayList<Double> data) {

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

```
            System.out.printf("%-8.2f", data.get(i));
            if ((i + 1) % 3 == 0) System.out.println();
        }
        System.out.println();
    }
}
```

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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


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:


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:
```

```
Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


Now removing the last observer from the list:

44.00   80.00   55.00
50.00   10.00   1000.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


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 ...
Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


Now removing two observers from the list:
```

```
john2@John-Laptop MINGW64 ~/Desktop/uofc/614/ENSF-614-assignment-repo/assignment5 (main)
$ java EXC.ObserverPatternController
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


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 ...
44.00   80.00   55.00
50.00   10.00   1000.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


Now removing the last observer from the list:

Adding a new value the end of the list:
44.00   80.00   55.00
50.00   10.00   1000.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


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

Notification to one-row Table Observer: Data Changed:
```

```
john2@John-Laptop MINGW64 ~/Desktop/uofc/614/ENSF-614-assignment-repo/assignment5 (main)
$ java EXC.ObserverPatternController


Notification to five-rows Table Observer: Data Changed:
10.00   30.00   11.00
20.00   60.00   23.00
66.00   70.00   34.00
44.00   80.00   55.00
50.00   10.00   1000.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00
44.00   80.00   55.00
50.00   10.00   1000.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00


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

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00   1000.00 2000.00


Now removing the last observer from the list:

Adding a new value the end of the list:
```

```
john2@John-Laptop MINGW64 ~/Desktop/uofc/614/ENSF-614-assignment-repo/assignment5 (main)
$ java EXC.ObserverPatternController
Changing the third value from 33, to 66 -- (All views must show this change):

Notification to three-Column Table Observer: Data Changed:
10.00   20.00   66.00
44.00   50.00   30.00
60.00   70.00   80.00
10.00   11.00   23.00
34.00   55.00

Notification to five-rows Table Observer: Data Changed:
10.00   30.00   11.00
20.00   60.00   23.00
66.00   70.00   34.00
44.00   80.00   55.00
50.00   10.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   66.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00


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

Notification to three-Column Table Observer: Data Changed:
10.00   20.00   66.00
44.00   50.00   30.00
60.00   70.00   80.00
10.00   11.00   23.00
34.00   55.00   1000.00


Notification to five-rows Table Observer: Data Changed:
10.00   30.00   11.00
20.00   60.00   23.00
66.00   70.00   34.00
```

```
john2@John-Laptop MINGW64 ~/Desktop/uofc/614/ENSF-614-assignment-repo/assignment5 (main)
$ java EXC.ObserverPatternController
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:
10.00   20.00   33.00
44.00   50.00   30.00
60.00   70.00   80.00
10.00   11.00   23.00
34.00   55.00

Notification to five-rows Table Observer: Data Changed:
10.00   30.00   11.00
20.00   60.00   23.00
33.00   70.00   34.00
44.00   80.00   55.00
50.00   10.00

Notification to one-row Table Observer: Data Changed:
10.00   20.00   33.00   44.00   50.00   30.00   60.00   70.00   80.00   10.00   11.00   23.00   34.00   55.00
```

```
Now, creating a new Three-Column observer that will be notified of existing data:
Notification to three-Column Table Observer: Data Changed:


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:
10.00   20.00   66.00
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:
10.00   20.00   66.00
44.00   50.00   30.00
Now, creating a new Three-Column observer that will be notified of existing data:
Notification to three-Column Table Observer: Data Changed:
10.00   20.00   66.00
44.00   50.00   30.00
10.00   20.00   66.00
44.00   50.00   30.00
60.00   70.00   80.00
44.00   50.00   30.00
60.00   70.00   80.00
60.00   70.00   80.00
10.00   11.00   23.00
10.00   11.00   23.00
34.00   55.00   1000.00
2000.00 3000.00
```