**ENSF 408**

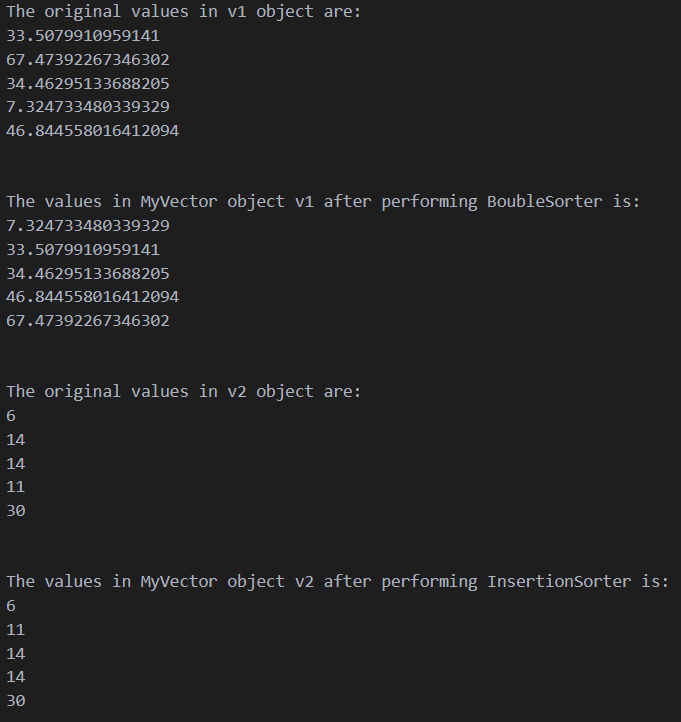
**Lab 5**

**Shamin Rahman (30037908)**

**18 oct 2019**

**B01**

**Output:**

****

**ExA:**

import java.util.ArrayList;

*// MyVector.java*

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);

        sorter = new BubbleSorter<E>();     *// as per notes, strategy is set as strategy1 by default*

        sorter.assignReference(this);

    }

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

        storageM = arr;

        sorter = new BubbleSorter<E>();     *// as per notes, strategy is set as strategy1 by default*

        sorter.assignReference(this);

    }

    public Item<E> get(int index){

        return storageM.get(index);

    }

    public void set(int index, Item<E> a){

        storageM.set(index, a);

    }

    public int size(){

        return storageM.size();

    }

    public void add(Item<E> value){

        storageM.add(value);

    }

    public void setSortStrategy(Sorter<E> s){

        sorter = s;

        sorter.assignReference(this);

    }

    public void performSort(){

        sorter.sort();

    }

    public void display(){

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

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

        }

        System.out.println();

    }

}

*// sorter.java*

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

    public void sort();

    public void assignReference(MyVector <E> contextReference);

}

*/\*\**

*\* BubbleSorter*

*\*/*

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

    MyVector<E> reference;

    @Override

    public void sort() {

        int  n = reference.size();

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

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

                if(reference.get(j).getItem().compareTo(reference.get(j + 1).getItem()) > 0){

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

                    Item<E> insert = reference.get(j + 1);

                    reference.set(j, insert);

                    reference.set(j + 1, temp);

                }

            }

        }

    }

    @Override

    public void assignReference(MyVector<E> contextReference) {

        reference = contextReference;

    }

}

*/\*\**

*\* InsertionSorter*

*\*/*

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

    MyVector<E> reference;

    @Override

    public void sort() {

        int  n = reference.size();

        for(int i = 1; i < n; ++i){

            Item<E> key = reference.get(i);

            int j = i - 1;

            while(j >= 0 && reference.get(j).getItem().compareTo(key.getItem()) > 0){

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

                reference.set(j + 1, temp);

                j = j - 1;

            }

            reference.set(j + 1, key);

        }

    }

    @Override

    public void assignReference(MyVector<E> contextReference) {

        reference = contextReference;

    }

}

**ExB:**

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

    MyVector<E> reference;

    @Override

    public void sort() {

        int  n = reference.size();

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

            int min = i;

            for(int j = i + 1; j < n; j++){

                if(reference.get(j).getItem().compareTo(reference.get(min).getItem()) < 0){

                    min = j;

                }

            }

            Item<E> temp = reference.get(min);

            Item<E> insert = reference.get(i);

            reference.set(min, insert);

            reference.set(i, temp);

        }

    }

    @Override

    public void assignReference(MyVector<E> contextReference) {

        reference = contextReference;

    }

}