Alessandra Schiavi Paz and Muhammed Umar Khan

**Principles of Software Design**

**B02**

**ENSF 480**

Lab 5

Instructor: M. Moussavi

October 28, 2024

# Exercise A

A screenshot of a computer program

Description automatically generated

## BubbleSorter.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    @Override

    public void sort(ArrayList<Item<E>> items) {

        int n = items.size();

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

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

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

                    // Swap

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

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

                    items.set(j + 1, temp);

                }

            }

        }

    }

}

## InsertionSorter.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    @Override

    public void sort(ArrayList<Item<E>> items) {

        int n = items.size();

        for (int i = 1; i < n; i++) {

            E key = items.get(i).getItem();

            int j = i - 1;

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

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

                j--;

            }

            items.set(j + 1, new Item<>(key));

        }

    }

}

## Sorter.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    void sort(ArrayList<Item<E>> items);

}

## MyVector.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    private ArrayList<Item<E>> storageM;

    private Sorter<E> sorter;

    // Constructor to initialize with capacity

    public MyVector(int n) {

        storageM = new ArrayList<>(n);

    }

    // Constructor to initialize with an ArrayList

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

        storageM = new ArrayList<>(arr);

    }

    // Add method to add an Item to storageM

    public void add(Item<E> value) {

        storageM.add(value);

    }

    // Set the sorting strategy

    public void setSortStrategy(Sorter<E> s) {

        this.sorter = s;

    }

    // Perform sorting using the assigned strategy

    public void performSort() {

        if (sorter != null) {

            sorter.sort(storageM);

        } else {

            System.out.println("No sorting strategy assigned.");

        }

    }

    // Display method to show contents of storageM

    public void display() {

        for (Item<E> item : storageM) {

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

        }

        System.out.println();

    }

}

# Exercise B

A screen shot of a computer

Description automatically generated

## BubbleSorter.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    @Override

    public void sort(ArrayList<Item<E>> items) {

        int n = items.size();

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

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

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

                    // Swap

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

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

                    items.set(j + 1, temp);

                }

            }

        }

    }

}

## InsertionSorter.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    @Override

    public void sort(ArrayList<Item<E>> items) {

        int n = items.size();

        for (int i = 1; i < n; i++) {

            E key = items.get(i).getItem();

            int j = i - 1;

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

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

                j--;

            }

            items.set(j + 1, new Item<>(key));

        }

    }

}

## MyVector.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    private ArrayList<Item<E>> storageM;

    private Sorter<E> sorter;

    // Constructor to initialize with capacity

    public MyVector(int n) {

        storageM = new ArrayList<>(n);

    }

    // Constructor to initialize with an ArrayList

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

        storageM = new ArrayList<>(arr);

    }

    // Add method to add an Item to storageM

    public void add(Item<E> value) {

        storageM.add(value);

    }

    // Set the sorting strategy

    public void setSortStrategy(Sorter<E> s) {

        this.sorter = s;

    }

    // Perform sorting using the assigned strategy

    public void performSort() {

        if (sorter != null) {

            sorter.sort(storageM);

        } else {

            System.out.println("No sorting strategy assigned.");

        }

    }

    // Display method to show contents of storageM

    public void display() {

        for (Item<E> item : storageM) {

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

        }

        System.out.println();

    }

}

## SelectionSorter.java

/\*

 \* Lab05 ExB

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    @Override

    public void sort(ArrayList<Item<E>> items) {

        int n = items.size();

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

            int min\_idx = i;

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

                if (items.get(j).getItem().compareTo(items.get(min\_idx).getItem()) < 0)

                    min\_idx = j;

            // Swap the found minimum element with the first element

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

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

            items.set(i, temp);

        }

    }

}

## Sorter.java

/\*

 \* Lab05 ExA

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

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

    void sort(ArrayList<Item<E>> items);

}

# Exercise C

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

## DoubleArrayListSubject.java

/\*

 \* Lab05 ExC

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

public class DoubleArrayListSubject implements Subject{

    private ArrayList<Double> data;

    private ArrayList<Observer> observers;

    public DoubleArrayListSubject(){

        this.data = new ArrayList<>();

        this.observers= new ArrayList<>();

    }

    public void remove(Observer o) {

        observers.remove(o);

    }

    @Override

    public void registerObserver(Observer o) {

        observers.add(o);

    }

    @Override

    public void removeObserver(Observer o) {

        observers.remove(o);

    }

    @Override

    public void notifyAllObservers() {

        for (Observer observer : observers) {

            observer.update(new ArrayList<>(data));

        }

    }

    public void addData(double value) {

        data.add(value);

        notifyAllObservers();

    }

    public void setData(double value, int index) {

        if (index >= 0 && index < data.size()) {

            data.set(index, value);

            notifyAllObservers();

        }

    }

    public void populate(double[] values) {

        for (double value : values) {

            data.add(value);

        }

        notifyAllObservers();

    }

    public void display() {

        if (data.isEmpty()) {

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

        } else {

            System.out.println(data);

        }

    }

}

## FiveRowsTable\_Observer.java

/\*

 \* Lab05 ExC

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

public class FiveRowsTable\_Observer implements Observer {

    private DoubleArrayListSubject subject;

    public FiveRowsTable\_Observer(DoubleArrayListSubject subject) {

        this.subject = subject;

        this.subject.registerObserver(this);

    }

    @Override

    public void update(ArrayList<Double> data) {

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

        display(data);

    }

    public void display(ArrayList<Double> data) {

        int rows = 5;

        int columns = (int) Math.ceil((double) (data.size()) / rows); // Calculate columns needed

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

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

            if ((i + 1) % columns == 0 || i == data.size() - 1) {

                System.out.println();

            }

        }

    }

}

## Observer.java

/\*

 \* Lab05 ExC

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

public interface Observer {

    void update(ArrayList<Double> data);

}

## OneRow\_Observer.java

/\*

 \* Lab05 ExC

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

public class OneRow\_Observer implements Observer {

    private DoubleArrayListSubject subject;

    public OneRow\_Observer(DoubleArrayListSubject subject) {

        this.subject = subject;

        this.subject.registerObserver(this);

    }

    @Override

    public void update(ArrayList<Double> data) {

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

        display(data);

    }

    public void display(ArrayList<Double> data) {

        for (Double value : data) {

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

        }

        System.out.println();

    }

}

## Subject.java

/\*

 \* Lab05 ExC

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

public interface Subject {

    void registerObserver(Observer o);

    void removeObserver(Observer o);

    void notifyAllObservers();

}

## ThreeColumnTable\_Observer.java

/\*

 \* Lab05 ExC

 \* Completed by: Alessandra Schiavi and Muhammed Umar Khan

 \* Submission Date: Oct 28, 2024

 \*/

import java.util.ArrayList;

public class ThreeColumnTable\_Observer implements Observer {

    private DoubleArrayListSubject subject;

    public ThreeColumnTable\_Observer(DoubleArrayListSubject subject) {

        this.subject = subject;

        this.subject.registerObserver(this);

    }

    @Override

    public void update(ArrayList<Double> data) {

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

        display(data);

    }

    public void display(ArrayList<Double> data) {

        int columns = 3;

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

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

            if ((i + 1) % columns == 0 || i == data.size() - 1) {

                System.out.println();

            }

        }

    }

}