ENSF 614

Advanced System Analysis and Software Design

LAB 6

Author:

Steven Duong
(30022492)

Affiliation
Department of Electrical and Software Engineering
University of Calgary
Calgary, Alberta

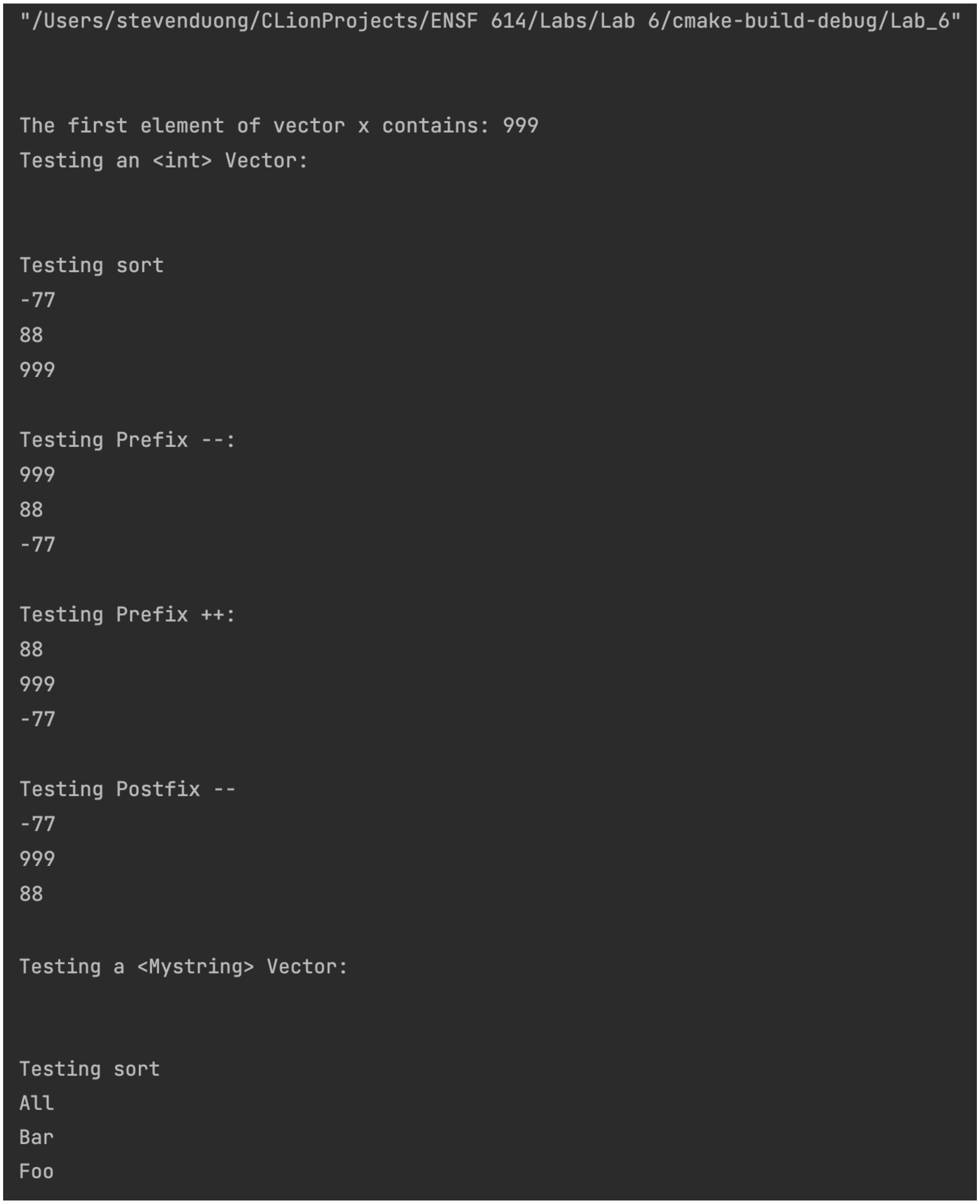
Lab Block: B01
Date of Report: Mar 22, 2023

Exercise A – Source File iterator.cpp

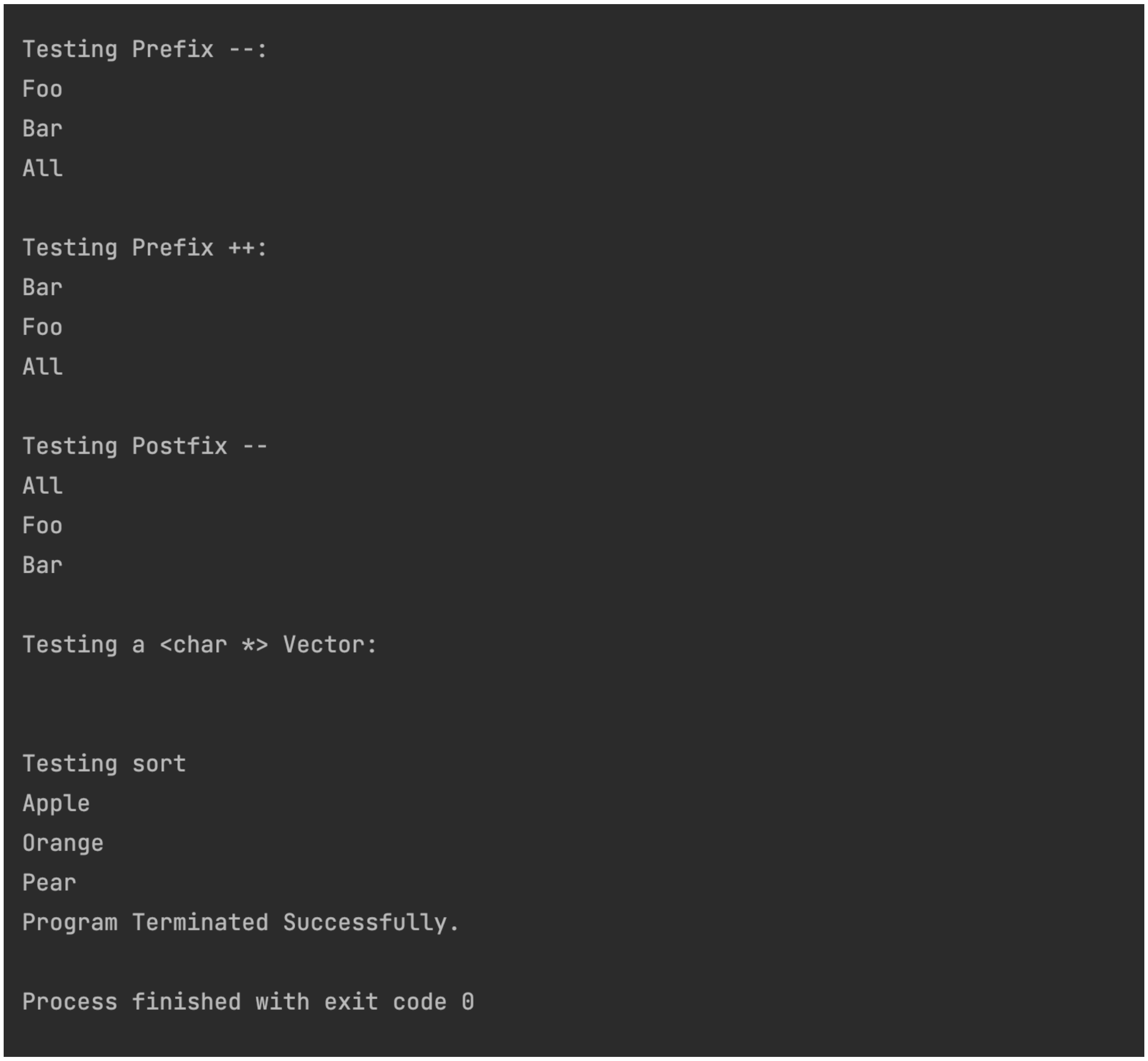
```
/*
 *  File Name: iterator.cpp
 *  Assignment: ENSF 614 Lab 6 Exercise A
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

#include <iostream>

#include <assert.h>

#include "mystring2.h"

using namespace std;

template < class T >
  class Vector {
    public:

      class VectIter {
        friend class Vector;
        private:
          Vector < T > * v; // points to a vector object of type T
        int index; // represents the subscript number of the vector's
        // array.
        public:
          VectIter(Vector < T > & x);

        T operator++();
        //PROMISES: increments the iterator's indices and return the
        //          value of the element at the index position. If
        //          index exceeds the size of the array it will
        //          be set to zero. Which means it will be circulated
        //          back to the first element of the vector.

        T operator++(int);
        // PROMISES: returns the value of the element at the index
        //           position, then increments the index. If
        //           index exceeds the size of the array it will
        //           be set to zero. Which means it will be circulated
        //           back to the first element of the vector.

        T operator--();
        // PROMISES: decrements the iterator index, and return
        //           the value of the element at the index. If
        //           index is less than zero it will be set to the
        //           last element in the array. Which means it will be
        //           circulated to the last element of the vector.

        T operator--(int);
        // PROMISES: returns the value of the element at the index
        //           position, then decrements the index. If
        //           index is less than zero it will be set to the
        //           last element in the aray. Which means it will be
        //           circulated to the last element of the vector.

        T operator * ();
        // PROMISES: returns the value of the element at the current
        //           index position.
      };

    Vector(int sz);
    ~Vector();

    T & operator[](int i);
    // PROMISES: returns existing value in the ith element of
    //           array or sets a new value to  the ith element in
    //           array.

    void ascending_sort();
    // PROMISES: sorts the vector values in ascending order.

    private: T * array; // points to the first element of an array of
T
    int size; // size of array
    void swap(T & , T & ); // swaps the values of two elements in
array
    public:
  };

template < typename T >
  void Vector < T > ::ascending_sort() {
    for (int i = 0; i < size - 1; i++)
      for (int j = i + 1; j < size; j++)
        if (array[i] > array[j])
          swap(array[i], array[j]);
  }

// Specialization of ascending_sort function for MyString type
template < >
  void Vector < Mystring > ::ascending_sort() {
    for (int i = 0; i < size - 1; i++) {
      for (int j = i + 1; j < size; j++) {
        if (strcmp(array[i].c_str(), array[j].c_str()) > 0) {
          swap(array[i], array[j]);
        }
      }
    }
  }

// Specialization of ascending_sort function for const char∗ type
template < >
  void Vector <
  const char ∗ > ::ascending_sort() {
    for (int i = 0; i < size − 1; i++) {
      for (int j = i + 1; j < size; j++) {
        if (strcmp(array[i], array[j]) > 0) {
          swap(array[i], array[j]);
        }
      }
    }
  }

template < typename T >
  void Vector < T > ::swap(T & a, T & b) {
    T tmp = a;
    a = b;
    b = tmp;
  }

template < typename T >
  T Vector < T > ::VectIter::operator ∗ () {
    return v −> array[index];
  }

template < typename T >
  Vector < T > ::VectIter::VectIter(Vector & x) {
    v = & x;
    index = 0;
  }

template < typename T >
  Vector < T > ::Vector(int sz) {
    size = sz;
    array = new T[sz];
    assert(array != NULL);
  }

template < typename T >
  Vector < T > ::~Vector() {
    delete[] array;
    array = NULL;
  }

template < typename T >
  T & Vector < T > ::operator[](int i) {
    return array[i];
  }

template < typename T >
  T Vector < T > ::VectIter::operator++() {
    index++;
    if (index >= v -> size) {
      index = 0;
    }
    return v -> array[index];
  }

template < typename T >
  T Vector < T > ::VectIter::operator++(int) {
    T currentVal = v -> array[index];
    index++;
    if (index >= v -> size) {
      index = 0;
    }
    return currentVal;
  }

template < typename T >
  T Vector < T > ::VectIter::operator--() {
    index--;
    if (index < 0) {
      index = v -> size - 1;
    }
    return v -> array[index];
  }

template < typename T >
  T Vector < T > ::VectIter::operator--(int) {
    T currentVal = v -> array[index];
    index--;
    if (index < 0) {
      index = v -> size - 1;
    }
    return currentVal;
  }

int main() {

  Vector < int > x(3);
  x[0] = 999;
  x[1] = -77;
  x[2] = 88;

  Vector < int > ::VectIter iter(x);
```

```
cout << "\n\nThe first element of vector x contains: " << * iter;

// the code between the  #if 0 and #endif is ignored by
// compiler. If you change it to #if 1, it will be compiled

#if 1
cout << "\nTesting an <int> Vector: " << endl;

cout << "\n\nTesting sort";
x.ascending_sort();

for (int i = 0; i < 3; i++)
  cout << endl << iter++;

cout << "\n\nTesting Prefix --:";
for (int i = 0; i < 3; i++)
  cout << endl << --iter;

cout << "\n\nTesting Prefix ++:";
for (int i = 0; i < 3; i++)
  cout << endl << ++iter;

cout << "\n\nTesting Postfix --";
for (int i = 0; i < 3; i++)
  cout << endl << iter--;

cout << endl << endl;

cout << "Testing a <Mystring> Vector: " << endl;
Vector < Mystring > y(3);
y[0] = "Bar";
y[1] = "Foo";
y[2] = "All";;

Vector < Mystring > ::VectIter iters(y);

cout << "\n\nTesting sort";
y.ascending_sort();

for (int i = 0; i < 3; i++)
  cout << endl << iters++.c_str();

cout << "\n\nTesting Prefix --:";
for (int i = 0; i < 3; i++)
  cout << endl << (--iters).c_str();

cout << "\n\nTesting Prefix ++:";
for (int i = 0; i < 3; i++)
  cout << endl << (++iters).c_str();
  cout << "\n\nTesting Postfix --";
  for (int i = 0; i < 3; i++)
    cout << endl << iters--.c_str();

  cout << endl << endl;
  cout << "Testing a <char *> Vector: " << endl;
  Vector <
    const char * > z(3);
  z[0] = "Orange";
  z[1] = "Pear";
  z[2] = "Apple";;

  Vector <
    const char * > ::VectIter iterchar(z);

  cout << "\n\nTesting sort";
  z.ascending_sort();

  for (int i = 0; i < 3; i++)
    cout << endl << iterchar++;

  #endif
  cout << "\nProgram Terminated Successfully." << endl;

  return 0;
}
```

Exercise A – Program Output



```
"/Users/stevenduong/CLionProjects/ENSF 614/Labs/Lab 6/cmake-build-debug/Lab_6"


The first element of vector x contains: 999
Testing an <int> Vector:


Testing sort
-77
88
999

Testing Prefix --:
999
88
-77

Testing Prefix ++:
88
999
-77

Testing Postfix --
-77
999
88

Testing a <Mystring> Vector:


Testing sort
All
Bar
Foo
```


```
Testing Prefix --:
Foo
Bar
All

Testing Prefix ++:
Bar
Foo
All

Testing Postfix --
All
Foo
Bar

Testing a <char *> Vector:


Testing sort
Apple
Orange
Pear
Program Terminated Successfully.

Process finished with exit code 0
```

Exercise B – Item.java

```java
/*
 *  File Name: Item.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

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;
    }
}
```

Exercise B – Sorter.java

```java
/*
 *  File Name: Sorter.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

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

Exercise B – BubbleSorter.java

```java
/*
 *  File Name: BubbleSorter.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

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

Exercise B – InsertionSorter.java

```
/*
 *  File Name: InsertionSorter.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

class InsertionSorter<E extends Number & Comparable<E>> implements 
Sorter<E> {
    @Override
    public void sort(ArrayList<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().compareTo(key.getItem()) > 0) {
                list.set(j + 1, list.get(j));
                j = j - 1;
            }
            list.set(j + 1, key);
        }
    }
}
```

Exercise B – MyVector.java

```
/*
 *  File Name: MyVector.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

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

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

    MyVector(ArrayList<Item<E>> arr) {
        this.storageM = new ArrayList<>(arr.size());

        for (int i = 0; i < arr.size(); i++) {
            this.storageM.add(arr.get(i));
        }
    }

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

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

    public void performSort() {
        this.sorter.sort(this.storageM);
    }

    public void display() {
        for (int i = 0; i < storageM.size(); i++) {
            E value = storageM.get(i).getItem();
            if (value instanceof Integer) {
                System.out.print(value.intValue() + " ");
            } else {
                System.out.print(String.format("%.2f",
value.doubleValue()) + " ");
            }
        }
        System.out.println();
    }

}
```

Exercise B – DemoStrategyPattern.java

```
/*
 *  File Name: DemoStrategyPattern.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

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 BubbleSorter 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();

        // create a MyVector<Integer> object v3
        MyVector<Integer> v3 = new MyVector<Integer>(50);

        // populate v3 with 5 randomly generated numbers
        for (int i = 7; i >= 0; i--) {
            Item<Integer> item;
            item = new
Item<Integer>(Integer.valueOf(rand.nextInt(100)));
            v3.add(item);
        }

        System.out.println("\nThe original values in v3 object are:");
        v3.display();

        // Set the sort strategy for v3 to use SelectionSorter
        v3.setSortStrategy(new SelectionSorter<Integer>());

        // Perform the sort on v3
        v3.performSort();

        System.out.println("\nThe values in MyVector object v3 after
performing SelectionSorter is:");
        v3.display();

    }
}
```

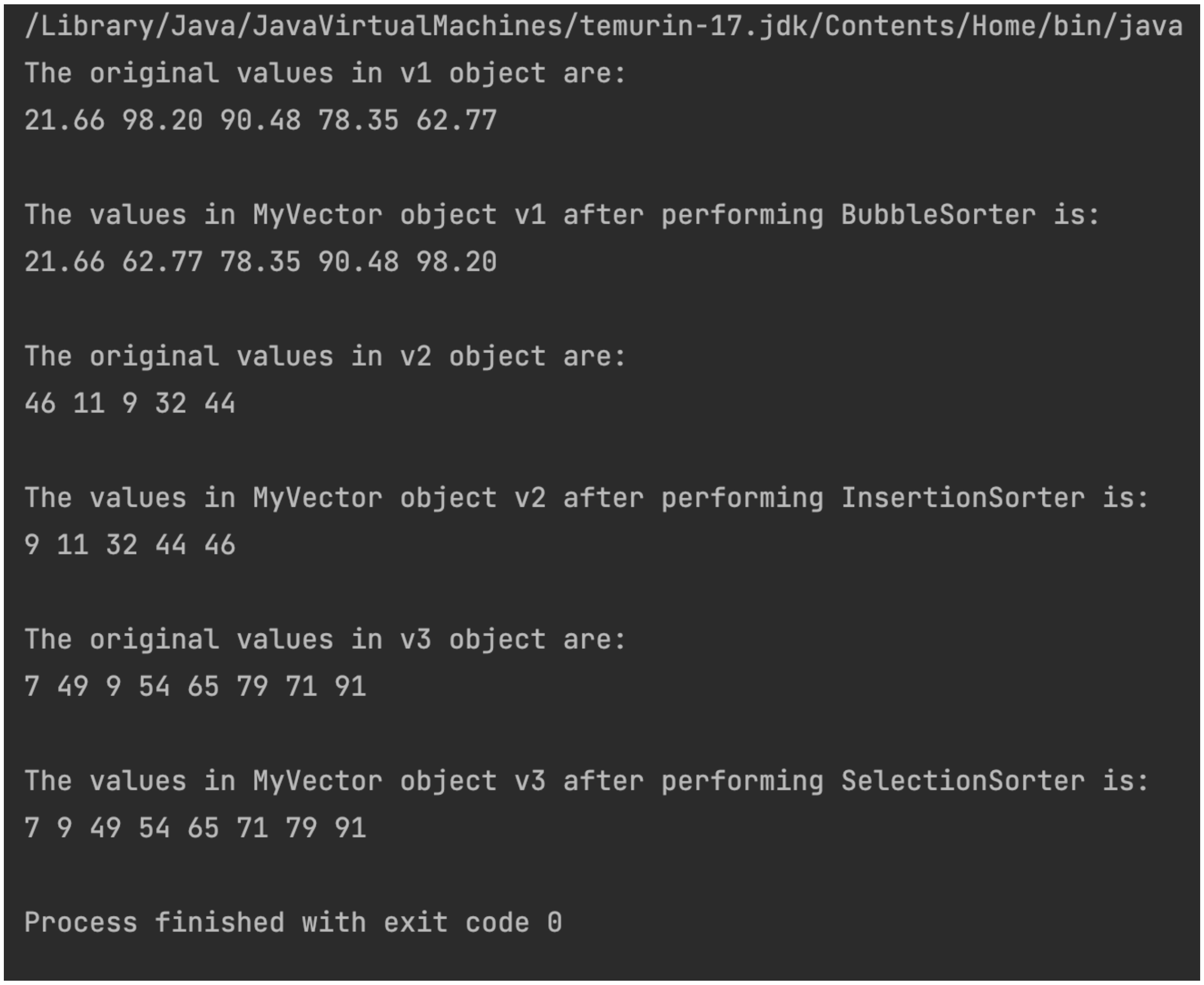
Exercise C – SelectionSorter.java

```
/*
 *  File Name: SelectionSorter.java
 *  Assignment: ENSF 614 Lab 6 Exercise B
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
```

```java
 */

import java.util.ArrayList;

// SelectionSorter class
class SelectionSorter<E extends Number & Comparable<E>> implements 
Sorter<E> {
    @Override
    public void sort(ArrayList<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);
            }
        }
    }
}
```

```java
}
```

Exercise B and C – Program Output



```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java
The original values in v1 object are:
21.66 98.20 90.48 78.35 62.77

The values in MyVector object v1 after performing BubbleSorter is:
21.66 62.77 78.35 90.48 98.20

The original values in v2 object are:
46 11 9 32 44

The values in MyVector object v2 after performing InsertionSorter is:
9 11 32 44 46

The original values in v3 object are:
7 49 9 54 65 79 71 91

The values in MyVector object v3 after performing SelectionSorter is:
7 9 49 54 65 71 79 91

Process finished with exit code 0
```

Exercise D – Observer.java

```java
/*
 *  File Name: Observer.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

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

Exercise D – Subject.java

```java
/*
 *  File Name: Subject.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

public interface Subject {
    public void registerObserver(Observer o);
    public void remove(Observer o);
    public void notifyAllObservers();
}
```

Exercise D – DoubleArrayListSubject.java

```java
/*
 *  File Name: DoubleArrayListSubject.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

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

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

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

    @Override
    public void remove(Observer o) {
        this.observers.remove(o);
    }
    @Override
    public void notifyAllObservers() {
        for (int i = 0; i < this.observers.size(); i++) {
            Observer o = this.observers.get(i);
            o.update(this.data);
        }
    }

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

    public void setData(Double value, int index) {
        this.data.set(index, value);
        notifyAllObservers();
    }

    public void populate(double[] arr) {
        for (int i = 0; i < arr.length; i++) {
            this.data.add(arr[i]);
        }

        notifyAllObservers();
    }

    public void display() {
        if (this.data.size() == 0) {
            System.out.println("Empty list ...");
        }

        for (int i = 0; i < this.data.size(); i++) {
            System.out.printf("%.1f ", this.data.get(i));
        }

        System.out.println();
    }

}
```

Exercise D – FiveRowsTable_Observer.java

```
/*
 *  File Name: FiveRowsTable_Observer.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

public class FiveRowsTable_Observer implements Observer {
    private Subject subject;
    private ArrayList<Double> arr;

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

    @Override
    public void update(ArrayList<Double> arr) {
        System.out.println("\nNotification to Five-Rows Table
Observer: Data Changed:");
        this.arr = arr;
        this.display();
    }

    public void display() {
        for (int i = 0; i < 5; i++) {
            for (int j = i; j < arr.size(); j+=5) {
                System.out.printf("%.1f\t", arr.get(j));
            }
            System.out.println();
        }
        System.out.println();
    }
}
```

Exercise D – ThreeColumnTable_Observer.java

```
/*
 *  File Name: ThreeColumnTable_Observer.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

public class ThreeColumnTable_Observer implements Observer {
    private Subject subject;
    private ArrayList<Double> arr;
    public ThreeColumnTable_Observer(Subject s) {
        this.subject = s;
        this.subject.registerObserver(this);
    }

    @Override
    public void update(ArrayList<Double> arr) {
        System.out.print("\nNotification to Three-Column Table 
Observer: Data Changed:");
        this.arr = arr;
        this.display();
    }

    public void display() {
        for (int i = 0; i < arr.size(); i++) {
            if (i % 3 == 0) {
                System.out.println();
            }
            System.out.printf("%.1f\t", this.arr.get(i));
        }
        System.out.println();
    }
}
```

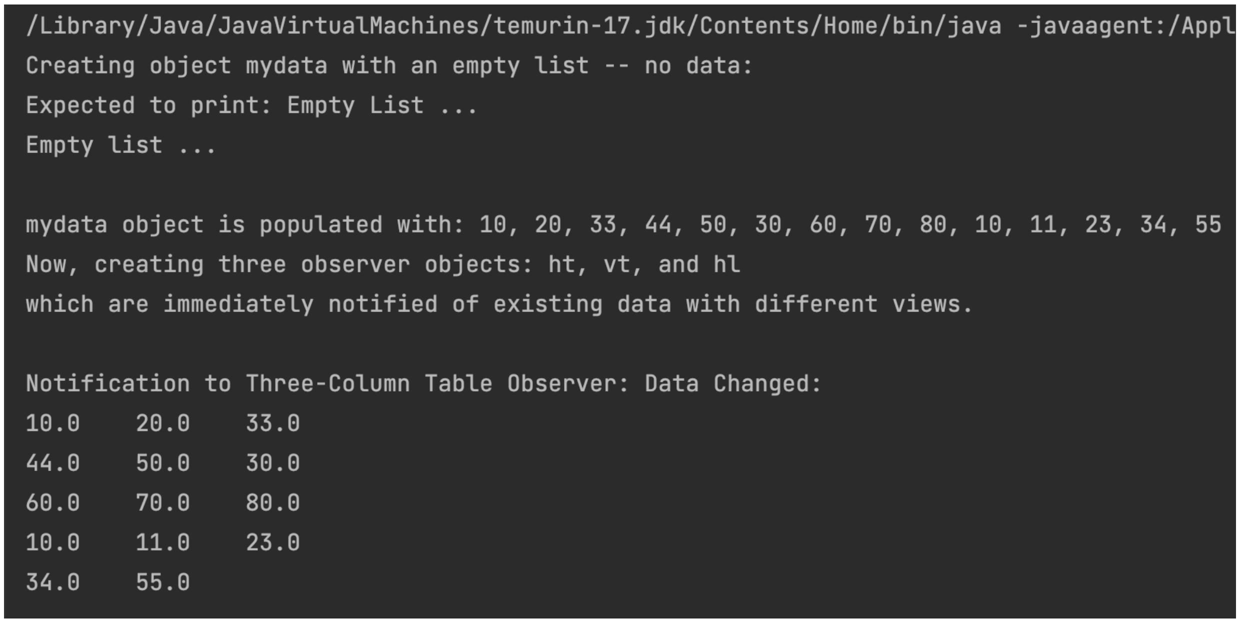
Exercise D – OneRow_Observer.java

```
/*
 *  File Name: OneRow_Observer.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.util.ArrayList;

public class OneRow_Observer implements Observer {
    private Subject subject;
    private ArrayList<Double> arr;

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

    @Override
    public void update(ArrayList<Double> arr) {
        System.out.println("\nNotification to One-Row Observer: Data
Changed:");
        this.arr = arr;
        this.display();
    }

    public void display() {
        for (int i = 0; i < arr.size(); i++) {
            System.out.printf("%.1f  ", arr.get(i));
        }
        System.out.println();
    }
}
```

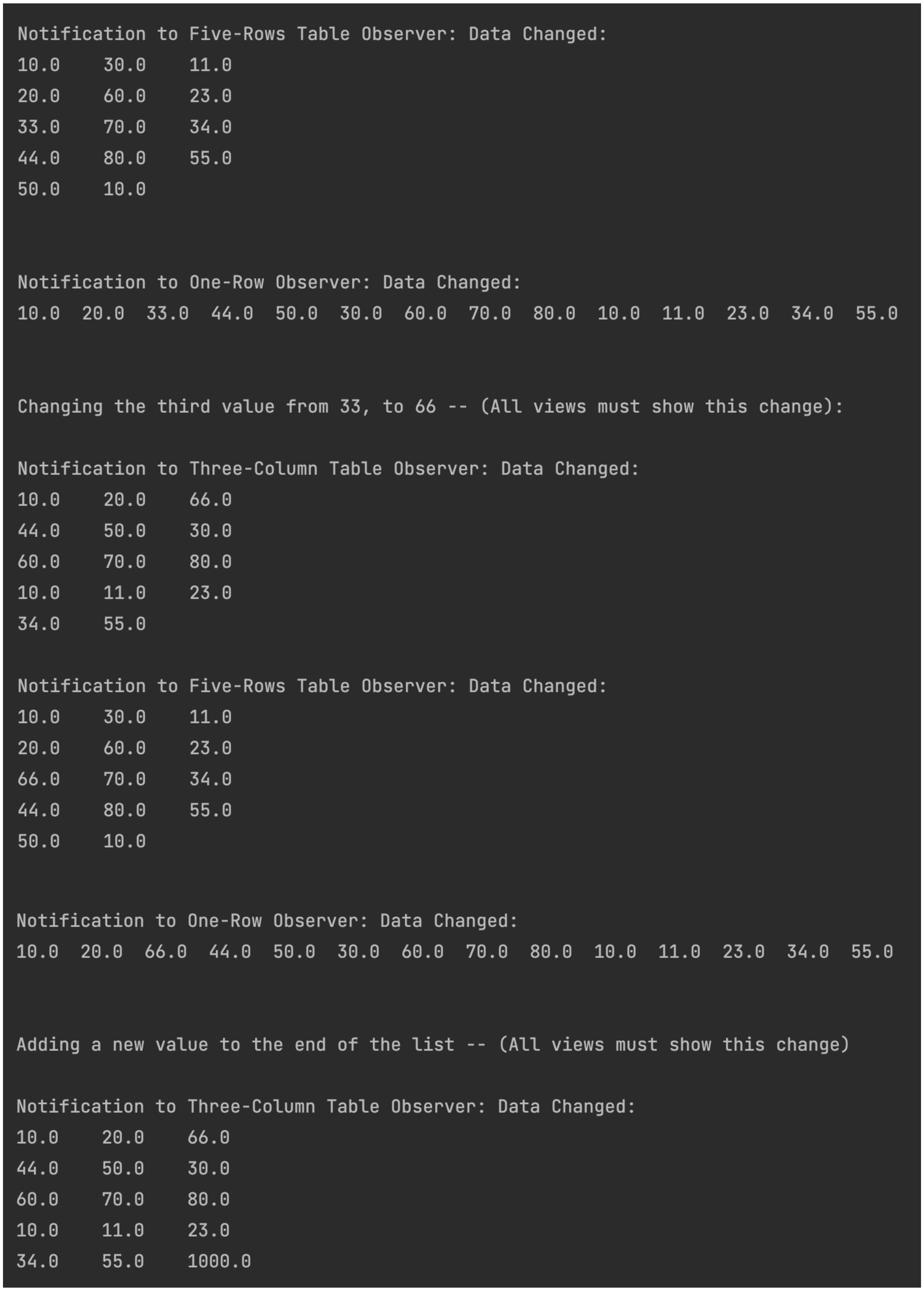
Exercise D – ObserverPatternController.java

```
/*
 *  File Name: ObserverPatternController.java
 *  Assignment: ENSF 614 Lab 6 Exercise D
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

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);
    }
}
```

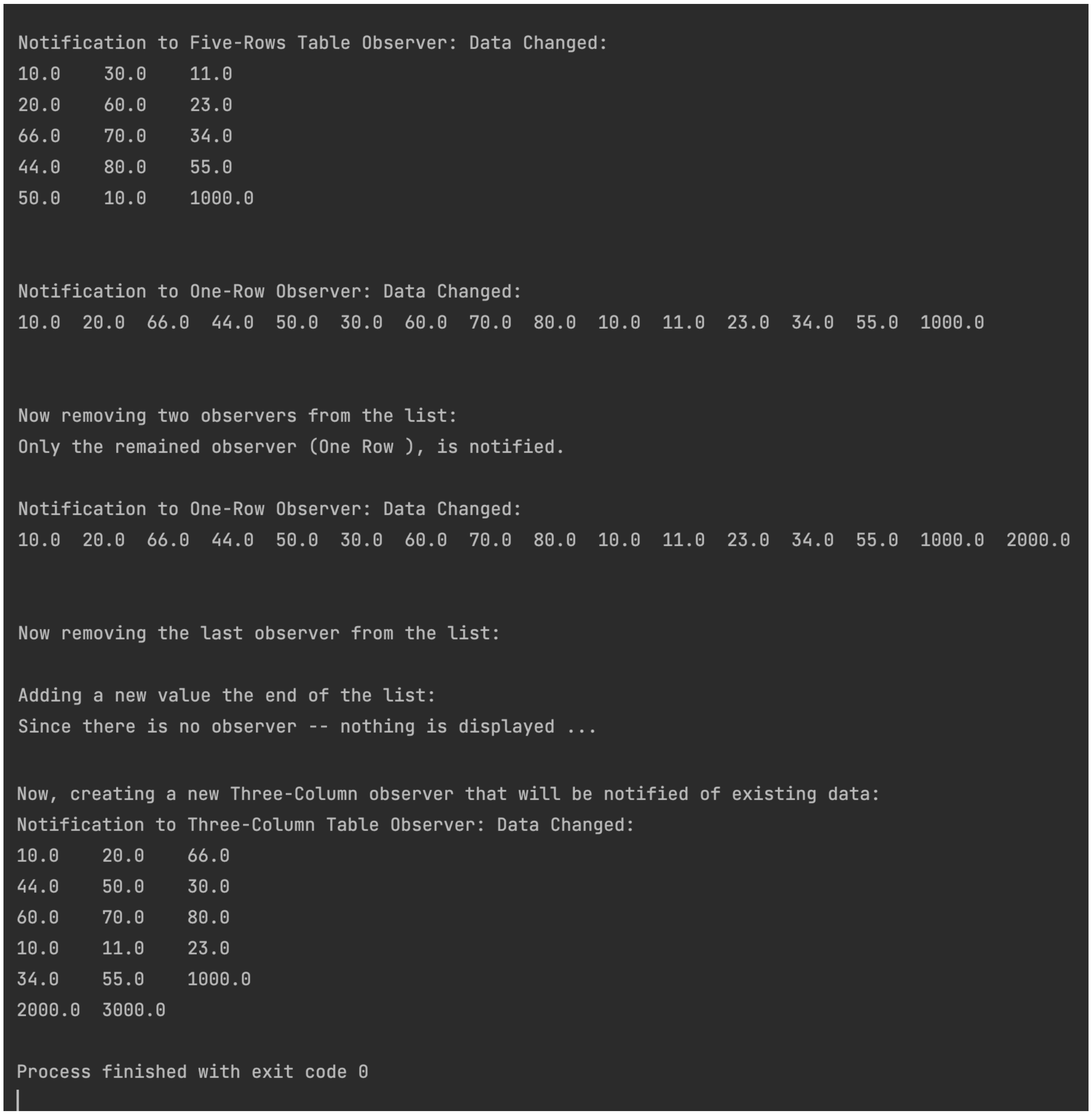
Exercise D – Program Output



```
/Library/Java/JavaVirtualMachines/temurin-17.jdk/Contents/Home/bin/java -javaagent:/Appl
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

Process finished with exit code 0
```

Exercise E – Component.java

```
/*
 *  File Name: Component.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */
```

```java
import java.awt.*;

public interface Component {
    public void draw(Graphics g);
}
```

Exercise E – Decorator.java

```java
/*
 *  File Name: Decorator.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

abstract class Decorator implements Component {
    protected Component cmp;
    protected int x;
    protected int y;
    protected int width;
    public int height;

    public Decorator(Component cmp, int x, int y, int width, int
height) {
        this.cmp = cmp;
        this.x = x;
        this.y = y;
        this.width = width;
        this.height = height;
    }

}
```

Exercise E – BorderDecorator.java

```java
/*
 *  File Name: BorderDecorator.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.awt.*;

public class BorderDecorator extends Decorator {
    public BorderDecorator(Component cmp, int x, int y, int width, int
height) {
        super(cmp, x, y, width, height);
    }

    @Override
    public void draw(Graphics g) {
        this.cmp.draw(g);
        Stroke dashed = new BasicStroke(3, BasicStroke.CAP_BUTT,
BasicStroke.JOIN_BEVEL, 0, new float[]{9},
                0);
        Graphics2D g2d = (Graphics2D) g;
        g2d.setStroke(dashed);
        Color oldColor = g2d.getColor();
        g2d.drawRect(x, y, width, height);
        g2d.setColor(oldColor);
    }
}
```

Exercise E – ColouredFrameDecorator.java

```java
/*
 *  File Name: ColouredFrameDecorator.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.awt.*;

public class ColouredFrameDecorator extends Decorator {

    protected int thickness;

    public ColouredFrameDecorator(Component cmp, int x, int y, int
width, int height, int thickness) {
        super(cmp, x, y, width, height);
        this.thickness = thickness;
    }

    @Override
    public void draw(Graphics g) {
        this.cmp.draw(g);
        Graphics2D g2d = (Graphics2D) g;
        Stroke oldStroke = g2d.getStroke();
        Color oldColor = g2d.getColor();
        g2d.setStroke(new BasicStroke(thickness));
        g2d.setColor(Color.red);
        g2d.drawRect(x, y, width, height);
        g2d.setStroke(oldStroke);
        g2d.setColor(oldColor);
    }
}
```

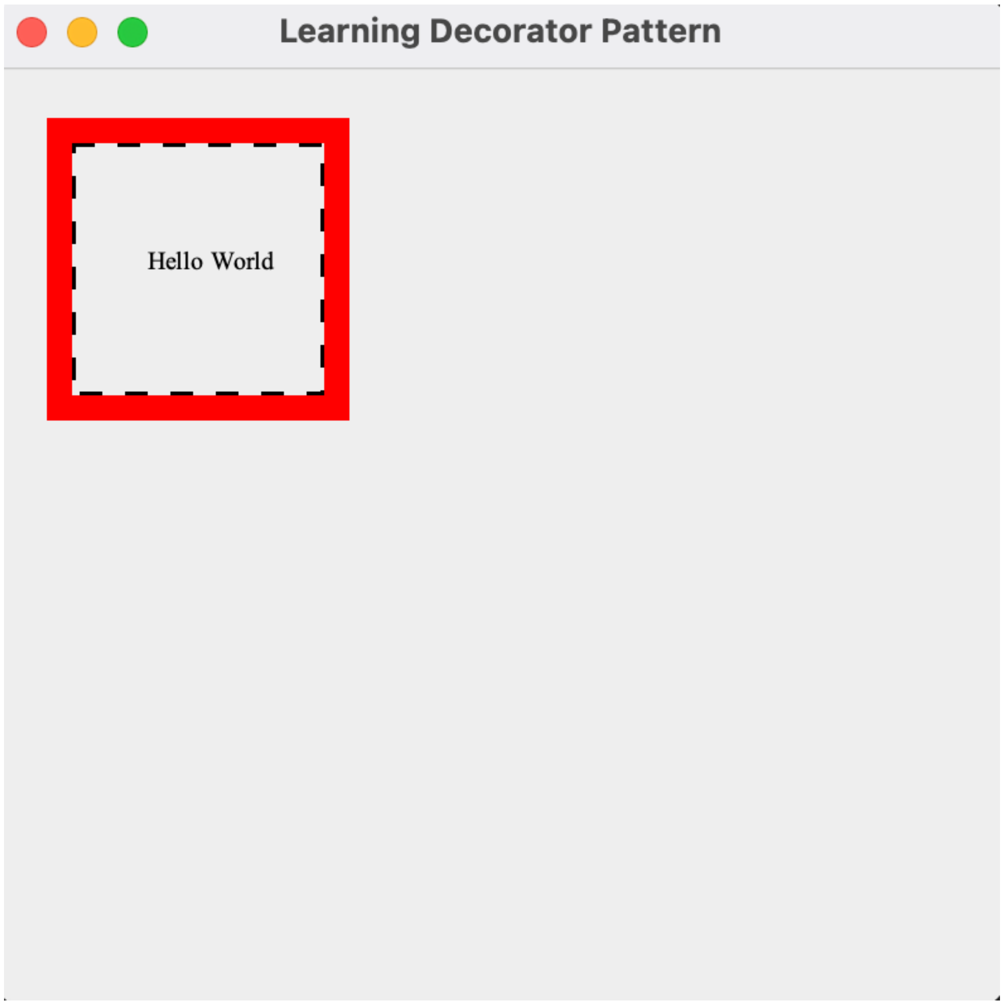
Exercise E – Text.java

```java
/*
 *  File Name: Text.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.awt.*;

public class Text implements Component {
    protected int x;
    protected int y;
    protected String text;

    public Text(String text, int x, int y) {
        this.x = x;
        this.y = y;
        this.text = text;
    }

    @Override
    public void draw(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        int fontSize = 10;
        g.setFont(new Font("TimesRoman", Font.PLAIN, fontSize));
        g2d.drawString(this.text, this.x, this.y);
    }
}
```

Exercise E – DemoDecoratorPattern.java

```java
/*
 *  File Name: DemoDecoratorPattern.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.awt.Font;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class DemoDecoratorPattern extends JPanel {
    Component t;

    public DemoDecoratorPattern(){
        t = new Text ("Hello World", 60, 80);
    }

    public void paintComponent(Graphics g){
        int fontSize = 10;
        g.setFont(new Font("TimesRoman", Font.PLAIN, fontSize));

        // Now let's decorate t with BorderDecorator: x = 30, y = 30,
width = 100, and height 100
        t = new BorderDecorator(t, 30, 30, 100, 100);

        // Now let's add a ColouredFrameDecorator with x = 25, y = 25,
width = 110, height = 110,
        // and thickness = 10.
        t = new ColouredFrameDecorator(t, 25, 25, 110, 110, 10);

        // Now lets draw the product on the screen
        t.draw(g);
    }

    public static void main(String[] args) {
        DemoDecoratorPattern panel = new DemoDecoratorPattern();
        JFrame frame = new JFrame("Learning Decorator Pattern");
        frame.getContentPane().add(panel);
        frame.setSize(400,400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

Exercise E – Program Output



Hello World

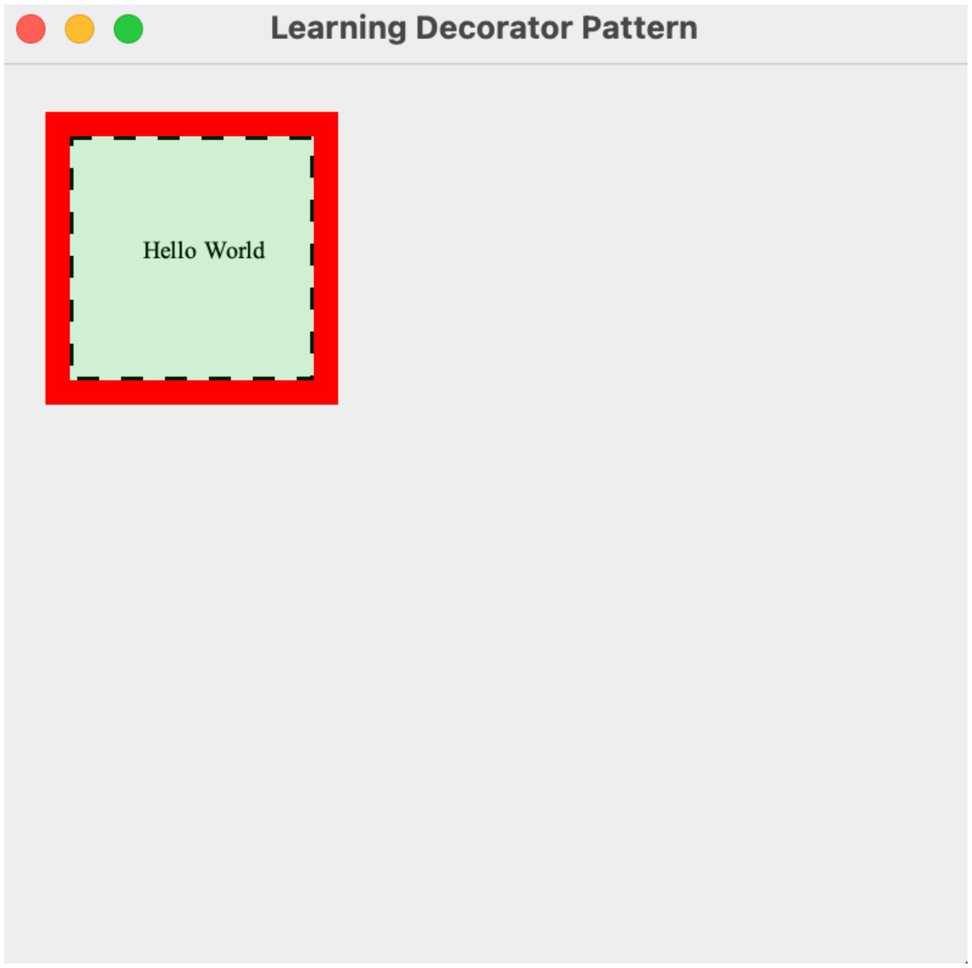
Exercise F – ColouredGlassDecorator.java

```
/*
 *  File Name: ColouredGlassDecorator.java
 *  Assignment: ENSF 614 Lab 6 Exercise F
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.awt.*;

public class ColouredGlassDecorator extends Decorator {

    public ColouredGlassDecorator(Component cmp, int x, int y, int
width, int height) {
        super(cmp, x, y, width, height);
    }

    @Override
    public void draw(Graphics g) {
        this.cmp.draw(g);
        Graphics2D g2d = (Graphics2D) g;
        g2d.setColor(Color.green);

g2d.setComposite(AlphaComposite.getInstance(AlphaComposite.SRC_OVER, 1
* 0.1f));
        g2d.fillRect(25, 25, 110, 110);
    }
}
```

Exercise F – DemoDecoratorPattern.java

```
/*
 *  File Name: DemoDecoratorPattern.java
 *  Assignment: ENSF 614 Lab 6 Exercise E
 *  Lab Section: Lab B01
 *  Completed by: Steven Duong (30022492)
 *  Submission Date: Mar 22, 2023
 */

import java.awt.Font;
import java.awt.Graphics;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class DemoDecoratorPattern extends JPanel {
    Component t;

    public DemoDecoratorPattern(){
        t = new Text ("Hello World", 60, 80);
    }

    public void paintComponent(Graphics g){
        int fontSize = 10;
        g.setFont(new Font("TimesRoman", Font.PLAIN, fontSize));
    // GlassFrameDecorator info: x = 25, y = 25, width = 110, and
height = 110
        t = new ColouredGlassDecorator(new ColouredFrameDecorator(
                new BorderDecorator(t, 30, 30, 100, 100), 25, 25, 110,
110, 10), 25, 25,
                110, 110);
        t.draw(g);
    }
    public static void main(String[] args) {
        DemoDecoratorPattern panel = new DemoDecoratorPattern();
        JFrame frame = new JFrame("Learning Decorator Pattern");
        frame.getContentPane().add(panel);
        frame.setSize(400,400);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setLocationRelativeTo(null);
        frame.setVisible(true);
    }
}
```

Exercise F – Program Output


Learning Decorator Pattern
Hello World