Course: ENSF 614 – Fall 2023

Lab 6:

Instructor: M. Moussavi

Student Name: Emmanuel Alafonye

Submission Date: November 10, 2023.

/\*

\* iterator.cpp

\* File Name: lab6Exe\_A.cpp

\* Assignment: ENSF 614 Lab 6, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex A

\* Submission Date: 10 November, 2023.

\*/

#include <iostream>

#include <algorithm> // Usde for sorting

#include <assert.h>

#include "mystring2.h" // Imported for Mystring

using namespace std;

// Define a template class Vector that can create vectors of different data types.

template <typename T>

class Vector

{

public:

// Nested class VectIter to represent an iterator for the Vector.

class VectIter

{

friend class Vector;

private:

Vector \*v; //Pointer to the parent object

int index; // Represent the subscript number of the vectors

public:

// Constrcutor of the VectorIter

VectIter(Vector &x) : v(&x), index(0) {}

T operator++()

{

if (++index == v->size)

index = 0;

return v->array[index];

}

T operator++(int)

{

T temp = v->array[index++];

if (index == v->size)

index = 0;

return temp;

}

T operator--()

{

if (index == 0)

index = v->size - 1;

else

index--;

return v->array[index];

}

T operator--(int)

{

T temp = v->array[index];

if (index == 0)

index = v->size - 1;

else

index--;

return temp;

}

T operator\*()

{

return v->array[index];

}

};

// Constructor for Vector class.

Vector(int sz) : size(sz)

{

array = new T[sz];

}

// Destructor for Vector class.

~Vector()

{

delete[] array;

}

// Overload the subscript operator ([]).

T &operator[](int i)

{

return array[i];

}

// Sort the vector's elements in ascending order using the STL sort function.

void ascending\_sort()

{

sort(array, array + size);

}

private:

T \*array; // Pointer to the first element of the array

int size; // Array size

};

// Main entry of the program

int main()

{

Vector<int> x(3);

x[0] = 999;

x[1] = -77;

x[2] = 88;

// Creates and iterator for the integer Vector

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

cout << "Testing an <int> Vector:" << endl;

cout << "Testing sort" << endl;

x.ascending\_sort();

for (int i = 0; i < 3; i++)

cout << iter++ << endl;

cout << "Testing Prefix --:" << endl;

for (int i = 0; i < 3; i++)

cout << --iter << endl;

cout << "Testing Prefix ++:" << endl;

for (int i = 0; i < 3; i++)

cout << ++iter << endl;

cout << "Testing Postfix --:" << endl;

for (int i = 0; i < 3; i++)

cout << iter-- << endl;

cout << "Program Terminated Successfully." << endl;

return 0;

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_A.cpp

\* Assignment: ENSF 614 Lab 6, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex A

\* Submission Date: 10 November, 2023.

\*/

#include "mystring2.h"

#include <string.h>

#include <iostream>

using namespace std;

Mystring::Mystring()

{

charsM = new char[1];

charsM[0] = '\0';

lengthM = 0;

}

Mystring::Mystring(const char \*s)

: lengthM(strlen(s))

{

charsM = new char[lengthM + 1];

strcpy(charsM, s);

}

Mystring::Mystring(int n)

: lengthM(0), charsM(new char[n])

{

charsM[0] = '\0';

}

Mystring::Mystring(const Mystring& source):

lengthM(source.lengthM), charsM(new char[source.lengthM+1])

{

strcpy (charsM, source.charsM);

}

Mystring::~Mystring()

{

delete [] charsM;

}

int Mystring::length() const

{

return lengthM;

}

char Mystring::get\_char(int pos) const

{

if(pos < 0 && pos >= length()){

cerr << "\nERROR: get\_char: the position is out of boundary." ;

}

return charsM[pos];

}

const char \* Mystring::c\_str() const

{

return charsM;

}

void Mystring::set\_char(int pos, char c)

{

if(pos < 0 && pos >= length()){

cerr << "\nset\_char: the position is out of boundary."

<< " Nothing was changed.";

return;

}

if (c != '\0'){

cerr << "\nset\_char: char c is empty."

<< " Nothing was changed.";

return;

}

charsM[pos] = c;

}

Mystring& Mystring::operator =(const Mystring& S)

{

if(this == &S)

return \*this;

delete [] charsM;

lengthM = (int) strlen(S.charsM);

charsM = new char [lengthM+1];

strcpy(charsM,S.charsM);

return \*this;

}

Mystring& Mystring::append(const Mystring& other)

{

char \*tmp = new char [lengthM + other.lengthM + 1];

lengthM+=other.lengthM;

strcpy(tmp, charsM);

strcat(tmp, other.charsM);

delete []charsM;

charsM = tmp;

return \*this;

}

void Mystring::set\_str(char\* s)

{

delete []charsM;

lengthM = (int) strlen(s);

charsM=new char[lengthM+1];

strcpy(charsM, s);

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_A.cpp

\* Assignment: ENSF 614 Lab 6, exercise A

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex A

\* Submission Date: 10 November, 2023.

\*/

#ifndef MYSTRING\_H

#define MYSTRING\_H

class Mystring {

public:

Mystring();

// PROMISES: Empty string object is created.

Mystring(int n);

// PROMISES: Creates an empty string with a total capacity of n.

// In other words, dynamically allocates n elements for

// charsM,sets the lengthM to zero, and fills the first

// element of charsM with '\0'.

Mystring(const char \*s);

// REQUIRES: s points to first char of a built-in string.

// REQUIRES: Mystring object is created by copying chars from s.

~Mystring(); // destructor

Mystring(const Mystring& source); // copy constructor

Mystring& operator =(const Mystring& rhs); // assignment operator

// REQUIRES: rhs is reference to a Mystring as a source

// PROMISES: to make this-object (object that this is pointing to, as a copy

// of rhs.

int length() const;

// PROMISES: Return value is number of chars in charsM.

char get\_char(int pos) const;

// REQUIRES: pos >= 0 && pos < length()

// PROMISES:

// Return value is char at position pos.

// (The first char in the charsM is at position 0.)

const char \* c\_str() const;

// PROMISES:

// Return value points to first char in built-in string

// containing the chars of the string object.

void set\_char(int pos, char c);

// REQUIRES: pos >= 0 && pos < length(), c != '\0'

// PROMISES: Character at position pos is set equal to c.

Mystring& append(const Mystring& other);

// PROMISES: extends the size of charsM to allow concatenate other.charsM to

// to the end of charsM. For example if charsM points to "ABC", and

// other.charsM points to XYZ, extends charsM to "ABCXYZ".

//

void set\_str(char\* s);

// REQUIRES: s is a valid C++ string of characters (a built-in string)

// PROMISES:copys s into charsM, if the length of s is less than or equal lengthM.

// Othrewise, extends the size of the charsM to s.lengthM+1, and copies

// s into the charsM.

private:

int lengthM; // the string length - number of characters excluding \0

char\* charsM; // a pointer to the beginning of an array of characters, allocated dynamically.

void memory\_check(char\* s);

// PROMISES: if s points to NULL terminates the program.

};

#endif

A screen shot of a computer

Description automatically generated

***Exercise B***

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

import java.util.Random;

public class DemoStrategyPattern {

public static void main(String[] args) {

// Create an object of MyVector<Double> with a capacity of 50 elements

MyVector<Double> v1 = new MyVector<>(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 = new Item<>(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 the bubble sort algorithm as a strategy to sort object v1

v1.setSortStrategy(new BubbleSorter<>());

// Perform algorithm bubble sort on v1

v1.performSort();

System.out.println("\nThe values in MyVector object v1 after performing BubbleSorter are:");

v1.display();

// Create a MyVector<Integer> object v2

MyVector<Integer> v2 = new MyVector<>(50);

// Populate v2 with 5 randomly generated numbers

for (int i = 4; i >= 0; i--) {

Item<Integer> item = new Item<>(rand.nextInt(50));

v2.add(item);

}

System.out.println("\nThe original values in v2 object are:");

v2.display();

v2.setSortStrategy(new InsertionSorter<>());

v2.performSort();

System.out.println("\nThe values in MyVector object v2 after performing InsertionSorter are:");

v2.display();

}

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// BubbleSorter class implementing the Sorter interface

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

@Override

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

// Bubble sort algorithm

int n = items.size();

boolean swapped;

do {

swapped = false;

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

if (items.get(i - 1).getItem().compareTo(items.get(i).getItem()) > 0) {

// Swap items

Item<E> temp = items.get(i - 1);

items.set(i - 1, items.get(i));

items.set(i, temp);

swapped = true;

}

}

} while (swapped);

}

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// InsertionSorter class implementing the Sorter interface

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

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

// Insertion sort algorithm

int n = items.size();

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

}

}

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

public 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;

}

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// MyVector class for managing a collection of items with sorting strategies

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() {

sorter.sort(storageM);

}

public void display() {

for (Item<E> item : storageM) {

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

}

System.out.println();

}

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// Interface for sorter classes

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

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

}

A computer screen shot of a program

Description automatically generated

***Exercise C***

/\*

\* iterator.cpp

\* File Name: lab6Exe\_C.cpp

\* Assignment: ENSF 614 Lab 6, exercise C

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex C

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

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

@Override

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

// Selection sort algorithm

int n = items.size();

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

int minIndex = i;

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

if (items.get(j).getItem().compareTo(items.get(minIndex).getItem()) < 0) {

minIndex = j;

}

}

// Swap items

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

items.set(minIndex, items.get(i));

items.set(i, temp);

}

}

}

/\*

\* iterator.cpp

\* File Name: lab6Exe\_B.cpp

\* Assignment: ENSF 614 Lab 6, exercise B

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex B

\* Submission Date: 10 November, 2023.

\*/

import java.util.Random;

public class DemoStrategyPattern {

public static void main(String[] args) {

// Create an object of MyVector<Double> with a capacity of 50 elements

MyVector<Double> v1 = new MyVector<>(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 = new Item<>(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 the bubble sort algorithm as a strategy to sort object v1

v1.setSortStrategy(new BubbleSorter<>());

v1.setSortStrategy(new SelectionSorter<>());

// Perform algorithm bubble sort on v1

v1.performSort();

System.out.println("\nThe values in MyVector object v1 after performing BubbleSorter are:");

v1.display();

// Create a MyVector<Integer> object v2

MyVector<Integer> v2 = new MyVector<>(50);

// Populate v2 with 5 randomly generated numbers

for (int i = 4; i >= 0; i--) {

Item<Integer> item = new Item<>(rand.nextInt(50));

v2.add(item);

}

System.out.println("\nThe original values in v2 object are:");

v1.performSort();

v2.display();

v2.setSortStrategy(new InsertionSorter<>());

v2.performSort();

System.out.println("\nThe values in MyVector object v1 after performing SelectionSorter is:");

v1.display();

System.out.println("\nThe values in MyVector object v2 after performing InsertionSorter are:");

v2.display();

}

}

A screenshot of a computer program

Description automatically generated

***Exercise D***

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 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);

}

}

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// This interface defines the contract for all observers in the observer pattern.

public interface Observer {

void update(ArrayList<Double> data);

}

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 2023.

\*/

public interface Subject {

void register(Observer observer); // Register the observer

void remove(Observer observer);

void notifyObservers(); // Notify all registered Oberser.

}

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// Represent the subject in the observer patter

public class DoubleArrayListSubject implements Subject {

private ArrayList<Observer> observers = new ArrayList<>();

private ArrayList<Double> data = new ArrayList<>();

@Override

public void register(Observer observer) {

observers.add(observer);

}

@Override

public void remove(Observer observer) {

observers.remove(observer);

}

@Override

public void notifyObservers() {

for (Observer observer : observers) {

observer.update(data);

}

}

// Add or populate the array double

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();

}

}

// Displays the current data list

public void populate(double[] values) {

for (double value : values) {

data.add(value);

}

notifyObservers();

}

public void display() {

if (data.isEmpty()) {

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

} else {

for (Double value : data) {

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

}

System.out.println();

}

}

}

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// Concrete Observer displays rows

public class FiveRowsTable\_Observer implements Observer {

private ArrayList<Double> data;

// Constructor function to register this oberserver

public FiveRowsTable\_Observer(Subject subject) {

subject.register(this);

}

@Override

public void update(ArrayList<Double> data) {

this.data = data;

display();

}

// Display the data in 5 rows and columns

public void display() {

int numRows = (int) Math.ceil(data.size() / 5.0);

for (int i = 0; i < numRows; i++) {

for (int j = 0; j < 5; j++) {

int index = i \* 5 + j;

if (index < data.size()) {

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

}

}

System.out.println();

}

}

}

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// This represents a concrete observer data in a three-column format.

public class ThreeColumnTable\_Observer implements Observer {

private ArrayList<Double> data;

public ThreeColumnTable\_Observer(Subject subject) {

subject.register(this);

}

@Override

public void update(ArrayList<Double> data) {

this.data = data;

display();

}

// Display method to show data in three columns.

public void display() {

int numRows = (int) Math.ceil(data.size() / 3.0);

for (int i = 0; i < numRows; i++) {

for (int j = 0; j < 3; j++) {

int index = i \* 3 + j;

if (index < data.size()) {

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

}

}

System.out.println();

}

}

}

/\*

\* OberserverPatternController.cpp

\* File Name: lab6Exe\_D.cpp

\* Assignment: ENSF 614 Lab 6, exercise D

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex D

\* Submission Date: 10 November, 2023.

\*/

import java.util.ArrayList;

// This concrete observer displays single row data

public class OneRow\_Observer implements Observer {

private ArrayList<Double> data;

public OneRow\_Observer(Subject subject) {

subject.register(this);

}

// Update the data to be displayed

@Override

public void update(ArrayList<Double> data) {

this.data = data;

display();

}

// Displays the data

public void display() {

for (Double value : data) {

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

}

System.out.println();

}

}

A screenshot of a computer program

Description automatically generated

***Exercise E***

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_E.cpp

\* Assignment: ENSF 614 Lab 6, exercise E

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex E

\* Submission Date: 10 November, 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 lets decorate t with BorderDecorator: x = 30, y = 30, width = 100, and height 100

t = new BorderDecorator(t, 30, 30, 100, 100);

// Now lets 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);

}

}

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_E.cpp

\* Assignment: ENSF 614 Lab 6, exercise E

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex E

\* Submission Date: 10 November, 2023.

\*/

import java.awt.Color;

import java.awt.Graphics;

/\*\*

\* The ColouredFrameDecorator class is responsible for adding a colored frame around a component.

\*/

public class ColouredFrameDecorator implements Component {

private Component component;

private int x;

private int y;

private int width;

private int height;

private int thickness;

public ColouredFrameDecorator(Component component, int x, int y, int width, int height, int thickness) {

this.component = component;

this.x = x;

this.y = y;

this.width = width;

this.height = height;

this.thickness = thickness;

}

@Override

public void draw(Graphics g) {

component.draw(g);

g.setColor(Color.RED);

for (int i = 0; i < thickness; i++) {

g.drawRect(x + i, y + i, width - 2 \* i, height - 2 \* i);

}

}

}

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_E.cpp

\* Assignment: ENSF 614 Lab 6, exercise E

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex E

\* Submission Date: 10 November, 2023.

\*/

import java.awt.Graphics;

/\*\*

\* Draws the object using the provided Graphics object.

\*/

public interface Component {

void draw(Graphics g);

}

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_E.cpp

\* Assignment: ENSF 614 Lab 6, exercise E

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex E

\* Submission Date: 10 November, 2023.

\*/

import java.awt.Graphics;

// Represent component to be drawned

public class Text implements Component {

private String text;

private int x;

private int y;

public Text(String text, int x, int y) {

this.text = text; // The diplayed Text

this.x = x;

this.y = y;

}

// Draws the text on the Graphics

@Override

public void draw(Graphics g) {

g.drawString(text, x, y);

}

}

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_E.cpp

\* Assignment: ENSF 614 Lab 6, exercise E

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex E

\* Submission Date: 10 November, 2023.

\*/

import java.awt.Graphics;

// The BorderDecorator class is responsible for decorating a component.

public class BorderDecorator implements Component {

private Component component;

private int x;

private int y;

private int width;

private int height;

public BorderDecorator(Component component, int x, int y, int width, int height) {

this.component = component;

this.x = x;

this.y = y;

this.width = width;

this.height = height;

}

/\*\*

\* Draws the component with the added border on the provided Graphics object.

\*

\* @param g The Graphics object to draw the component with a border.

\*/

@Override

public void draw(Graphics g) {

component.draw(g);

}

}

A screenshot of a computer

Description automatically generated

***Exercise F***

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_E.cpp

\* Assignment: ENSF 614 Lab 6, exercise E

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex E

\* Submission Date: 10 November, 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 lets decorate t with BorderDecorator: x = 30, y = 30, width = 100, and height 100

// t = new BorderDecorator(t, 30, 30, 100, 100);

// // Now lets 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);

// }

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_F.cpp

\* Assignment: ENSF 614 Lab 6, exercise F

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex F

\* Submission Date: 10 November, 2023.

\*/

public void paintComponent(Graphics g) {

int fontSize = 10;

g.setFont(new Font("TimesRoman", Font.PLAIN, fontSize));

// Decorate 't' with a ColouredGlassDecorator, ColouredFrameDecorator, and BorderDecorator.

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);

}

}

/\*

\* DemoDecoratorPatter.java

\* File Name: lab6Exe\_F.cpp

\* Assignment: ENSF 614 Lab 6, exercise F

\* Created by Mahmood Moussavi

\* Completed by: Emmanuel Alafonye

\* ENSF 614 - Fall 2023 - Lab 6, Ex F

\* Submission Date: 10 November, 2023.

\*/

import java.awt.Graphics;

import java.awt.Color;

import java.awt.Graphics2D;

public class ColouredGlassDecorator implements Component {

private Component component;

private int x;

private int y;

private int width;

private int height;

public ColouredGlassDecorator(Component component, int x, int y, int width, int height) {

this.component = component;

this.x = x;

this.y = y;

this.width = width;

this.height = height;

}

@Override

public void draw(Graphics g) {

component.draw(g);

// Add green glass cover with transparency

Graphics2D g2d = (Graphics2D) g;

g2d.setColor(new Color(0, 255, 0, 128)); // Transparent green color

g2d.fillRect(x, y, width, height);

}

}

A screenshot of a computer

Description automatically generated