Министерство транспорта Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»

РУТ (МИИТ)

Институт управления и цифровых технологий

Кафедра «Цифровые технологии управления транспортными процессами»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2

**ТЕМА: «Декоратор»**

**по дисциплине «Корпоративные информационные системы»**

**Вариант №8**

Выполнил студент группы УВА-411 Лёвин

Принял преподаватель кафедры ЦТУТП П. О. Козьяков

Москва 2022

**Текст задания на лабораторную работу**

**Вариант 8.** Создать интерфейс Операция над матрицей. Создать класс, выполняющий сложение матриц. Создать декоратор, выполняющий умножение матрицы на число и декоратор, выполняющий вычитание матриц. Продемонстрировать операции с различным набором декораторов.

**Текст программы на Java**

**Класс Lab2Start**

package ru.miit.decorator;  
  
import org.jboss.weld.environment.se.Weld;  
import org.jboss.weld.environment.se.WeldContainer;  
import java.util.Random;  
  
  
public class Lab2Start {  
 protected static Weld *weld*;  
 protected static WeldContainer *container*;  
  
 public static void main(String args[]) throws Exception {  
 *weld* = new Weld();  
 *container* = *weld*.initialize();  
  
 try {  
 Random random = new Random();  
 int size1 = random.nextInt(5)+2;//колличество столбцов матрицы  
 int size2 = random.nextInt(5)+2;//колличество строк матрицы  
  
 int i,j;  
 System.*out*.println("Матрица A:");  
 int[][] A = new int[size1][size2];  
 for (i = 0; i < size1; i++) {  
 for (j = 0; j < size2; j++) {  
 A[i][j]=random.nextInt(30)+1;//заполнение массива А  
 System.*out*.print(A[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 System.*out*.println("Матрица B:");  
 int[][] B = new int[size1][size2];  
 for (i = 0; i < size1; i++) {  
 for (j = 0; j < size2; j++) {  
 B[i][j]=random.nextInt(30)+1;//заполнение массива В  
 System.*out*.print(B[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
  
 OperationBean searcherBean = *container*.instance().select(OperationBean.class).get();  
 searcherBean.stringSearch(A, B);  
  
 } finally {  
 *weld*.shutdown();  
 }  
 }  
}

**Интерфейс IMatrixOperation**

package ru.miit.decorator;  
  
public interface IMatrixOperation {  
 int[][] searcher(int[][] A, int[][] B);  
}

**Управляющий класс OperationBean**

package ru.miit.decorator;  
  
import javax.inject.Inject;  
  
public class OperationBean {  
  
 @Inject  
 private IMatrixOperation IMatrix;  
  
 public void stringSearch(int[][] A, int[][] B) {  
 IMatrix.searcher(A, B);  
 }  
}

**Класса выполняющий сложение матриц MatrixAddition**

package ru.miit.decorator;  
  
import javax.enterprise.inject.Default;  
  
@Default  
public class MatrixAddition implements IMatrixOperation {  
 @Override  
 public int[][] searcher(int[][] A, int[][] B) {  
 int i,j,k;  
 k=0;  
 int[][] C = new int[A.length][A[0].length];  
  
 for (i = 0; i < A.length; i++) {  
 for (j = 0; j < A[0].length; j++) {  
 C[i][j] = A[i][j]+B[i][j];  
 if (C[i][j]<0){  
 k=1;  
 }  
 }  
 }  
 if (k==0) {  
 System.*out*.println("Сумма двух матриц:");  
 for (i = 0; i < A.length; i++) {  
 for (j = 0; j < A[0].length; j++) {  
 System.*out*.print(C[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 }  
 return C;  
 }  
}

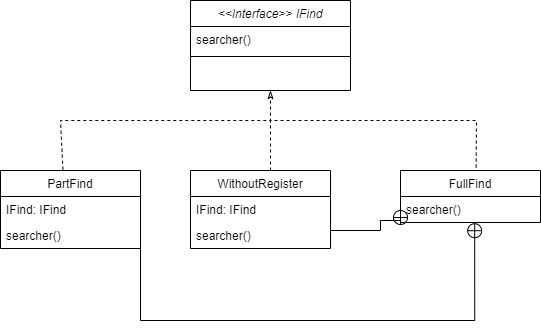
**Декоратор, выполняющий умножение матрицы на число MatrixMultiplication**

package ru.miit.decorator;  
  
import java.util.Random;  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
  
@Decorator  
public class MatrixMultiplication implements IMatrixOperation {  
 @Inject  
 @Delegate  
 private IMatrixOperation IMatrix;  
  
 @Override  
 public int[][] searcher(int[][] A, int[][] B) {  
 Random random = new Random();  
 int h = random.nextInt(9)+1;  
 int i,j;  
 int[][] C;  
 C=IMatrix.searcher(A, B);  
 System.*out*.println("Число на котрое умножется матрица = "+h);  
 for (i = 0; i < A.length; i++) {  
 for (j = 0; j < A[0].length; j++) {  
 C[i][j]=h\*C[i][j];  
 }  
 }  
 System.*out*.println("Умножение результат матрицы на число "+h+":");  
 for (i = 0; i < A.length; i++) {  
 for (j = 0; j < A[0].length; j++) {  
 System.*out*.print(C[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 return C;  
 }  
}

**Декоратор, выполняющий вычитание матриц MatrixSubtraction**

package ru.miit.decorator;  
  
import javax.decorator.Decorator;  
import javax.decorator.Delegate;  
import javax.inject.Inject;  
  
@Decorator  
public class MatrixSubtraction implements IMatrixOperation {  
 @Inject  
 @Delegate  
 private IMatrixOperation IMatrix;  
  
 @Override  
 public int[][] searcher(int[][] A, int[][] B) {  
 int i,j;  
 int[][] C;  
 C = IMatrix.searcher(A, B);  
 for (i = 0; i < A.length; i++) {  
 for (j = 0; j < A[0].length; j++) {  
 B[i][j]=-B[i][j];  
 }  
 }  
 C = IMatrix.searcher(A, B);  
 System.*out*.println("Вычитани матрицы B из A:");  
 for (i = 0; i < A.length; i++) {  
 for (j = 0; j < A[0].length; j++) {  
 System.*out*.print(C[i][j] + " ");  
 }  
 System.*out*.println();  
 }  
 return C;  
 }  
}

**Диаграмма классов**



На диаграмме изображена связь интерфейса, базового класса и декораторов.

**Заключение**

В ходе лабораторной работы был изучен механизм использования Декораторов. Декоратор — это альтернатива созданию подклассов с целью расширения функциональности. Идея состоит в том, чтобы взять класс и обернуть вокруг него другой класс (то есть декорировать его). В контексте данной лабораторной работы декораторы используются для расширения функциональности класса MatrixAddition