303626-2-11E AID:258164 | 23/08/2020

**Depreciation:**

ADT specification: Data abstraction is effective tool in design of complex system. And it provide key factors is to maintain and adaption of software system. In application , it declare the data type and operations without implementation details but possibly with semantics of operations. These provide application information needed to use ADT in program. These include data structures are accessible to user although details are hide. Declaration of function and procedures accessible to use which bodies are not provided. It support all ADT operations.

**The following explanation of the code of square matrix problem:**

1. Create squareMatrix.java code which show the creation of interface.
2. In this interface, declare an integer variable with row and column size 50. And also create function such as Make Empty (n), Store Values (I. j. value), Add, Subtract, Copy()
3. Then create ImplInterface.java which declare an class which implements interface. And showing the matrix array function.
4. Then create annotations to ensure that function overridden and these can calculate the function related data and through stored them.
5. Create MatrixTesting.java these is main program of the problem these can show the output of result data.
6. The output can be show when run these program before running first compile like “javac MatrixTesting.java” and then run “java MatrixTesting.java”.

**The following program code for depreciation:**

SquareMatrix.java

public interface SquareMatrix //Create the interface

{

int rows = 50; // Declare an integer variable

int cols = 50; // Set the values of rows and column to 50/

void MakeEmpty(int n); // Declare a method MakeEmpty

void StoreValue(int i, int j, int value); // Declare a method StoreValue

SquareMatrix add(SquareMatrix x, SquareMatrix y); // Declare a method add.

SquareMatrix subtract(SquareMatrix x, SquareMatrix y); // Declare a method subtract.

void copy(SquareMatrix s, SquareMatrix d); // Declare a method copy

}

ImplInterface.java

public class ImplInterface implements SquareMatrix

//Declare an class which implements interface.

{

int[][] mat = new int[rows][cols]; // Declare a 2d array

public void displayMatrix(int n) // Definition of the function to display matrix

{

for (int i = 0; i < n; i++) // Start the for loop for rows

{

for (int j = 0; j < n; j++) // Start the for loop for column

{

System.out.print(mat[i][j] + " ") ; // Display matrix

}

System.out.println();

} }

// annotations to ensure that function is overridden

@Override

public void MakeEmpty(int n) // Definition of the MakeEmpty

{

if (n > 50) // Check whether matrix size is greater than 50.

{

System.out.println( "Array size should be less than equal to 50 "); // Display statement

return;

}

for (int i = 0; i < n; i++) // Start the for loop to make matrix empty

{ for (int j = 0; j < n; j++) // Start the for loop

{

this.mat[i][j] = 0; // Initializing the element of matrix to zero

}

}

}

@Override

public void StoreValue(int i, int j, int value) // Definition of the function

{ this.mat[i][j] = value; // Store the value of matrix on specified location.

}

@Override

public SquareMatrix add(SquareMatrix x, SquareMatrix y) // Definition of the functionadd

{

ImplInterface impl = new ImplInterface();

// Create an object of the class.

for (int i = 0; i < 50; i++) // Start the for loop

{

for (int j = 0; j < 50; j++) // Start the for loop for column.

{

// Adding the values of 2 matrix

impl.StoreValue(i, j , ((ImplInterface) x).mat[i][j] + ((ImplInterface) y).mat[i][j]);

} }

// return reference id.

return impl;

}

@Override

// Definition of the function add

public SquareMatrix subtract(SquareMatrix x,

SquareMatrix y)

{

// Definition of the function add

ImplInterface impl = new ImplInterface();

// Start the for loop for rows

for (int i = 0; i < 50; i++)

{

// Start the for loop for columns

for (int j = 0; j < 50; j++)

{

// Subtract 2 matrixes

impl.StoreValue(i, j,

((ImplInterface) x).mat[i][j]

- ((ImplInterface) y).mat[i][j]);

}

}

// return reference id.

return impl;

}

@Override

public void copy(SquareMatrix s, SquareMatrix d) // Definition of the function copy

{

ImplInterface impl = (ImplInterface) d; //Create an object of an interface.

// Start the for loop

for (int i = 0; i < 50; i++)

{

for (int j = 0; j < 50; j++)

{

// Copying the elements in the other matrix

impl.StoreValue(i, j,

((ImplInterface) s).mat[i][j]);

} } } }

MatrixTesting.java

//Import necessary package

import java.util.Scanner;

//Declare a class

public class MatrixTesting

{

// Start the main method.

public static void main(String[] args)

{

// Create the object of scanner class.

Scanner sc = new Scanner(System.in);

// prompt the user to enter the size of matrix.

System.out

.println("Enter size of square matrix: ");

int a = sc.nextInt();

// Create the object of interface

ImplInterface imp1 = new ImplInterface();

ImplInterface imp2 = new ImplInterface();

// Start the for loop for rows

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

{

// Start the for loop for column

for (int j = 0; j < a; j++)

{

// Store the values in the matrix

imp2.StoreValue(i, j, 10 + (i \* j));

}

}

// Start the for loop for rows

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

{

// Start the for loop for column

for (int j = 0; j < a; j++)

{

// Store the values in the matrix

imp1.StoreValue(i, j, (int) (5 \* i \* j));

}

}

System.out.println("MakeEmpty method used");

// Call to the function.

imp1.MakeEmpty(a);

imp1.displayMatrix(a);

System.out.println("---------------------");

System.out.println("Display the first matrix:");

// Call to the function

imp1.displayMatrix(a);

System.out.println("---------------------");

System.out.println("Display the second matrix:");

// Call to the function.

imp2.displayMatrix(a);

System.out.println("---------------------");

System.out.println("Store value method used");

// Call to the function

imp1.StoreValue(0, 1, 999);

imp1.displayMatrix(a);

System.out.println("---------------------");

// Call to function add

// Add 2 matrixes

ImplInterface impl = (ImplInterface) imp1.add(imp1,

imp2);

System.out.println("Result of the add method");

// Call to the function

impl.displayMatrix(a);

System.out.println("---------------------");

// Subtract two matrixes

impl = (ImplInterface) imp1.subtract(imp1, imp2);

System.out

.println("Result of the subtract method:");

impl.displayMatrix(a);

System.out.println("---------------------");

// Create object of class.

ImplInterface impl3 = new ImplInterface();

// Call to the function

impl3.copy(imp2, impl3);

System.out.println("Result of the copy method:");

impl3.displayMatrix(a);

}

}

Output:

The output of code contain Make Empty (n), Store Values (I. j. value), Add, Subtract, Copy function using the squareMatrix function with row and column size 50.

