

SP SITHUNGU

200000000

PRACTICAL 5 DESIGN

## Matrix2D

```
-_rows: int
-_cols: int
-_data: int**
+DEFAULT_ROWS: const int = 2
+DEFAULT_COLS: const int = 2
+DEFAULT_VALUE: const int = 0
+MIN_DIMENSION_SIZE: const int = 2
+MAX_DIMENSION_SIZE: const int = 100000

+Matrix2D()
+Matrix2D(intRows:int,intCols:int,intDefaultValue:int)
+Matrix2D(objOriginal:const Matrix2D&)
+~Matrix2D()
+toString(): string
+getRows(): int
+getCols(): int
+getValueAt(intRow:int,intCol:int): int
+setValueAt(intRow:int,intCol:int,intValue:int): void
```

```

1  #ifndef MATRIX2D_H_INCLUDED
2  #define MATRIX2D_H_INCLUDED
3
4  #include <string>
5
6  enum ERROR_CODE{
7      SUCCESS,
8      ERROR_ARGS,
9      ERROR_RANGE
10 };
11
12 class Matrix2D{
13 public:
14     Matrix2D();
15     Matrix2D(int intRows, int intCols, int intDefaultValue);
16     Matrix2D(const Matrix2D& objOriginal);
17
18     std::string toString() const;
19
20     int getRows() const;
21     int getCols() const;
22     int getValueAt(int intRow, int intCol) const;
23
24     void setValueAt(int intRow, int intCol, int intValue);
25
26     static const int DEFAULT_ROWS = 2;
27     static const int DEFAULT_COLS = 2;
28     static const int DEFAULT_VALUE = 0;
29     static const int MIN_DIMENSION_SIZE = 2;
30     static const int MAX_DIMENSION_SIZE = 100000;
31
32     ~Matrix2D();
33 private:
34     void alloc(int intRows, int intCols, int intDefaultValue);
35     void dealloc();
36     void clone(const Matrix2D& objOriginal);
37     void enforceRange(int intArg, int intMin, int intMax) const;
38     int** _data;
39     int _rows;
40     int _cols;
41 };
42
43 #endif // MATRIX2D_H_INCLUDED

```

```

1  #include "Matrix2D.h"
2
3  #include <cassert>
4  #include <iostream>
5  #include <sstream>
6  #include <string>
7
8  using namespace std;
9
10 Matrix2D::Matrix2D() : Matrix2D(DEFAULT_ROWS, DEFAULT_COLS, DEFAULT_VALUE){}
11
12 Matrix2D::Matrix2D(int intRows, int intCols, int intDefaultValue){
13     alloc(intRows, intCols, intDefaultValue);
14 }
15
16 Matrix2D::Matrix2D(const Matrix2D& objOriginal) : Matrix2D(objOriginal._rows,
objOriginal._cols, DEFAULT_VALUE){
17     clone(objOriginal);
18 }
19
20 string Matrix2D::toString() const{
21     stringstream ssReturn;
22     for(int r = 0; r < _rows; r++){
23         for(int c = 0; c < _cols; c++){
24             ssReturn << _data[r][c] << ' ';
25         }
26         ssReturn << endl;
27     }
28     return ssReturn.str();
29 }
30
31 int Matrix2D::getRows() const{
32     return _rows;
33 }
34
35 int Matrix2D::getCols() const{
36     return _cols;
37 }
38
39 int Matrix2D::getValueAt(int intRow, int intCol) const{
40     enforceRange(intRow, 0, _rows - 1);
41     enforceRange(intCol, 0, _cols - 1);
42     return _data[intRow][intCol];
43 }
44
45 void Matrix2D::setValueAt(int intRow, int intCol, int intValue){
46     enforceRange(intRow, 0, _rows - 1);
47     enforceRange(intCol, 0, _cols - 1);
48     _data[intRow][intCol] = intValue;
49 }
50
51 void Matrix2D::alloc(int intRows, int intCols, int intDefaultValue){
52     enforceRange(intRows, MIN_DIMENSION_SIZE, MAX_DIMENSION_SIZE);
53     enforceRange(intCols, MIN_DIMENSION_SIZE, MAX_DIMENSION_SIZE);
54     _rows = intRows;
55     _cols = intCols;
56     _data = new int*[_rows];
57     for(int r = 0; r < _rows; r++){
58         _data[r] = new int[_cols];
59         for(int c = 0; c < _cols; c++){
60             _data[r][c] = intDefaultValue;
61         }
62     }
63 }
64
65 void Matrix2D::dealloc(){

```

```

66     assert(_data != nullptr);
67     for(int r = 0; r < _rows; r++){
68         delete [] _data[r];
69     }
70     delete [] _data;
71 }
72
73 void Matrix2D::clone(const Matrix2D& objOriginal){
74     for(int r = 0; r < _rows; r++){
75         for(int c = 0; c < _cols; c++){
76             _data[r][c] = objOriginal._data[r][c];
77         }
78     }
79 }
80
81 void Matrix2D::enforceRange(int intArg, int intMin, int intMax) const{
82     if(intArg < intMin || intArg >intMax){
83         cerr << intArg << " must be in [" << intMin
84             << ", " << intMax << "]" << endl;
85         exit(ERROR_RANGE);
86     }
87 }
88
89 Matrix2D::~Matrix2D(){
90     dealloc();
91 }

```

```

1  #include "Matrix2D.h"
2
3  #include <iostream>
4  #include <string>
5  #include <cstdlib>
6
7  using namespace std;
8
9  bool matrixEquals(const Matrix2D& objLHS, const Matrix2D& objRHS);
10 void populateMatrix(Matrix2D& objMatrix);
11 void displayMatrix(const Matrix2D& objMatrix);
12
13 int main()
14 {
15     Matrix2D objMatrix(5, 5, 1);
16     populateMatrix(objMatrix);
17     Matrix2D objClone(objMatrix);
18     if(matrixEquals(objClone, objMatrix))
19         displayMatrix(objClone);
20     return SUCCESS;
21 }
22
23 bool matrixEquals(const Matrix2D& objLHS, const Matrix2D& objRHS){
24     if(objLHS.getRows() != objRHS.getRows())
25         return false;
26     if(objLHS.getCols() != objRHS.getCols())
27         return false;
28
29     for(int r = 0; r < objLHS.getRows(); r++){
30         for(int c = 0; c < objLHS.getCols(); c++){
31             if(objLHS.getValueAt(r, c) != objRHS.getValueAt(r, c))
32                 return false;
33         }
34     }
35     return true;
36 }
37
38 void populateMatrix(Matrix2D& objMatrix){
39     for(int r = 0; r < objMatrix.getRows(); r++){
40         for(int c = 0; c < objMatrix.getCols(); c++){
41             objMatrix.setValueAt(r, c, rand() % 10);
42         }
43     }
44 }
45
46 void displayMatrix(const Matrix2D& objMatrix){
47     cout << objMatrix.toString();
48 }

```