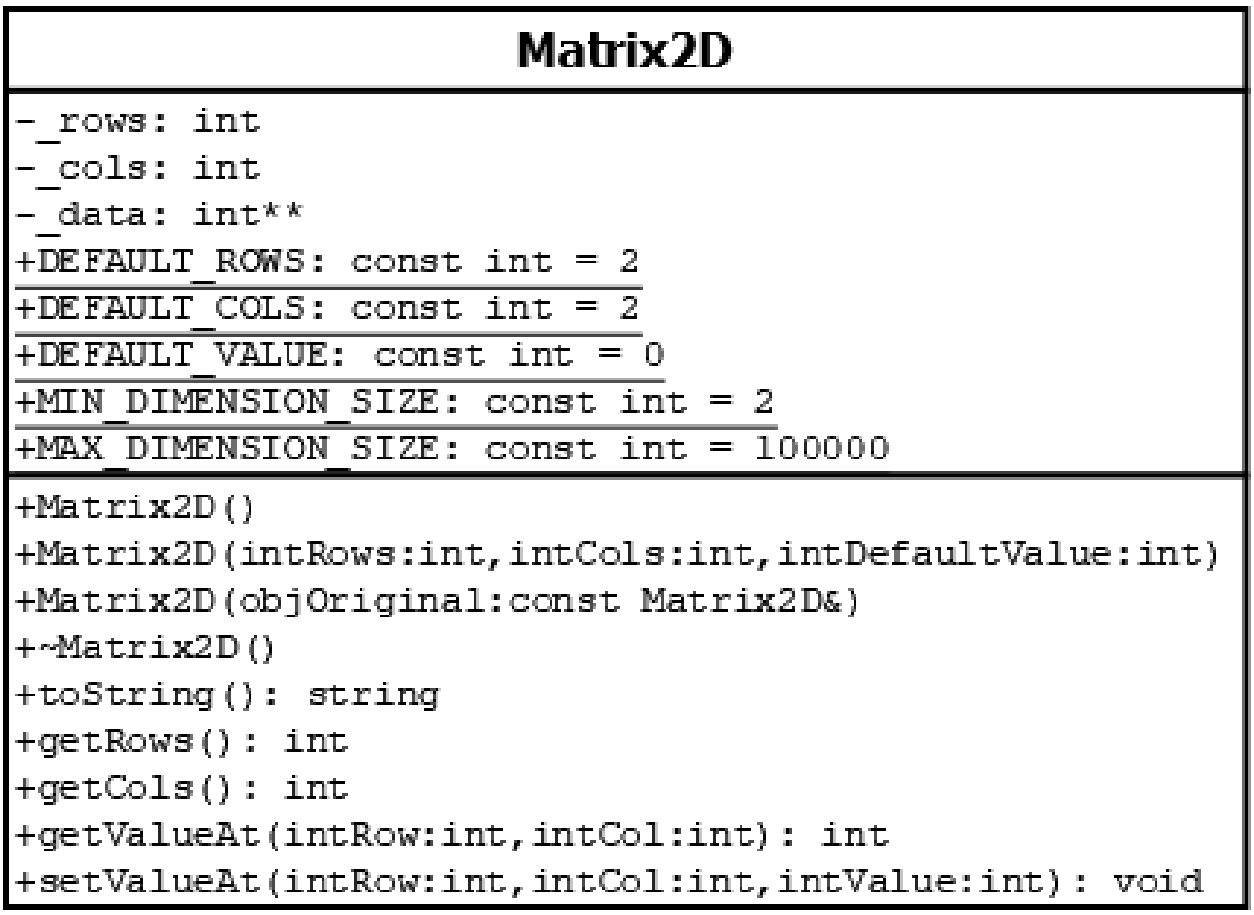
SP SITHUNGU

200000000

PRACTICAL 5 DESIGN



```cpp
#ifndef MATRIX2D_H_INCLUDED
#define MATRIX2D_H_INCLUDED

#include <string>

enum ERROR_CODE{
    SUCCESS,
    ERROR_ARGS,
    ERROR_RANGE
};

class Matrix2D{
public:
    Matrix2D();
    Matrix2D(int intRows, int intCols, int intDefaultValue);
    Matrix2D(const Matrix2D& objOriginal);

    std::string toString() const;

    int getRows() const;
    int getCols() const;
    int getValueAt(int intRow, int intCol) const;

    void setValueAt(int intRow, int intCol, int intValue);

    static const int DEFAULT_ROWS = 2;
    static const int DEFAULT_COLS = 2;
    static const int DEFAULT_VALUE = 0;
    static const int MIN_DIMENSION_SIZE = 2;
    static const int MAX_DIMENSION_SIZE = 100000;

    ~Matrix2D();
private:
    void alloc(int intRows, int intCols, int intDefaultValue);
    void dealloc();
    void clone(const Matrix2D& objOriginal);
    void enforceRange(int intArg, int intMin, int intMax) const;
    int** _data;
    int _rows;
    int _cols;
};

#endif // MATRIX2D_H_INCLUDED
```

```cpp
#include "Matrix2D.h"

#include <cassert>
#include <iostream>
#include <sstream>
#include <string>

using namespace std;

Matrix2D::Matrix2D() : Matrix2D(DEFAULT_ROWS, DEFAULT_COLS, DEFAULT_VALUE){}

Matrix2D::Matrix2D(int intRows, int intCols, int intDefaultValue){
    alloc(intRows, intCols, intDefaultValue);
}

Matrix2D::Matrix2D(const Matrix2D& objOriginal) : Matrix2D(objOriginal._rows,
objOriginal._cols, DEFAULT_VALUE){
    clone(objOriginal);
}

string Matrix2D::toString() const{
    stringstream ssReturn;
    for(int r = 0; r < _rows; r++){
        for(int c = 0; c < _cols; c++){
            ssReturn << _data[r][c] << ' ';
        }
        ssReturn << endl;
    }
    return ssReturn.str();
}

int Matrix2D::getRows() const{
    return _rows;
}

int Matrix2D::getCols() const{
    return _cols;
}

int Matrix2D::getValueAt(int intRow, int intCol) const{
    enforceRange(intRow, 0, _rows - 1);
    enforceRange(intCol, 0, _cols - 1);
    return _data[intRow][intCol];
}

void Matrix2D::setValueAt(int intRow, int intCol, int intValue){
    enforceRange(intRow, 0, _rows - 1);
    enforceRange(intCol, 0, _cols - 1);
    _data[intRow][intCol] = intValue;
}

void Matrix2D::alloc(int intRows, int intCols, int intDefaultValue){
    enforceRange(intRows, MIN_DIMENSION_SIZE, MAX_DIMENSION_SIZE);
    enforceRange(intCols, MIN_DIMENSION_SIZE, MAX_DIMENSION_SIZE);
    _rows = intRows;
    _cols = intCols;
    _data = new int*[_rows];
    for(int r = 0; r < _rows; r++){
        _data[r] = new int[_cols];
        for(int c = 0; c < _cols; c++){
            _data[r][c] = intDefaultValue;
        }
    }
}

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  }
```

```cpp
#include "Matrix2D.h"

#include <iostream>
#include <string>
#include <cstdlib>

using namespace std;

bool matrixEquals(const Matrix2D& objLHS, const Matrix2D& objRHS);
void populateMatrix(Matrix2D& objMatrix);
void displayMatrix(const Matrix2D& objMatrix);

int main()
{
    Matrix2D objMatrix(5, 5, 1);
    populateMatrix(objMatrix);
    Matrix2D objClone(objMatrix);
    if(matrixEquals(objClone, objMatrix))
        displayMatrix(objClone);
    return SUCCESS;
}

bool matrixEquals(const Matrix2D& objLHS, const Matrix2D& objRHS){
    if(objLHS.getRows() != objRHS.getRows())
        return false;
    if(objLHS.getCols() != objRHS.getCols())
        return false;

    for(int r = 0; r < objLHS.getRows(); r++){
        for(int c = 0; c < objLHS.getCols(); c++){
            if(objLHS.getValueAt(r, c) != objRHS.getValueAt(r, c))
                return false;
        }
    }
    return true;
}

void populateMatrix(Matrix2D& objMatrix){
    for(int r = 0; r < objMatrix.getRows(); r++){
        for(int c = 0; c < objMatrix.getCols(); c++){
            objMatrix.setValueAt(r, c, rand() % 10);
        }
    }
}

void displayMatrix(const Matrix2D& objMatrix){
    cout << objMatrix.toString();
}
```