Image2D # pixels: int** # rows: int # cols: int # bgColour: int +DEFAULT ROWS: const int = 500 +DEFAULT COLS: const int = 500 +DEFAULT MAX INTENSITY: const int = 255 +DEFAULT MIN INTENSITY: const int = 0 +MIN DIMENSION SIZE: const int = 2 +MAX DIMENSION SIZE: const int = 100000 +Image2D() +Image2D(intRows:int,intCols:int) +Image2D(objOriginal:const Image2D&) +~Image2D() +toString(): string +getRows(): int +getCols(): int +getPixel(): int +setPixel(intRow:int,intCol:int): void +alloc(intRows:int,intCols:int): void +dealloc(): void +clone(objOriginal:const Image2D&): void

PGMImage

+PGMImage(intRows:int,intCols:int) +toString(): string

PBMImage

+PBMImage(intRows:int,intCols:int) +toString(): string

```
1 #ifndef COMMONLIB_H
2 #define COMMONLIB_H
3
4 #include <string>
5
6 namespace CommonSpace{
7 enum ERROR_CODE{
8
          SUCCESS,
9
          ERROR_ARGS,
10
           ERROR_RANGE
11
      };
12
13
       int CInt(std::string arg);
       void enforceRange(int intArg, int intMin, int intMax);
14
15
       int rangedRandom(int intLBound, int intUBound);
16 }
17
18 #endif // COMMONLIB_H
```

```
1 #include "CommonLib.h"
3 #include <iostream>
4 #include <sstream>
5 using namespace std;
6
7 namespace CommonSpace{
8
       int CInt(string arg){
9
          stringstream ssConv(arg);
10
           int intResult = 0;
11
           ssConv >> intResult;
12
           return intResult;
13
14
       void enforceRange(int intArg, int intMin, int intMax){
15
           if(intArg < intMin | intArg >intMax){
16
17
               cerr << intArg << " must be in [" << intMin</pre>
                    << ", " << intMax << "]" << endl;
18
19
               exit(ERROR_RANGE);
           }
20
21
22
        int rangedRandom(int intLBound, int intUBound){
23
           return rand() % (intUBound - intLBound + 1) + intLBound;
24
25
26 }
27
```

```
1 #ifndef IMAGE2D_H
 2
   #define IMAGE2D_H
 3
 4 #include <string>
 5
 6 class Image2D{
 7 public:
 8
        Image2D();
 9
        Image2D(int intRows, int intCols);
10
        Image2D(const Image2D& objOriginal);
11
        virtual std::string toString() const = 0;
12
13
14
        // Accessor methods.
15
        int getRows() const;
16
        int getCols() const;
        int getPixel(int intRow, int intCol) const;
17
18
        // Mutator method.
        void setPixel(int intRow, int intCol, int intPixel);
19
20
21
        static const int DEFAULT ROWS = 500;
22
        static const int DEFAULT COLS = 500;
23
        static const int DEFAULT MAX INTENSITY = 255;
        static const int DEFAULT_MIN_INTENSITY = 0;
25
        static const int MIN_DIMENSION_SIZE = 2;
26
        static const int MAX_DIMENSION_SIZE = 1000000;
27
28
       ~Image2D();
29 protected:
30
        void alloc(int intRows, int intCols);
31
        void dealloc();
32
        void clone(const Image2D& objOriginal);
33
        int _bgColour;
34
        int _rows;
35
        int _cols;
        int** _pixels;
36
37
   };
38
39
   #endif // IMAGE2D_H
```

```
1
   #include "CommonLib.h"
 2
   #include "Image2D.h"
 3
 4
   #include <cassert>
 5
   #include <iostream>
 6
7
   using namespace std;
8
   using namespace CommonSpace;
9
10
   Image2D::Image2D() : Image2D(DEFAULT_ROWS, DEFAULT_COLS){}
11
12
   Image2D::Image2D(int intRows, int intCols){
13
        alloc(intRows, intCols);
14
15
16
   Image2D::Image2D(const Image2D& objOriginal): Image2D(objOriginal._rows, objOriginal
._cols){
17
        clone(objOriginal);
18
19
20
    int Image2D::getRows() const{
21
        return _rows;
22
23
24
    int Image2D::getCols() const{
25
        return _cols;
26
27
28
    int Image2D::getPixel(int intRow, int intCol) const{
29
        enforceRange(intRow, 0, _rows - 1);
30
        enforceRange(intCol, 0, _cols - 1);
31
        return _pixels[intRow][intCol];
32
33
34
   void Image2D::setPixel(int intRow, int intCol, int intPixel){
35
        enforceRange(intRow, 0, _rows - 1);
        enforceRange(intCol, 0, _cols - 1);
36
        _pixels[intRow][intCol] = intPixel;
37
38
39
40
   void Image2D::alloc(int intRows, int intCols){
41
        enforceRange(intRows, MIN_DIMENSION_SIZE, MAX_DIMENSION_SIZE);
42
        enforceRange(intCols, MIN_DIMENSION_SIZE, MAX_DIMENSION_SIZE);
        _rows = intRows;
43
        _cols = intCols;
44
        _pixels = new int*[_rows];
45
        for(int r = 0; r < _rows; r++) {</pre>
46
47
            _pixels[r] = new int[_cols];
48
            for(int c = 0; c < _cols; c++){</pre>
49
                _pixels[r][c] = rangedRandom(DEFAULT_MIN_INTENSITY, DEFAULT_MAX_INTENSITY
);
50
51
52
53
54
   void Image2D::dealloc(){
55
        assert(_pixels != nullptr);
56
        for(int r = 0; r < _rows; r++){</pre>
57
            delete [] _pixels[r];
58
59
        delete [] _pixels;
60
61
   void Image2D::clone(const Image2D& objOriginal){
62
63
        for(int r = 0; r < _rows; r++){</pre>
64
            for(int c = 0; c < _cols; c++){</pre>
```

```
1 #ifndef PGMIMAGE_H
2 #define PGMIMAGE_H
3
4 #include "Image2D.h"
5
6 class PGMImage : public Image2D{
7 public:
8    PGMImage(int intRows, int intCols);
9    virtual std::string toString() const;
10 };
11
12 #endif // PGMIMAGE_H
```

```
1 #include "PGMImage.h"
3 #include <iostream>
4 #include <sstream>
5 using namespace std;
6
7 PGMImage::PGMImage(int intRows, int intCols)
8
    : Image2D(intRows, intCols){}
9
10 string PGMImage::toString() const{
stringstream ssPPM;
12
       ssPPM << "P2" << end1
            << _cols << ' ' << _rows << endl
13
             << 255 << endl;
14
15
      for(int r = 0; r < _rows; r++){</pre>
         for(int c = 0; c < _cols; c++){</pre>
16
              ssPPM << _pixels[r][c] << ' ';
17
18
19
           ssPPM << endl;
20
       return ssPPM.str();
22 }
```

```
1 #ifndef PBMIMAGE_H
2 #define PBMIMAGE_H
3
4 #include "Image2D.h"
5
6 class PBMImage : public Image2D{
7 public:
8    PBMImage(int intRows, int intCols);
9    virtual std::string toString() const;
10 };
11
12 #endif // PBMIMAGE_H
```

```
1 #include "PBMImage.h"
 2
 3 #include <sstream>
 4 using namespace std;
 5
 6 PBMImage::PBMImage(int intRows, int intCols){
 7
 8
9
10 string PBMImage::toString() const{
11
   stringstream ssPPM;
12
       ssPPM << "P1" << endl
13
            << _cols << ' ' << _rows << endl;
        for(int r = 0; r < _rows; r++){</pre>
14
15
           for(int c = 0; c < _cols; c++){</pre>
               _pixels[r][c] > 0 ? ssPPM << 1 : ssPPM << _pixels[r][c];
16
17
18
19
           ssPPM << endl;
20
21
       return ssPPM.str();
22 }
```

```
#include "CommonLib.h"
 1
   #include "PBMImage.h"
 3
   #include "PGMImage.h"
 4
 5
   #include <iostream>
 6
 7
   using namespace std;
 8
   using namespace CommonSpace;
 9
10 void printImage(Image2D& objImage);
11
12 int main(int argc, char** argv)
13
14
        if(argc != 4){
15
            cerr << "Usage: " << argv[0]</pre>
                 << " NUM_ROWS NUM_COLS IMAGE_TYPE"
16
17
                 << endl;
18
            exit(ERROR_ARGS);
19
20
        int intRows = CInt(argv[1]);
21
        int intCols = CInt(argv[2]);
22
        int intImageType = CInt(argv[3]);
23
        enforceRange(intImageType, 0, 1);
24
25
        if(intImageType == 0)
26
            printImage(*(new PGMImage(intRows, intCols)));
27
            printImage(*(new PBMImage(intRows, intCols)));
28
29
30
        return SUCCESS;
31
32
33
   void printImage(Image2D& objImage){
34
        cout << objImage.toString();</pre>
35
```