```cpp
#pragma once
/*! Current class version*/
#define IMAGEPROCESSING_VERSION 1

/*! MACRO which sets the original pointer to the original image or a clone of the earlier processed image */
#define CHAIN_PROCESS(chain, O, type)   if (chain) { TempImg = ProcessedImg.clone(); O = (type *)TempImg.data; } else { O = (type *)   OriginalImg.data; }
/*! MACRO which trows an EmtpyImageException if the matrix is empty*/
#define EMPTY_CHECK(img) if (img.empty()) { throw Exception::EmtpyImageException(); }

#include <stdint.h>
#include <opencv2/core.hpp>
#include <cmath>
#include <vector>

#include "EmptyImageException.h"
#include "WrongKernelSizeException.h"
#include "ChannelMismatchException.h"
#include "PixelValueOutOfBoundException.h"

using namespace cv;

namespace Vision
{
    class ImageProcessing
    {
    protected:
        uchar* GetNRow(int nData, int hKsize, int nCols, uint32_t totalRows);
        Mat TempImg;

    public:
        ImageProcessing();
        ~ImageProcessing();
        Mat OriginalImg;
        Mat ProcessedImg;

        std::vector<Mat> extractChannel(const Mat &src, uint8_t channel);

        /*! Copy a matrix to a new matrix with a LUT mask
        \param src the source image
        \param *LUT type T with a LUT to filter out unwanted pixel values
```

```
41          \param cvType an in where you can pas CV_UC8C1 etc.
42          \return The new matrix
43          */
44          template <typename T>   Mat CopyMat(const Mat &src, T *LUT, int cvType)
45          {
46              Mat dst(src.size(), cvType);
47              uint32_t i = 0;
48              uint32_t nData = dst.rows * dst.cols * dst.step[1];
49              while (i < nData)
50              {
51                  dst.data[i] = LUT[(T)src.data[i * src.step[1]]];
52                  i++;
53              }
54              return dst;
55          }
56      };
57  }
58
```