```cpp
/*! \class ImageProcessing
\brief Core class of all the image classes
Core class of all the image classes with a few commonly shared functions and variables
*/
#include "ImageProcessing.h"

namespace Vision
{
    /*! Constructor of the core class*/
    ImageProcessing::ImageProcessing()  {   }

    /*! De-constructor of the core class*/
    ImageProcessing::~ImageProcessing() {   }

    /*! Create a LUT indicating which iteration variable i is the end of an row
    \param nData an int indicating total pixels
    \param hKsize int half the size of the kernel, if any. which acts as an offset from the border pixels
    \param nCols int number of columns in a row
    \return array of uchars where a zero is a middle column and a 1 indicates an end of an row minus the offset from half the kernel size
    */
    uchar* ImageProcessing::GetNRow(int nData, int hKsize, int nCols, uint32_t totalRows)
    {
        // Create LUT to determine when there is an new row
        uchar *nRow = new uchar[nData] { };
        int i = 0;
        int shift = nCols - hKsize - 1;
        while (i <= totalRows)
        {
            nRow[(i * nCols) + shift] = 1;
            i++;
        }
        return nRow;
    }

    // Todo: Optimize
    std::vector<Mat> ImageProcessing::extractChannel(const Mat &src, uint8_t channel)
    {
        if (channel >= src.channels()) { throw Exception::ChannelMismatchException(); }

        //Mat chans[3] = { Mat(src.size(), src.type()), Mat(src.size(), src.type()), Mat(src.size(), src.type()) };

```

```
42          //uint32_t nData = src.rows * src.cols * src.step.buf[1];
43          //uint32_t stepSize = src.step.buf[1] / 3;
44          //uint32_t count = 0;
45          //for (uint32_t i = 0; i < nData; i += stepSize)
46          //{
47          //  chans[0].data[count] = src.data[i];
48          //  i += stepSize;
49          //  chans[1].data[count] = src.data[i];
50          //  i += stepSize;
51          //  chans[2].data[count] = src.data[i];
52          //  count += stepSize;
53          //}
54
55
56          vector<Mat> chans;
57          split(src, chans);
58          return chans;
59      }
60  }
```