I wrote a class named “ImageCompressionSVD”. It’s nothing but reading the argument from user from command line to create a different module for each input argument. There are 4 types of arguments you could parse in as bellow:

1. {1} {pgm text image file name}

Example: “1 image.pgm”, where “image.pgm” is a PGM image files in plain text format that you could easily open and edit in any text reader application like notepad on Window. This program will convert its format into a binary PGM image files. The output binary file name will be added “\_b” at the end. As per example above, “image.pgm” will output to “image\_b.pgm”.

1. {2} {pgm binary image file name}

Example: “2 image\_b.pgm”. This program will revert the process it had done in option one. It is going to convert this binary image file into a text image file. The output file name will be removed the “\_b” at the end. In case it is not ending with “\_b” then the prefix “\_t” will be added at the end of the file name. For example, “image\_b.pgm” will output to “image.pgm”, “image.pgm” will output to “image\_t.pgm”, and so on.

1. {3} [header text file name] [SVD text file name] [k integer value]

Example: “3 header.txt SVD.txt 5”. The program will read the content of header file to get the height (M), the width (N), and the grey scale level of the image. It will read the content of SVD text file to get the unitary matrix (UMatrix), the rectangular diagonal value (RD), and the V matrix, and then write all the value into one binary file format named it “image\_b.pgm.SVD”. All three parameter [header text file name], [SVD text file name], and [k integer value] are optional, since if they are missing then the default values will be used as: “header.txt”, “SVD.txt”, and M value will be use for “k”.

1. {4} [binary SVD file name of a PGM image]

Example: “4 image\_b.pgm.SVD”. The program will read all of these values: the height (M), the width (N), the grey scale level, the unitary matrix (UMatrix), the rectangular diagonal value (RD), and the V matrix to reconstruct the PGM text image named it “image\_k.pgm”. The parameter [binary SVD file name of a PGM image] is optional, since if it is missing then the default values will be used as: “image\_b.pgm.SVD”.

**The “textPgmToBinary” class**

This class will be use in argument option one to convert from text to binary PGM image file. Bellow rule will be applied:

1. Remove every comment lines start with “#” sign.
2. Ignore any line starting with any non integer number as "P2".
3. All of the negative number will be ignored because they are not eligible for image and gray scale.
4. The width and the height of the image will be written in 2 byte (store the value up to 32767); all other color and gray scale will be written in 1 byte only for compacting file size.

**The “BinaryPgmTotext” class**

This class will be use in argument option two to convert from binary PGM to a text image file. Bellow rule will be applied:

1. The first 2 number are the width and the height of the image will be read in 2 byte (store the value up to 32767).
2. All other color and gray scale will be read in 1 byte.

**The “Header” class and “SVD” class**

These two classes will be use in argument option 3 to convert into a binary PGM image file from header text file and a SVD text file. “Header” class will be used to read the height (M), the width (N), and the grey scale level of the image. Whereas the “SVD” class will be used to read the unitary matrix (UMatrix), the rectangular diagonal value (RD), and the V matrix. The method “saveToFile” in the “SVD” class will write 2 byte for M value, 2 byte for N, 2 byte for K value, 1 byte for grey scale level, U matrix (M x M) with 2 byte double value for every cell in U matrix, only the 2 byte double value on the diagonal line of rectangular diagonal matrix RD (M x N), V matrix (N x N) with 2 byte double value for every cell in V matrix.

The format of a 2 byte double number:

* Bit 15 is a sign bit. If this number is negative then bit 15 =1, otherwise bit 15 =0.
* Bit 14 is a sign bit for exponent. If the exponent is negative then bit 14 =1, otherwise bit 14 =0.
* Exponent will use 4 bit from bit 10 to bit 13 to store the value from up to 2^5-1 = 63.

*[-63 < exponent < +63]*

* Mantissa will use 10 bit from bit 0 to bit 9 to store the value from 100 to 999

The double range can store: -999 x 10^63 ... +999 x 10^63 with precision 100 x 10 ^ (-63)

*For example:*

*Double f =0.59497426f = 594 x 10 ^ (-3)*

*Will be store as: 0 1 0 0 1 1 1 0 0 1 0 1 0 0 1 0 (2 byte: 78 82)*

**The “binarySVDtoKPgmImage” class**

This class will be use in argument option 4 to read from a binary SVD image and then it will convert into a text PGM image file. It follow the SVD file format that it does in option 3, which is: M,N, K are in 2 byte, grey scale level in byte, and all other value of matrix are in 2 byte double.

The program then reconstructs the PGM text image named it “image\_k.pgm”.