# ITEC3150

## Assignment – Read/Write Binary PNM FIles

Create an app that can read, write and manipulate binary PNM files.

|  |  |
| --- | --- |
|  | P3 # "P3" means this is a RGB color image in ASCII 3 2 # "3 2" is the width and height of the image in pixels 255 # "255" is the maximum value for each color # The part above is the header # adapted from https://en.wikipedia.org/wiki/Netpbm # The part below is the image data: RGB triplets 255 0 0 # red  0 255 0 # green  0 0 255 # blue 255 255 0 # yellow 255 255 255 # white  0 0 0 # black |
|  | P3 # Created by GIMP version 2.1 0.32 PNM plug-in 5 4 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 0 255 0 0 255 0 0 255 255 255 255 255 255 255 0 0 255 0 0 255 0 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 |
|  | P3 # Created by GIMP version 2.1 0.32 PNM plug-in 5 4 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 255 255 255 255 255 255 255 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 |
|  | P3 # Created by GIMP version 2.1 0.32 PNM plug-in 5 4 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 0 255 255 255 255 255 255 255 0 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 |

The simplistic images above are provided with their corresponding ASCII source files. The first image is provided to illustrate the basic formatting. The fourth image can be created by overlaying the third image upon the second, with the assumption that white pixels in the third will be treated as if they are transparent when performing the combination. The fourth image shows the result of the overlaying (or stacking) operation.

Note that these examples utilize PNM’s human-readable (ASCII) P3 format. Your solution will provide capability to read, write, and manipulate PNM’s binary P6 format.

Note that the images above show excessive blurring between pixels, this appears to be an artifact of MS Word. When viewed in a photo editing software like Photoshop or GIMP, the high contrasting pixel edges will be more discernable.

You are free to refer to these resources when completing your solution:

* <https://netpbm.sourceforge.net/doc/ppm.html>
* <https://en.wikipedia.org/wiki/Netpbm>
* A demonstration of a valid solution (will be demoed in class)
* A reference application that implements all of the functionality for a text PNM file. This is provided as an IntelliJ project zip and is provided in the D2L assignment area. (see PNMStarter.zip in the D2L dropbox). Not that this project contains multiple junit tests, if you get all the tests to run successfully you will have a valid solution! (That’s providing you do not simply hardcode a hack to get the tests to pass, as that would not be acceptable).

### Your implementation must:

|  |  |
| --- | --- |
| *Points* | *Description* |
| 30 | Reads a binary PNM file |
| 30 | Writes a binary PNM file |
| 20 | Multiple images can be stacked, after a color is identified as transparent for the top layer |
| 10 | Author attribution appears in comment field on written pnm files |
| 10 | Demonstrate Read/Write/Stack sequence with student provided binary files |
| - | Follow good coding practices and standards |

### Submission Info

Your D2L Dropbox submission must include:

* An expository video. The video will highlight each rubric point and you will describe your solution by showing the associate code. The video must be shared privately with me. Submissions missing this item receive **a grade of zero**.
* *Zip* file with your entire project. An IntelliJ project is highly preferred. Missing archives or archives in non-supported formats (rar, tgz, bzip, 7z, etc) will receive ***a grade of zero***.

Your linked video assets must:

* Be accessible (check your links!)

### Hints

* Remember to use the troubleshooting techniques we've been exploring this semester: Lots of System.out.printlns(), debugger, etc.
* Build features incrementally and test often.
* Your demo video should highlight and describe each item in the rubric. Keep it short! Your video need not be more than 1-3 minutes.