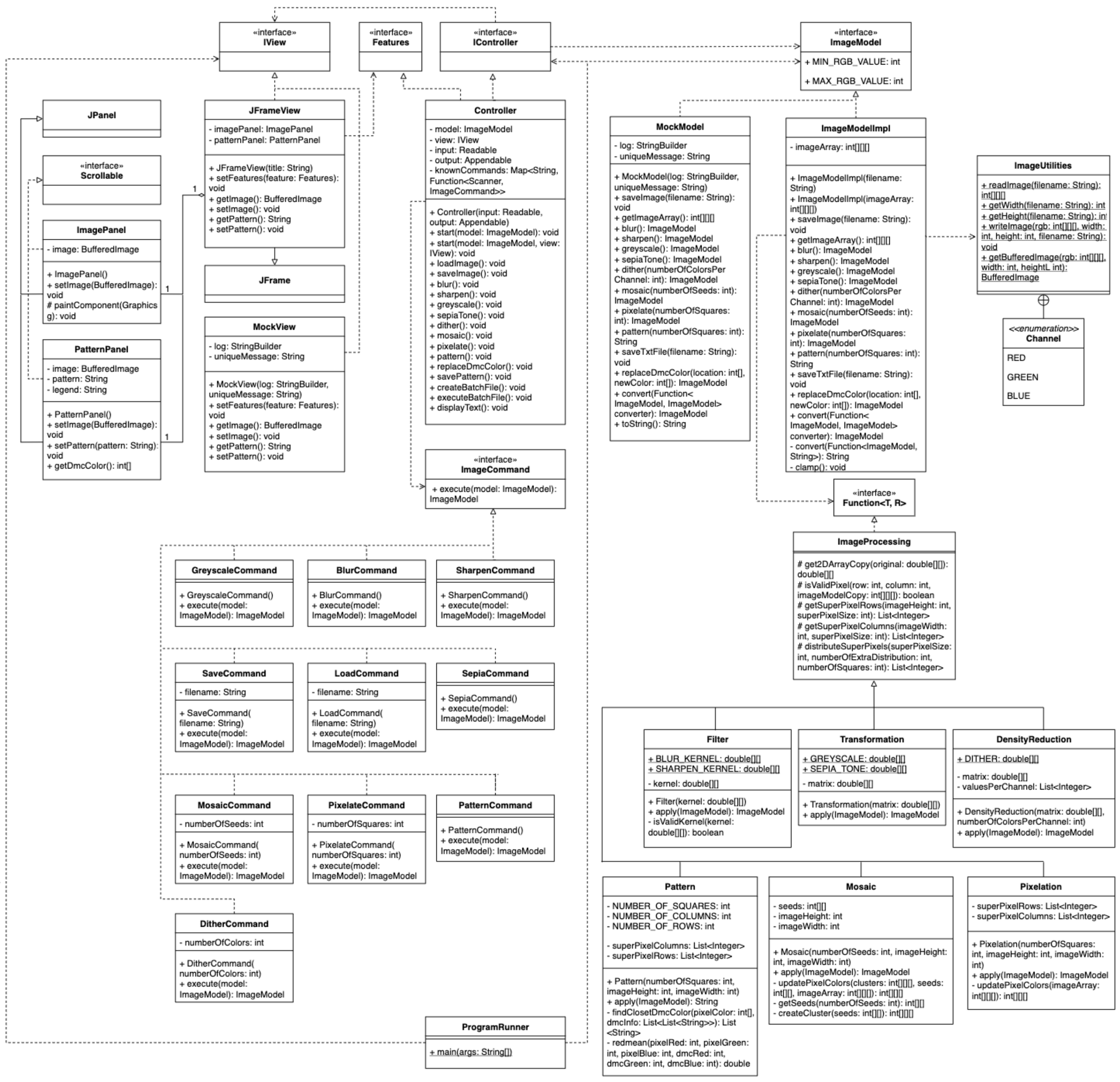
Project 5 UML Diagram


«interface» IView
«interface» Features
«interface» IController
«interface» ImageModel
+ MIN_RGB_VALUE: int
+ MAX_RGB_VALUE: int
JPanel
«interface» Scrollable
ImagePanel
- image: BufferedImage
+ ImagePanel()
+ setImage(BufferedImage): void
# paintComponent(Graphics g): void
PatternPanel
- image: BufferedImage
- pattern: String
- legend: String
+ PatternPanel()
+ setImage(BufferedImage): void
+ setPattern(pattern: String): void
+ getDmcColor(): int[]
1
1
1
JFrameView
- imagePanel: ImagePanel
- patternPanel: PatternPanel
+ JFrameView(title: String)
+ setFeatures(feature: Features): void
+ getImage(): BufferedImage
+ setImage(): void
+ getPattern(): String
+ setPattern(): void
JFrame
MockView
- log: StringBuilder
- uniqueMessage: String
+ MockView(log: StringBuilder, uniqueMessage: String)
+ setFeatures(feature: Features): void
+ getImage(): BufferedImage
+ setImage(): void
+ getPattern(): String
+ setPattern(): void
Controller
- model: ImageModel
- view: IView
- input: Readable
- output: Appendable
- knownCommands: Map<String, Function<Scanner, ImageCommand>>
+ Controller(input: Readable, output: Appendable)
+ start(model: ImageModel): void
+ start(model: ImageModel, view: IView): void
+ loadImage(): void
+ saveImage(): void
+ blur(): void
+ sharpen(): void
+ greyscale(): void
+ sepiaTone(): void
+ dither(): void
+ mosaic(): void
+ pixelate(): void
+ pattern(): void
+ replaceDmcColor(): void
+ savePattern(): void
+ createBatchFile(): void
+ executeBatchFile(): void
+ displayText(): void
«interface» ImageCommand
+ execute(model: ImageModel): ImageModel
GreyscaleCommand
+ GreyscaleCommand()
+ execute(model: ImageModel): ImageModel
BlurCommand
+ BlurCommand()
+ execute(model: ImageModel): ImageModel
SharpenCommand
+ SharpenCommand()
+ execute(model: ImageModel): ImageModel
SaveCommand
- filename: String
+ SaveCommand(filename: String)
+ execute(model: ImageModel): ImageModel
LoadCommand
- filename: String
+ LoadCommand(filename: String)
+ execute(model: ImageModel): ImageModel
SepiaCommand
+ SepiaCommand()
+ execute(model: ImageModel): ImageModel
MosaicCommand
- numberOfSeeds: int
+ MosaicCommand(numberOfSeeds: int)
+ execute(model: ImageModel): ImageModel
PixelateCommand
- numberOfSquares: int
+ PixelateCommand(numberOfSquares: int)
+ execute(model: ImageModel): ImageModel
PatternCommand
+ PatternCommand()
+ execute(model: ImageModel): ImageModel
DitherCommand
- numberOfColors: int
+ DitherCommand(numberOfColors: int)
+ execute(model: ImageModel): ImageModel
ProgramRunner
+ main(args: String[])
MockModel
- log: StringBuilder
- uniqueMessage: String
+ MockModel(log: StringBuilder, uniqueMessage: String)
+ saveImage(filename: String): void
+ getImageArray(): int[][][]
+ blur(): ImageModel
+ sharpen(): ImageModel
+ greyscale(): ImageModel
+ sepiaTone(): ImageModel
+ dither(numberOfColorsPer Channel: int): ImageModel
+ mosaic(numberOfSeeds: int): ImageModel
+ pixelate(numberOfSquares: int): ImageModel
+ pattern(numberOfSquares: int): String
+ saveTxtFile(filename: String): void
+ replaceDmcColor(location: int[], newColor: int[]): ImageModel
+ convert(Function< ImageModel, ImageModel> converter): ImageModel
+ toString(): String
ImageModelImpl
- imageArray: int[][][]
+ ImageModelImpl(filename: String)
+ ImageModelImpl(imageArray: int[][][])
+ saveImage(filename: String): void
+ getImageArray(): int[][][]
+ blur(): ImageModel
+ sharpen(): ImageModel
+ greyscale(): ImageModel
+ sepiaTone(): ImageModel
+ dither(numberOfColorsPer Channel: int): ImageModel
+ mosaic(numberOfSeeds: int): ImageModel
+ pixelate(numberOfSquares: int): ImageModel
+ pattern(numberOfSquares: int): String
+ saveTxtFile(filename: String): void
+ replaceDmcColor(location: int[], newColor: int[]): ImageModel
+ convert(Function< ImageModel, ImageModel> converter): ImageModel
- convert(Function<ImageModel, String>): String
- clamp(): void
ImageUtilities
+ readImage(filename: String): int[][][]
+ getWidth(filename: String): int
+ getHeight(filename: String): int
+ writeImage(rgb: int[][][], width: int, height: int, filename: String): void
+ getBufferedImage(rgb: int[][][], width: int, heightL int): BufferedImage
<<enumeration>> Channel
RED
GREEN
BLUE
«interface» Function<T, R>
ImageProcessing
# get2DArrayCopy(original: double[][]): double[][]
# isValidPixel(row: int, column: int, imageModelCopy: int[][][]): boolean
# getSuperPixelRows(imageHeight: int, superPixelSize: int): List<Integer>
# getSuperPixelColumns(imageWidth: int, superPixelSize: int): List<Integer>
# distributeSuperPixels(superPixelSize: int, numberOfExtraDistribution: int, numberOfSquares: int): List<Integer>
Filter
+ BLUR_KERNEL: double[][]
+ SHARPEN_KERNEL: double[][]
- kernel: double[][]
+ Filter(kernel: double[][])
+ apply(ImageModel): ImageModel
- isValidKernel(kernel: double[][]): boolean
Transformation
+ GREYSCALE: double[][]
+ SEPIA_TONE: double[][]
- matrix: double[][]
+ Transformation(matrix: double[][])
+ apply(ImageModel): ImageModel
DensityReduction
+ DITHER: double[][]
- matrix: double[][]
- valuesPerChannel: List<Integer>
+ DensityReduction(matrix: double[][], numberOfColorsPerChannel: int)
+ apply(ImageModel): ImageModel
Pattern
- NUMBER_OF_SQUARES: int
- NUMBER_OF_COLUMNS: int
- NUMBER_OF_ROWS: int
- superPixelColumns: List<Integer>
- superPixelRows: List<Integer>
+ Pattern(numberOfSquares: int, imageHeight: int, imageWidth: int)
+ apply(ImageModel): String
- findClosetDmcColor(pixelColor: int[], dmcInfo: List<List<String>>): List <String>
- redmean(pixelRed: int, pixelGreen: int, pixelBlue: int, dmcRed: int, dmcGreen: int, dmcBlue: int): double
Mosaic
- seeds: int[][]
- imageHeight: int
- imageWidth: int
+ Mosaic(numberOfSeeds: int, imageHeight: int, imageWidth: int)
+ apply(ImageModel): ImageModel
- updatePixelColors(clusters: int[][][], seeds: int[][], imageArray: int[][][]): int[][][]
- getSeeds(numberOfSeeds: int): int[][]
- createCluster(seeds: int[][]): int[][][]
Pixelation
- superPixelRows: List<Integer>
- superPixelColumns: List<Integer>
+ Pixelation(numberOfSquares: int, imageHeight: int, imageWidth: int)
+ apply(ImageModel): ImageModel
- updatePixelColors(imageArray: int[][][]): int[][][]

Project 5 Testing Plan

Change(s) made:

(1) Previously, the program would ask the user for a filename when the user executes a "generate cross stitch pattern" command. It automatically saves the pattern in the given filename. For this project, since changes can be made after a cross stitch pattern is generated, the "save pattern to filename" command would be executed only when the user explicitly asks the program to do so.

Assumption(s) made:

(1) For the "exchange one color for another in a cross-stitch pattern" operation, I am making an assumption that the user can replace the old color with any color from the DMC color options. If the user selects the old color again, nothing happens. If the user selects a new color, the operation will be executed.
(2) For the "create and execute a batch file" operation, I will provide the user with examples of commands with the right format. If the user creates a batch file with the wrong format, the batch file will not be executed successfully, and an error message would show up. If the user creates a batch file with the right format, the batch file will be executed successfully.

For this project, I plan to make a similar approach with what I did for project 4. For project 4, I created a "mock" model class, MockModel, which implements the ImageModel interface. Since I will add a view class, JFrameView, which implements the IView interface, and the controller will serve as the connection between the Model and the View, I plan to create a "mock" view class, MockView, which also implements the IView interface. In this way, I can make sure that the controller, in isolation, works correctly.

MockModel

Since I plan to add more methods in the ImageModel interface, I will update my MockModel class and test for these new methods.

Testing MockModel and Controller	Input	Expected Output
Loading an image, generating a pattern, and saving the pattern to a desired file	load goat.png pattern 50 save pattern goat-pattern.txt	Success assertEquals("load goat.png\npattern 50\nsave pattern goat- pattern.txt\n", log.toString());
Loading an image and saving a pattern to a desired file	load goat.png save pattern goat-pattern.text	Error message: pattern has to be generated first
Loading an image, generating a pattern, and replacing an old color with a new color	load goat.png pattern 50 replace DMC1 with DMC2	Success assertEquals("load goat.png\npattern 50\nreplace DMC1 with DMC2\n", log.toString());

Loading an image, generating a pattern, and replacing an old color with the same color	load goat.png pattern 50 replace DMC1 with DMC1	Success but nothing really happens for the last command assertEquals("load goat.png\npattern 50\nreplace DMC1 with DMC1\n", log.toString());

MockView

Testing MockView, MockModel, and Controller	Input	Expected Output
The setFeatures() method is called when we call Controller.start(MockModel, MockView)	Controller.start(MockModel, MockView)	Success assertEquals("Set features\n", log.toString())
Loading an image	Click "load" from the menu in GUI and select an image	Success assertEquals("Set features\nload image\n", log.toString());
Loading an image and executing a filter operation	"load" -> "blur" "load" -> "sharpen"	Success assertEquals("Set features\nload image\nset image\n", log.toString());
Loading an image and executing a color transformation operation	"load" -> "greyscale" "load" -> "sepia"	Success assertEquals("Set features\nload image\nset image\n", log.toString());
Loading an image and executing a color density reduction operation	"load" -> "dither" -> 8 "load" -> "dither" -> 16	Success assertEquals("Set features\nload image\nset image\n", log.toString());
Loading an image and executing a chunk operation	"load" -> "mosaic" -> 570 "load" -> "pixelate" -> 50	Success assertEquals("Set features\nload image\nset image\n", log.toString());
Loading an image and executing an invalid chunk operation Example: the number of seeds is greater than the number of pixels in the image	"load" -> "mosaic" -> 50000 "load" -> "mosaic" -> -50 "load" -> "pixelate" -> 50000 "load" -> "pixelate" -> 0	Error message: invalid mosaic command or invalid pixelation command
Loading an image and executing a pattern operation	"load" -> "pattern" -> 50	Success

		assertEquals("Set features\nload image\nset pattern\n", log.toString());
Loading an image and executing an invalid pattern operation Example: the number of squares entered is greater than the width of the image	"load" -> "pattern" -> 5000 "load" -> "pattern" -> -3	Error message: invalid pattern command
Loading an image, executing a pattern operation, and replacing a DMC color	"load" -> "pattern" -> 50 -> "replace color" -> 1 -> 2	Success assertEquals("Set features\nload image\nset pattern\nset pattern\n", log.toString());
Loading an image, executing a pattern operation, and removing a DMC color	"load" -> "pattern" -> 50 -> "remove color" -> 1	Success assertEquals("Set features\nload image\nset pattern\nset pattern\n", log.toString());