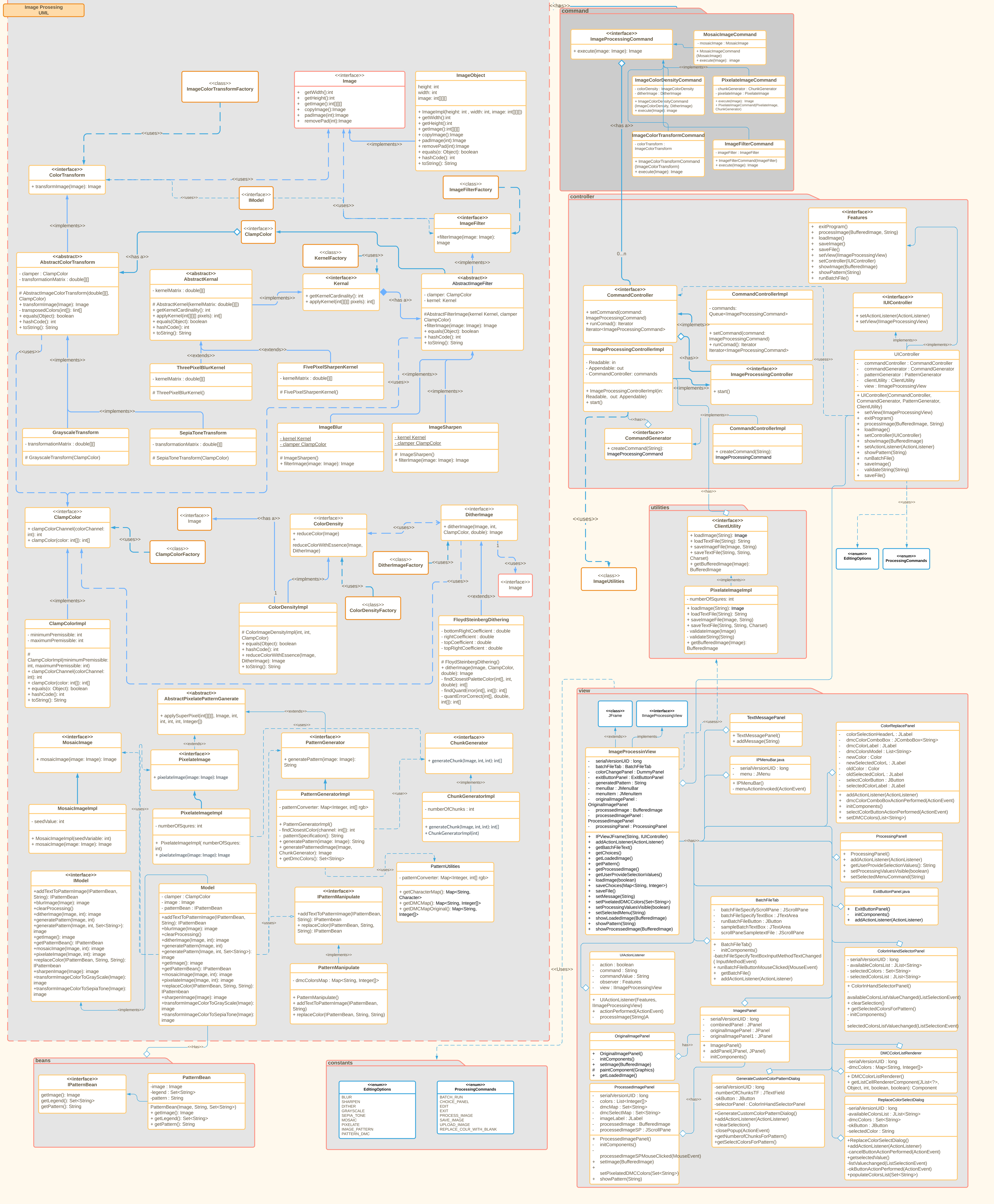
Image Prosesing UML
<<class>> ImageColorTransformFactory
<<interface>> Image
+ getWidth():int
+ getHeight():int
+ getImage():int[][][]
+ copyImage():Image
+ padImage(int):Image
+ removePad(int):Image
ImageObject
height: int
width: int
image: int[][][]
+ ImageImpl(height: int , width: int, image: int[][][])
+ getWidth():int
+ getHeight():int
+ getImage():int[][][]
+ copyImage():Image
+ padImage(int):Image
+ removePad(int):Image
+ equals(o: Object): boolean
+ hashCode(): int
+ toString(): String
<<implements>>
<<uses>>
<<interface>> ColorTransform
+ transformImage(Image): Image
<<interface>> IModel
<<class>> ImageFilterFactory
<<interface>> ImageFilter
+filterImage(image: Image): Image
<<interface>> ClampColor
<<abstract>> AbstractColorTransform
- clamper : ClampColor
- transformationMatrix : double[][]
# AbstractImageColorTransform(double[][], ClampColor)
+ transformImage(Image): Image
- transposedColors(int[]): int[]
+ equals(Object): boolean
+ hashCode(): int
+ toString(): String
<<has a>>
<<abstract>> AbstractKernal
- kernelMatrix : double[][]
# AbstractKernel(kernelMatrix: double[][])
+ getKernelCardinality(): int
+ applyKernel(int[][][] pixels): int[]
+ equals(Object): boolean
+ hashCode(): int
+ toString(): String
<<class>> KernalFactory
<<interface>> Kernal
+ getKernelCardinality(): int
+ applyKernel(int[][][] pixels): int[]
<<abstract>> AbstractImageFilter
- clamper: ClampColor
- kernel: Kernel
#AbstractFilterImage(kernel Kernel, clamper ClampColor)
+filterImage(image: Image): Image
+ equals(Object): boolean
+ hashCode(): int
+ toString(): String
<<extends>>
ThreePixelBlurKernel
- kernelMatrix : double[][]
# ThreePixelBlurKernel()
FivePixelSharpenKernel
- kernelMatrix : double[][]
# FivePixelSharpenKernel()
GrayscaleTransform
- transformationMatrix : double[][]
# GrayscaleTransform(ClampColor)
SepiaToneTransform
- transformationMatrix : double[][]
# SepiaToneTransform(ClampColor)
ImageBlur
- kernel Kernel
- clamper ClampColor
# ImageSharpen()
+ filterImage(image: Image): Image
ImageSharpen
- kernel Kernel
- clamper ClampColor
# ImageSharpen()
+ filterImage(image: Image): Image
<<interface>> ClampColor
+ clampColorChannel(colorChannel: int): int
+ clampColor(color: int[]): int[]
<<class>> ClampColorFactory
<<interface>> Image
<<interface>> ColorDensity
+ reduceColor(Image)
+ reduceColorWithEssence(Image, DitherImage)
<<interface>> DitherImage
+ ditherImage(Image, int, ClampColor, double): Image
<<class>> DitherImageFactory
<<interface>> Image
<<implments>>
ClampColorImpl
- minimumPremissible: int
- maximumPremissible: int
# ClampColorImpl(minimumPremissible: int, maximumPremissible: int)
+ clampColorChannel(colorChannel: int): int
+ clampColor(color: int[]): int[]
+ equals(Object): boolean
+ hashCode(): int
+ toString(): String
ColorDensityImpl
# ColorImageDensityImpl(int, int, ClampColor)
+ equals(Object): boolean
+ hashCode(): int
+ reduceColorWithEssence(Image, DitherImage): Image
+ toString(): String
<<class>> ColorDensityFactory
FloydSteinbergDithering
- bottomRightCoefficient : double
- rightCoefficient : double
- topCoefficient : double
- topRightCoefficient : double
# FloydSteinbergDithering()
+ ditherImage(Image, ClampColor, double): Image
- findClosestPaletteColor(int[], int, double): int[]
- findQuantError(int[], int[]): int[]
- quantErrorCorrect(int[], double, int[]): int[]
<<abstract>> AbstractPixelatePatternGenerate
+ applySuperPixel(int[][][], Image, int, int, Integer[])
<<interface>> MosaicImage
+ mosaicImage(image: Image): Image
<<interface>> PixelateImage
+ pixelateImage(image: Image): Image
<<interface>> PatternGenerator
+ generatePattern(image: Image): String
<<interface>> ChunkGenerator
+ generateChunk(Image, int, int): int[]
MosaicImageImpl
- seedValue: int
+ MosaicImageImpl(seedVariable: int)
+ mosaicImage(image: Image): Image
PixelateImageImpl
- numberOfSqures: int
+ PixelateImageImpl(numberOfSqures: int)
+ pixelateImage(image: Image): Image
PatternGeneratorImpl
- patternConverter: Map<Integer, int[] rgb>
+ PatternGeneratorImpl()
- findClosestColor(channel: int[]): int
- patternSpecification(): String
+ generatePattern(image: Image): String
+ generatePatternedImage(Image, String): Image
+ getDmcColors(): Set<String>
ChunkGeneratorImpl
- numberOfChunks : int
+ generateChunk(Image, int, int): int[]
+ ChunkGeneratorImpl(int)
PatternUtilities
- patternConverter: Map<Integer, int[] rgb>
+ getCharacterMap(): Map<String, Character>
+ getDMCMap(): Map<String, Integer[]>
+ getDMCMapOriginal(): Map<String, Integer[]>
<<interface>> IModel
+addTextToPatternImage(IPatternBean, String): IPatternBean
+blurImage(Image): image
+clearProcessing()
+ditherImage(Image, int): image
+generatePattern(Image, int)
+generatePattern(Image, int, Set<String>): image
+getImage(): image
+getPatternBean(): IPatternBean
+mosaicImage(Image, int): image
+pixelateImage(Image, int): image
+replaceColor(IPatternBean, String, String): IPatternBean
+sharpenImage(Image): image
+transformImageColorToGrayScale(Image): image
+transformImageColorToSepiaTone(Image): image
Model
- clamper : ClampColor
- image : Image
- patternBean : IPatternBean
+addTextToPatternImage(IPatternBean, String): IPatternBean
+blurImage(Image): image
+clearProcessing()
+ditherImage(Image, int): image
+generatePattern(Image, int)
+generatePattern(Image, int, Set<String>): image
+getImage(): image
+getPatternBean(): IPatternBean
+mosaicImage(Image, int): image
+pixelateImage(Image, int): image
+replaceColor(IPatternBean, String, String): IPatternBean
+sharpenImage(Image): image
+transformImageColorToGrayScale(Image): image
+transformImageColorToSepiaTone(Image): image
<<interface>> IPatternManipulate
+addTextToPatternImage(IPatternBean, String): IPatternBean
+ replaceColor(IPatternBean, String, String): IPatternBean
PatternManipulate
- dmcColorsMap : Map<String, Integer[]>
+ PatternManipulate()
+ addTextToPatternImage(IPatternBean, String)
+ replaceColor(IPatternBean, String, String)
<<Has>>
beans
<<interface>> IPatternBean
getImage(): Image
getLegend(): Set<String>
getPattern(): String
PatternBean
-image : Image
-legend : Set<String>
-pattern : String
PatternBean(Image, String, Set<String>)
+ getImage(): Image
+ getLegend(): Set<String>
+ getPattern(): String
constants
<<enum>> EditingOptions
BLUR
SHARPEN
DITHER
GRAYSCALE
SEPIA_TONE
MOSAIC
PIXELATE
IMAGE_PATTERN
PATTERN_DMC
<<enum>> ProcessingCommands
BATCH_RUN
CHOICE_PANEL
EDIT
EXIT
PROCESS_IMAGE
SAVE_IMAGE
UPLOAD_IMAGE
REPLACE_COLR_WITH_BLANK
<<has>>
command
<<interface>> ImageProcessingCommand
+ execute(image: Image): Image
MosaicImageCommand
- mosaicImage : MosaicImage
+ MosaicImageCommand (MosaicImage)
+ execute(image): image
ImageColorDensityCommand
- colorDensity : ImageColorDensity
- ditherImage : DitherImage
+ ImageColorDensityCommand (ImageColorDensity, DitherImage)
+ execute(image): image
PixelateImageCommand
- chunkGenerator : ChunkGenerator
- pixelateImage : PixelateImage
+ execute(image): image
+ PixelateImageCommand(PixelateImage, ChunkGenerator)
ImageColorTransformCommand
- colorTransform : ImageColorTransform
+ ImageColorTransformCommand (ImageColorTransform)
+ execute(Image): Image
ImageFilterCommand
- imageFilter : ImageFilter
+ ImageFilterCommand(ImageFilter)
+ execute(Image): Image
<<has a>>
0...n
controller
<<interface>> Features
+ exitProgram()
+ processImage(BufferedImage, String)
+ loadImage()
+ saveImage()
+ saveFile()
+ setView(IImageProcessingView)
+ setController(IUIController)
+ showImage(BufferedImage)
+ showPattern(String)
+ runBatchFile()
<<interface>> CommandController
+ setCommand(command: ImageProcessingCommand)
+ runComad(): Iterator
Iterator<ImageProcessingCommand>
CommandControllerImpl
- commands: Queue<ImageProcessingCommand>
+ setCommand(command: ImageProcessingCommand)
+ runComad(): Iterator
Iterator<ImageProcessingCommand>
<<interface>> IUIController
+ setActionListener(ActionListener)
+ setView(IImageProcessingView)
ImageProcessingControllerImpl
- Readable: in
- Appendable: out
- CommandController: commands
+ ImageProcessingControllerImpl(in: Readable, out: Appendable)
+ start()
<<interface>> ImageProcessingController
+ start()
UIController
- commandController : CommandController
- commandGenerator : CommandGenerator
- patternGenerator : PatternGenerator
- clientUtility : ClientUtility
- view : IImageProcessingView
+ UIController(CommandController, CommandGenerator, PatternGenerator, ClientUtility)
+ setView(IImageProcessingView)
+ exitProgram()
+ processImage(BufferedImage, String)
+ loadImage()
+ setController(IUIController)
+ showImage(BufferedImage)
+ setActionListener(ActionListener)
+ showPattern(String)
+ runBatchFile()
+ saveImage()
- validateString(String)
+ saveFile()
<<interface>> CommandGenerator
+ createCommand(String): ImageProcessingCommand
CommandControllerImpl
+ createCommand(String): ImageProcessingCommand
<<uses>>
<<class>> ImageUtilities
utilities
<<interface>> ClientUtility
+ loadImage(String): Image
+ loadTextFile(String): String
+ saveImageFile(Image, String)
+ saveTextFile(String, String, Charset)
+ getBufferedImage(Image): BufferedImage
PixelateImageImpl
- numberOfSqures: int
+ loadImage(String): Image
+ loadTextFile(String): String
+ saveImageFile(Image, String)
+ saveTextFile(String, String, Charset)
- validateImage(Image)
- validateString(String)
+ getBufferedImage(Image): BufferedImage
<<enum>> EditingOptions
<<enum>> ProcessingCommands
view
<<class>> JFrame
<<interface>> IImageProcessingView
TextMessagePanel
+ TextMessagePanel()
+ addMessage(String)
ImageProcessinView
- serialVersionUID : long
- batchFileTab : BatchFileTab
- colorChangePanel : DummyPanel
- exitButtonPanel : ExitButtonPanel
- generatedPattern : String
- menuBar : JMenuBar
- menuItem : JMenuItem
- originalImagePanel : OriginalImagePanel
- processedImage : BufferedImage
- processedImagePanel : ProcessedImagePanel
- processingPanel : ProcessingPanel
+ IPViewJFrame(String, IUIController)
+ addActionListener(ActionListener)
+ getBatchFileText()
+ getChoices()
+ getLoadedImage()
+ getPattern()
+ getProcessedImage()
+ getUserProvideSelectionValues()
+ loadImage(boolean)
+ saveChoices(Map<String, Integer>)
+ saveFile()
+ setMessage(String)
+ setPixelatedDMCColors(Set<String>)
+ setProcessingValuesVisible(boolean)
+ setSelectedMenu(String)
+ showLoadedImage(BufferedImage)
+ showPattern(String)
+ showProcessedImage(BufferedImage)
IPMenuBar.java
- serialVersionUID : long
- menu : JMenu
+ IPMenuBar()
- menuActionInvoked(ActionEvent)
ColorReplacePanel
- colorSelectionHeaderL : JLabel
- dmcColorBox : JComboBox<String>
- dmcColorLabel : JLabel
- dmcColorsModel : List<String>
- newColor : Color
- newSelectedColorL : JLabel
- oldColor : Color
- oldSelectedColorL : JLabel
- selectColorButton : JButton
- selectedColorLabel : JLabel
+ addActionListener(ActionListener)
+ dmcColorComboBoxActionPerformed(ActionEvent)
+ initComponents()
+ selectColorButtonActionPerformed(ActionEvent)
+ setDMCColors(List<String>)
ProcessingPanell
+ ProcessingPanel()
+ addActionListener(ActionListener)
+ getUserProvideSelectionValues(): String
+ setProcessingValuesVisible(boolean)
+ setSelectedMenuCommand(String)
ExitButtonPanel.java
+ ExitButtonPanel()
+ initComponents()
+ addActionListener(ActionListener)
BatchFileTab
- batchFileSpecifyScrollPane : JScrollPane
- batchFileSpecifyTextBox : JTextArea
- runBatchFileButton : JButton
- sampleBatchTextBox : JTextArea
- scrollPaneSampletextFile : JScrollPane
+ BatchFileTab()
+ initComponents()
-batchFileSpecifyTextBoxInputMethodTextChanged (InputMethodEvent)
+ runBatchFileButtonMouseClicked(MouseEvent)
+ getBatchFile()
+ addActionListener(ActionListener)
UIActionListener
- action : boolean
- command : String
- commandValue : String
- observer : Features
- view : IImageProcessingView
+ UIActionListener(Features, IImageProcessingView)
+ actionPerformed(ActionEvent)
- processImage(String)A
ColorInHandSelectorPanel
- serialVersionUID : long
- availableColorsList : JList<String>
- selectedColors : Set<String>
- selectedColorsList : JList<String>
+ ColorInHandSelectorPanel()
+ availableColorsListValueChanged(ListSelectionEvent)
+ clearSelection()
+ getSelectedColorsForPattern()
- initComponents()
- selectedColorsListValuechanged(ListSelectionEvent)
ImagesPanel
- serialVersionUID : long
- combinedPanel : JPanel
- originalImagePanel : JPanel
- originalImagePanel1 : JPanel
+ ImagesPanel()
+ addPanel(JPanel, JPanel)
- initComponents()
OriginalImagePanel
+ OriginalImagePanel()
- initComponents()
+ setImage(BufferedImage)
# paintComponent(Graphics)
+ getLoadedImage()
DMCColorListRenderer
-serialVersionUID : long
-dmcColors : Map<String, Integer[]>
+ DMCColorListRenderer()
+ getListCellRendererComponent(JList<?>, Object, int, boolean, boolean): Component
ProcessedImagePanel
- serialVersionUID : long
- colors : List<Integer[]>
- dmcMap : Set<String>
- dmcSelectMap : Set<String>
- imageLabel : JLabel
- processedImage : BufferedImage
- processedImageSP : JScrollPane
+ ProcessedImagePanel()
- initComponents()
- processedImageSPMouseClicked(MouseEvent)
+ setImage(BufferedImage)
+ setPixelatedDMCColors(Set<String>)
+ showPattern(String)
GenerateCustomColorPatternDialog
-serialVersionUID : long
-numberOfChunksTF : JTextField
-okButton : JButton
-selectorPanel : ColorInHandSelectorPanel
+GenerateCustomColorPatternDialog()
+addActionListener(ActionListener)
+clearSelection()
-closePopup(ActionEvent)
+getNumberofChunksForPattern()
+getSelectedColorsForPattern()
ReplaceColorSelectDialog
-serialVersionUID : long
-availableColorsList : JList<String>
-dmcColors : Set<String>
-okButton : JButton
-selectedColor : String
+ReplaceColorSelectDialog()
+addActionListener(ActionListener)
-cancelButtonActionPerformed(ActionEvent)
+getselectedValue()
-listValuechanged(ListSelectionEvent)
-okButtonActionPerformed(ActionEvent)
+populateColorsList(Set<String>)
<<extends>>
implements
<<uses>>
<<has>>
has>>
<<Uses>>

Testing Plan UI Controller(Project 5):

IUIController Testing		
Test Case	**Values passed**	**Expected Result**
generatePatternWithCustomColors()	view.setUploadedImagePath("user/image.jpg"); view.setProcessedImage(new BufferedImage(20, 20, BufferedImage.TYPE_INT_RGB)); view.setNumberOfChunksForPattern("40"); view.setPatternBean(new IPatternBeanMock()); model.setPatternBean(new IPatternBeanMock()); Set<String> colors = new LinkedHashSet<>(); colors.add("1"); colors.add("2"); view.setSelectedColorForPattern(colors); feature.generatePatternWithCustomColors(); StringBuilder expectedValue = new StringBuilder();	append("IModelMock.getImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getSelectColorsForPattern() Invoked\n IImageProcessingViewMock.getNumberOfChunksForPattern() Invoked\n IModelMock.generatePattern() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n IImageProcessingViewMock.showPattern() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n
generatePatternWithCustomColorsChunkNull()	view.setUploadedImagePath("user/image.jpg"); view.setProcessedImage(new BufferedImage(20, 20, BufferedImage.TYPE_INT_RGB)); Set<String> colors = new LinkedHashSet<>(); colors.add("1"); colors.add("2"); view.setSelectedColorForPattern(colors); feature.generatePatternWithCustomColors(); StringBuilder expectedValue = new StringBuilder();	IModelMock.getImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getSelectColorsForPattern() Invoked\n IImageProcessingViewMock.getNumberOfChunksForPattern() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n Number of chunks for pattern generation must be a number greater than zero
patternWithCustomColorSelected()	feature.patternWithCustomColorSelected();	IImageProcessingViewMock.popupCustomColorsDialog() Invoked\n
processImageBlurTest()	feature.editOptionSelection(EditingOption	IImageProcessingViewMock.resetPattern() Invoked\n

	s.BLUR.toString());
	
	 view.setUploadedImagePath("user/image.jpg");
	 feature.processImage();
		IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IModelMock.blurImage() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n;

processImageDitherTest()	feature.editOptionSelection(EditingOptions.DITHER.toString());
	
	 view.setUploadedImagePath("user/image.jpg");
	
	 view.setUserProvideSelectionValues("2");
	 feature.processImage();
		IImageProcessingViewMock.resetPattern() Invoked\n IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLo

		adedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IModelMock.ditherImage() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n

processImageDitherWrongCommandTest()	feature.editOptionSelection(EditingOptions.DITHER.toString());
	
	view.setUploadedImagePath("user/image.jpg");
	feature.processImage();
		IImageProcessingViewMock.resetPattern() Invoked\n IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n CommandGeneratorImpl: Dither image must be in format \"dither <noOfColors>\";

processImageEmptyCommand()	feature.editOptionSelection(""); feature.processImage();	IImageProcessingViewMock.setMessage() Invoked\n Error: unknown command > IImageProcessingViewMock.setMessage() Invoked\n You must select editing option before processing the image.

processImageGrayscaleTest()	feature.editOptionSelection(EditingOptions.GRAYSCALE.toString());
	
	 view.setUploadedImagePath("user/image.jpg"); feature.processImage();
		IImageProcessingViewMock.resetPattern() Invoked\n IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IModelMock.transformImageColorToGrayScale() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n

processImageMosaicEmptyUserValueTest()	feature.editOptionSelection(EditingOptions.MOSAIC.toString());
	
	 view.setUserProvideSelectionValues("50");
	
	 view.setUploadedImagePath("user/image.
		IImageProcessingViewMock.resetPattern() Invoked\n IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPattern

	jpg");
	    feature.processImage();
		Bean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IModelMock.mosaicImage() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n

processImagePatternBlankUserValueTest()	feature.editOptionSelection(EditingOptions.GENERATE_PATTERN.toString());
	    view.setUserProvideSelectionValues(" ");
	
	view.setUploadedImagePath("user/image.jpg");
	    feature.processImage();
		IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n

		Pixelation/ Mosaic/ Dither/ Pattern user provided value " + "must be specified and must be a non zero number. Provided: ;
processImagePatternEmptyUservalueTest()	feature.editOptionSelection(EditingOptions.GENERATE_PATTERN.toString()); view.setUserProvideSelectionValues(""); view.setUploadedImagePath("user/image.jpg"); feature.processImage();	IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n Pixelation/ Mosaic/ Dither/ Pattern user provided value" + " must be specified and must be a non zero number. Provided:
testLoadImage()	feature.loadImage(); StringBuilder expectedValue = new StringB	IImageProcessingViewMock.loadImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showLoadedImage() Invoked\n
testLoadImageException()	clientUtility = new ClientUtilityImpl(); UIController uiController = new UIController(commandController, commandGenerator, clientUtility, model); uiController.setView(view);	IImageProcessingViewMock.loadImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n

	feature = uiController;
	
	feature.loadImage();
		ClientUtilityImpl: String argument cannot be null and empty

testNullArguments()	new UIController(commandController, null, clientUtility, model);
	new UIController(commandController, commandGenerator, clientUtility, null);
	new UIController(commandController, commandGenerator, clientUtility, model);
	new UIController(null, commandGenerator, clientUtility, model);
		UIController: Arguments must not be null

testRunBatchFile()	view.setBatchFile("load salad.jpg\n" + "mosaic 1650\n" + "save salad-mosaic-1650.jpg");	IImageProcessingViewMock.getBatchFileText() Invoked\n Invoked MockClientUtility.loadImage\n IModelMock.mosaicImage() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n Image with name salad.jpg loaded successfully.\n Image processing command \"mosaic 1650\" called successfully.\n

testSaveFileLoadedImage()	view.setUploadedImagePath("user/image.jpg");
	feature.saveFile();
		IModelMock.getPatternBean() Invoked\n
		IModelMock.getImage() Invoked\n
		IImageProcessingViewMock.setMessage() Invoked\n.append(
		"You must load and process the image before proceeding to save the image/ pattern.

testSaveFileNullLoadedImage()	feature.saveFile();	IModelMock.getPatternBean() Invoked\n IModelMock.getImage() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n You must load and process the image before proceeding to save the image/ " + "pattern.;

testSaveFileNullSavePath()	feature.loadImage();
	
	view.setUploadedImagePath("user/image.jpg");
		IImageProcessingViewMock.loadImage() Invoked\n
		IImageProcessingViewMock.getLoadedImagePath() Invoked\n

	model.setImage(
	
	ImageFactory.builImage(ImageType.STANDARD, 1, 1, new int[][][] { { { 1, 1, 1 } } }));
	feature.saveFile();
		Invoked MockClientUtility.loadImage\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showLoadedImage() Invoked\n IModelMock.getPatternBean() Invoked\nIModelMock.getImage() Invoked\n IModelMock.getImage() Invoked\n IImageProcessingViewMock.saveFile() Invoked\n IImageProcessingViewMock.setMessage() Invoked\nNot a valid path: null

testSaveFilePatterGenerated()	feature.loadImage();
	
	view.setUploadedImagePath("user/image.jpg");
	Image image = ImageFactory.builImage(ImageType.STANDARD, 1, 1,
	new int[][][] { { { 1, 1, 1 } } });
	this.model.setPatternBean(new PatternBean(image, "abc", new LinkedHashSet<>()));
	
	this.view.setFileName("user/pattern.txt");
	feature.saveFile();
		IImageProcessingViewMock.loadImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showLoadedImage() Invoked\n IModelMock.getPatternBean() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.saveFile() Invoked\n Invoked MockClientUtility.saveTextFile\n IImageProcessingViewMock.setMessage() Invoked\n File saved to file location: user/pattern.txt

testSaveFilePattern()	view.setUploadedImagePath("user/image.jpg");
	view.setProcessedImage(new BufferedImage(20, 20, BufferedImage.TYPE_INT_RGB));
	view.setPatternBean(new IPatternBeanMock());
		IModelMock.getPatternBean() Invoked\n IModelMock.getImage() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n.append("You must load and process the image before proceeding to save the image/ pattern."

	view.setFileName("user/pattern.txt");
	feature.saveFile();	

testSaveFilePatternBean()	feature.loadImage();
	Image image = ImageFactory.builImage(ImageType.STANDARD, 1, 1,
	new int[][][] { { { 1, 1, 1 } } });
	this.model.setPatternBean(new PatternBean(image, "abc", new LinkedHashSet<>()));
	this.view.setFileName(null);
	feature.saveFile();
		IImageProcessingViewMock.loadImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showLoadedImage() Invoked\n IModelMock.getPatternBean() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.saveFile() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n Pattern and file data must be given.\n file name:null

testSaveFileUnloaded()	view.setUploadedImagePath("user/image.jpg");	IModelMock.getPatternBean() Invoked\n IModelMock.getImage() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n.append("You must load and process the image before proceeding to save the image/ pattern.
testShowMessage()	feature.showMessage("Message");	IImageProcessingViewMock.setMessage() Invoked\nMessage"

generatePatternWithCustomColors()	view.setUploadedImagePath("user/image.jpg");
	view.setProcessedImage(new BufferedImage(20, 20, BufferedImage.TYPE_INT_RGB));
	
	view.setNumberOfChunksForPattern("40");
	view.setPatternBean(new IPatternBeanMock());
	model.setPatternBean(new IPatternBeanMock());
	Set<String> colors = new LinkedHashSet<>();
		append("IModelMock.getImage() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getSelectColorsForPattern() Invoked\n IImageProcessingViewMock.getNumberOfChunksForPattern() Invoked\n IModelMock.generatePattern() Invoked\n Invoked

	colors.add("1");
	colors.add("2");
	
	view.setSelectedColorForPattern(colors);
	
	feature.generatePatternWithCustomColors();
	StringBuilder expectedValue = new StringBuilder();
		MockClientUtility.getBufferedImage\n
		IImageProcessingViewMock.showProcessedImage() Invoked\n
		IImageProcessingViewMock.showPattern() Invoked\n
		IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n
		IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n

generatePatternWithCustomColorsChunkNull()	view.setUploadedImagePath("user/image.jpg");
	view.setProcessedImage(new BufferedImage(20, 20, BufferedImage.TYPE_INT_RGB));
	Set<String> colors = new LinkedHashSet<>();
	colors.add("1");
	colors.add("2");
	
	view.setSelectedColorForPattern(colors);
	
	feature.generatePatternWithCustomColors();
	StringBuilder expectedValue = new StringBuilder();
		IModelMock.getImage() Invoked\n
		IImageProcessingViewMock.getLoadedImagePath() Invoked\n
		Invoked MockClientUtility.loadImage\n
		IImageProcessingViewMock.getSelectColorsForPattern() Invoked\n
		IImageProcessingViewMock.getNumberOfChunksForPattern() Invoked\n
		IImageProcessingViewMock.setMessage() Invoked\n
		Number of chunks for pattern generation must be a number greater than zero

patternWithCustomColorSelected()	feature.patternWithCustomColorSelected();	IImageProcessingViewMock.popupCustomColorsDialog() Invoked\n

processImageBlurTest()	feature.editOptionSelection(EditingOptions.BLUR.toString());
	
	view.setUploadedImagePath("user/image.jpg");
	feature.processImage();
		IImageProcessingViewMock.resetPattern() Invoked\n
		IImageProcessingViewMock.setSelectedMenu() Invoked\n
		IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n
		IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n
		IImageProcessingViewMock.getLoadedImagePath() Invoked\n
		Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n
		IModelMock.getPatternBean() Invoked\n
		IImageProcessingViewMock.getLoadedImagePath() Invoked\n
		Invoked MockClientUtility.loadImage\n

		IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IModelMock.blurImage() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n;

processImageDitherTest()	feature.editOptionSelection(EditingOptions.DITHER.toString());
	
	 view.setUploadedImagePath("user/image.jpg");
	
	 view.setUserProvideSelectionValues("2"); feature.processImage();
		IImageProcessingViewMock.resetPattern() Invoked\n IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IModelMock.ditherImage() Invoked\n Invoked MockClientUtility.getBufferedImage\n IImageProcessingViewMock.showProcessedImage() Invoked\n

processImageDitherWrongCommandTest()	feature.editOptionSelection(EditingOptions.DITHER.toString());
	
	view.setUploadedImagePath("user/image.jpg");
	feature.processImage();
		IImageProcessingViewMock.resetPattern() Invoked\n IImageProcessingViewMock.setSelectedMenu() Invoked\n IImageProcessingViewMock.setProcessingValuesVisible() Invoked\n IModelMock.getImage() Invoked\nIModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\nIModelMock.getImage() Invoked\n IModelMock.getPatternBean() Invoked\n IImageProcessingViewMock.getLoadedImagePath() Invoked\n Invoked MockClientUtility.loadImage\n IImageProcessingViewMock.getUserProvideSelectionValues() Invoked\n IImageProcessingViewMock.toggleRemoveColorButton() Invoked\n IImageProcessingViewMock.toggleReplaceColorButton() Invoked\n IImageProcessingViewMock.setMessage() Invoked\n CommandGeneratorImpl: Dither image must be in format \"dither <noOfColors>\";

processImageEmptyCommand()	feature.editOptionSelection("");
	feature.processImage();
		IImageProcessingViewMock.setMessage() Invoked\n Error: unknown command > IImageProcessingViewMock.setMessage() Invoked\n You must select editing option before processing the image.

Controller Testing Controller of Batch processing (Project 4):

IUIController Testing		
	Method values	**Expected result**
Testing Constructor		
UIController(CommandController, CommandGenerator,		

PatternGenerator, ClientUtility) contains all instances		
Testing testBlur()		

Test if blur is called	in ="load salad.jpg\nblur\nsave salad-blur.jpg
	 controller.start();
		Invoked MockClientUtility.loadImage\n")
		.append("Image with name salad.jpg loaded successfully.\n")
		.append("Image blur, method: filterImage\n")
		.append("Image processing command \"blur\" called successfully.\n")
		.append("Invoked MockClientUtility.saveImageFile\n")
		.append("Image with name \"salad-blur.jpg\" saved successfully.\n

Testing testCommandController()		
test if command controller methods called	commandController.runCommand(image);	Image blur, method: filterImage\n" + "Image grayscale, method: transformImage\n
testing testCommandLength()		

test if commans are valid	in =load\nsharpen\nsave salad-sharpen.jpg
	start()
		"Load command must be in format \"load <filename>.\"\n

testing testDither()		

Test dithering method called	in = load salad.jpg\ndither 2\nsave salad-dither.jpg
	start()
		Invoked MockClientUtility.loadImage\n
		Image with name salad.jpg loaded successfully.\nImage processing command \"dither 2\" called successfully.\nInvoked MockClientUtility.saveImageFile\nImage with name \"salad-dither.jpg\" saved successfully.\n"

testing testGrayscale()		

test gray scale called]	in = load salad.jpg\grayscale\nsave salad-grayscale.jpg
	start()
		Invoked MockClientUtility.loadImage\n
		Image with name salad.jpg loaded successfully.\nImage processing command \"dither 2\" called successfully.\nInvoked MockClientUtility.saveImageFile\nImage with name \"salad-dither.jpg\" saved successfully.\n"

testing testMosaic()		

test mosaic called	in = load salad.jpg\nmosaic 2000\nsave salad-mosaic.jpg
	start()
		Invoked MockClientUtility.loadImage\n
		Image with name salad.jpg loaded successfully.\nImage processing command \"mosaic 200\" called successfully.\nInvoked MockClientUtility.saveImageFile\nImage with name \"salad-mosaic.jpg\" saved successfully.\n"

testing testPixelate()		

test pixelate called	in = load salad.jpg\npixelate 50\nsave salad-pixelate.jpg
	start()
		Invoked MockClientUtility.loadImage\n
		Image with name salad.jpg loaded successfully.\nImage processing command \"pixelate 50\" called successfully.\nInvoked MockClientUtility.saveImageFile\nImage with name \"salad-pixelate.jpg\" saved successfully.\n"

testing testSharpen()		

test sharpen methods called	in = load salad.jpg\nsharpen\nsave salad-sharpen.jpg
	start()
		Invoked MockClientUtility.loadImage\n
		Image with name salad.jpg loaded successfully.\nImage processing command \"sharpen\" called successfully.\nInvoked MockClientUtility.saveImageFile\nImage with name \"salad-sharpened.jpg\" saved successfully.\n"

testPattern()		

Test pattern genertae method is called	in = load salad.jpg\generate-pattern\nsave salad-pattern.txt
	start()
		Invoked MockClientUtility.loadImage\n
		Image with name salad.jpg loaded successfully.\nImage processing command \"generate-pattern\" called successfully.\nInvoked MockClientUtility.saveImageFile\nImage with name \"salad-pattern.txt\" saved successfully.\n"

CommandGeneratorImpl Testing		
Test Case	**Method values**	**Expected result**
Testing createCommand()		

Test if blur is called	commandObject =createCommand("blur")
	commandObject.execute()
		Invoked MockModel.blur()

Test if sharpen is called	commandObject =createCommand("sharpen")
	commandObject.execute()
		Invoked MockModel.sharpen()

Test if dither is called	commandObject =createCommand("dither 2")
	commandObject.execute()
		Invoked MockModel.dither()

Test if mosaic is called	commandObject =createCommand("mosaic 100")
	commandObject.execute()
		Invoked MockModel.mosaic()

Test if pixelate is called	commandObject =createCommand("pixelate 50")
	commandObject.execute()
		Invoked MockModel.pixelate()

Test if pattern is called	commandObject =createCommand("pattern 50")
	commandObject.execute()
		Invoked MockModel.pattern()

Test if grayscale is called	commandObject =createCommand("grayscale")
	commandObject.execute()
		Invoked MockModel.grayScale()

Test if sepia-tone is called	commandObject =createCommand("sepia_tone")
	commandObject.execute()
		Invoked MockModel.sepiaTone()