ImageEditor – главный класс приложения, реализующий GUI и отвечающий за обработку событий. Для того, чтобы реализовать GUI необходимо унаследовать данный класс от класса Application. Также, ImageEditor объединяет все классы приложения.

ExitStage – класс для реализации вспомогательного окна. Данное окно появляется при выходе из приложения. В этом окне реализовано «подтверждение выхода» из приложения. Данный класс, так же как и ImageEditor, наследуется от класса Application.

ChangeSizeStage – класс, в котором реализована операция уменьшения изображения. При нажатии на кнопку появляется окно. С помощью этого окна можно уменьшить размер изображения. Как и предыдущие классы, данный класс наследуется от класса Application.

ImageActions. Как следует из названия, данный класс хранит методы для работы с изображением. Например, метод для сохранения изображения, поворота, получения зеркального отражения и т.д.

ImageKeeper – класс, который хранит изображение, его высоту, ширину и массив пикселей.

FileChooserHelper – вспомогательный класс, который хранит объект класса FileChooser и его конфигурационный метод.

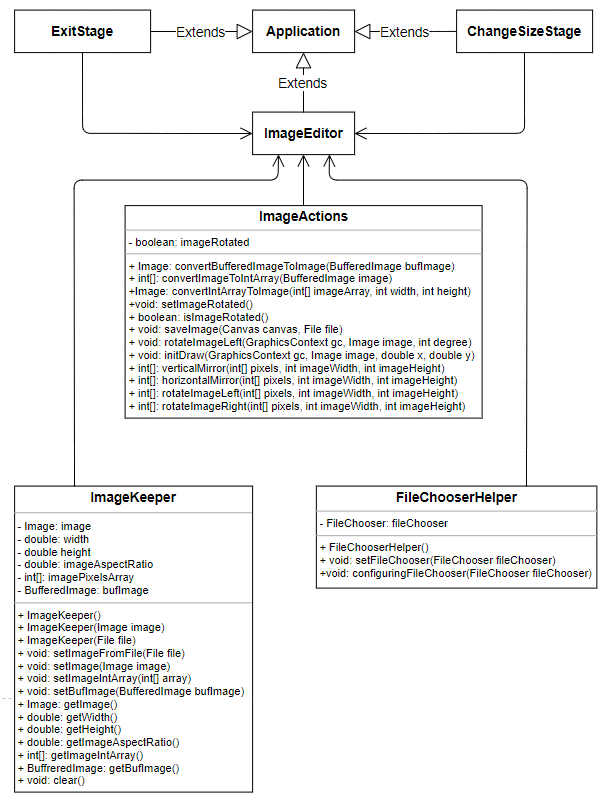


Рисунок 1 - UML - диаграмма разрабатываемого приложения

Реализация алгоритмов

Нет необходимости в описании всех методов, так как большинство из них стандартные или не имеют отношения к обработке изображения, поэтому в отчете будут описаны только методы непосредственное связанные с обработкой изображений.

Метод для сохранения изображения

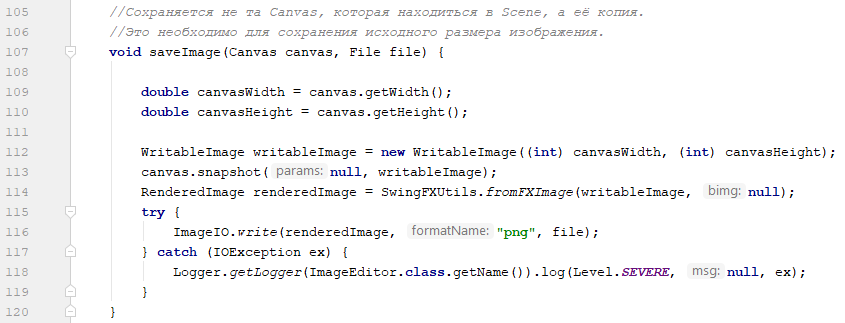


Рисунок 2 - Метод для сохранения изображения

После работы над изображением, его необходимо сохранить. Однако, необходимо сохранять и изменения проведенные над изображением. Для этого можно воспользоваться методом класса Canvas – snapshot.

**Строки 109 – 110:** объявляется переменные типа double и им присваивается значения ширины и высоты холста.

**Строка 112:** объявляется объект класса WritableImage и задается ее размеры. Размеры изображения равны соответствующим размерам холста.

**Строка 113:** вызывается метод объекта canvas – snapshot и результат записывается во writableImage.

**Строка 114:** объявляется объект класса RenderedImage и при помощи метода fromFXImage класса SwingFXUtils присваивается изображение, сохраненное в объекте writableImage.

**Строки 115 – 119:** используется конструкция try-catch. Данная конструкция необходима для разрешения непредвиденных ситуаций. Например, если, изображение, которое необходимо сохранить, не найдено. Для разрешения данной ситуации надо прописать внутри catch альтернативные команды, которые выполняются, если произошла непредвиденная ситуация.

**Строка 116:** изображение сохраняется при помощи метода write класса ImageIO. В метод write передаются:

1. Объект класса RenderedImage, в котором сохранено изображение.
2. Формат, в котором необходимо сохранить изображение.
3. Объект класса File.

**Строка 118:** если возникала ошибка при попытке сохранения изображения, возвращается ее название.

Метод для создания зеркального отражения относительно вертикальной линии

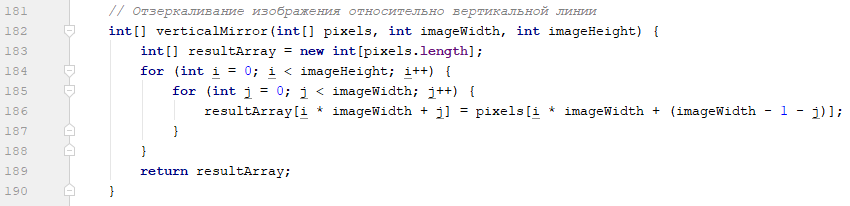


Рисунок 3 - Метод для создания зеркального отражения относительно вертикальной линии

Любое изображение – это массив пикселей. Другими словами, каждый элемент массива хранит в себе значение соответствующего пикселя. В данном методе, преобразования производятся над массивом пикселей, а не над специальными классами, такими как Image, WritableImage или RenderedImage.

Идея метода проста, при помощи простых преобразований из матрицы:

необходимо получить матрицу:

Для получения данной матрицы, в методе используется два массива и двойной цикл for.

Первая матрица перебирается в правильном порядке (слева на право), а вторая наоборот, с права на лево.

**Строка 183:** объявляется вспомогательный массив типа int, размер которого равен размеру массива пикселей.

**Строка 184 – 188:** преобразования, описанные выше.

**Строка 189:** возвращается преобразованная матрица.

Метод для создания зеркального отражения относительно горизонтальной линии

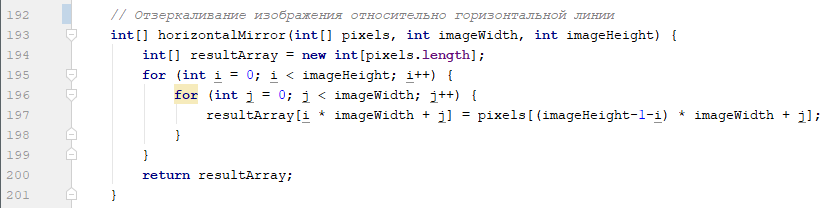


Рисунок 4 - Метод для создания зеркального отражения относительно горизонтальной линии

Идея данного метода похожа на идею предыдущего метода.

Из матрицы:

надо получить матрицу:

Как и в предыдущем методе, первая матрица перебирается в правильном направлении, а вторая матрица сверху вниз. Сначала, перебирается последняя строка, потом средняя, а в конце первая.

**Строка 194:** объявляется вспомогательный массив типа int, размер которого равен размеру массива пикселей.

**Строки 195 – 199:** преобразования, описанные выше.

**Строка 200:** возвращается преобразованная матрица.

Метод для поворота изображения на 90 градусов по часовой стрелке

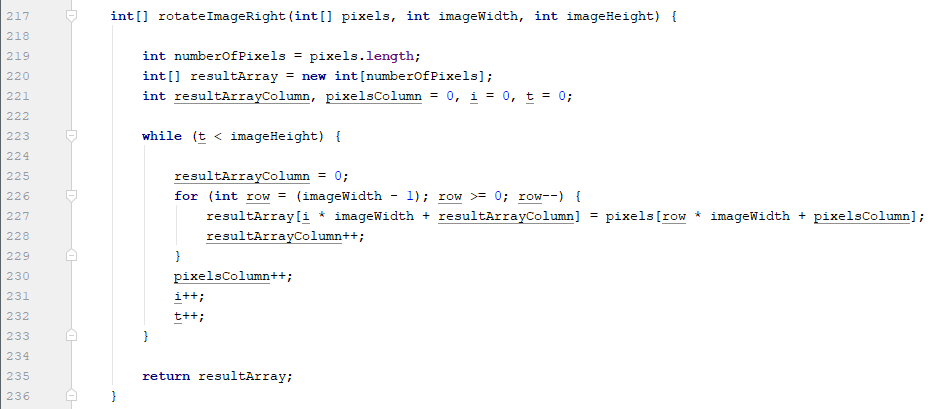


Рисунок 5 - Метод для поворота изображения по часовой стрелке

Как и раньше, в данном методе будем преобразовывать матрицу пикселей.

Из матрицы:

необходимо получить матрицу:

В данном случае, способ преобразования матрицы сложнее, чем в предыдущих методах. В место двойного цикла for используется один цикл while и один цикл for.

**Строки 219 – 221:** объявление необходимых переменных.

**Строки 223 – 233:** цикл while, который изменяется от 0 до значения высоты изображения. Данный цикл отвечает за перебор строк матрицы.

**Строки 226 – 229:** цикл for, который изменяется от (ширина изображения – 1) до 0.

**Строки 230 -232:** увеличение соответствующих значений.

**Строка 235:** возвращается преобразованная матрица.

Метод для поворота изображения на 90 градусов против часовой стрелки

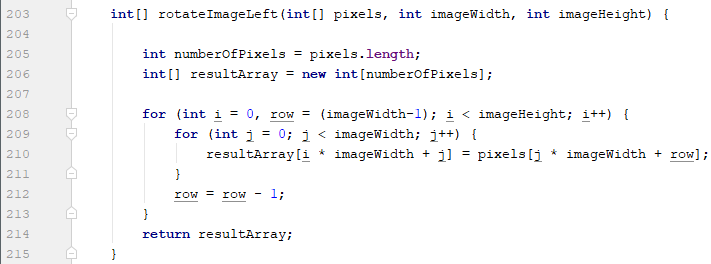


Рисунок 6 - Метод для поворота изображения против часовой стрелки

Из матрицы:

необходимо получить матрицу:

В отличии от предыдущих методов, в данном методе используется дополнительная переменная row. Она отвечает за переход между столбцами матрицы пикселей.

**Строки 205 – 206:** объявление вспомогательных переменных.

**Строка 208:** объявление первого цикла for. В данном цикле объявляются два переменных: i, который меняется от 0 до значения высоты изображения и row, который меняется от значения ширины изображения до 0.

**Строка 210:** записывается в массив resultArray, который перебирается в правильном порядке, значения из массива пикселей, который перебирается по столбцам, начиная с последнего.

**Строка 212:**  уменьшение значения row на 1.

**Строка 214:** возвращается преобразованная матрица.