ФЕДЕРАЛЬНОЕ АГЕНТСТВО МОРСКОГО И РЕЧНОГО ТРАНСПОРТА

ФЕДЕРАЛЬНОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«Морской государственный университет имени адмирала Г.И. Невельского»

Факультет электроники и информационных технологий

Кафедра Вычислительной техники

Дисциплина «Программирование»

**Курсовая работа**

«Векторный редактор. Объектно-ориентированное программирование.»

Выполнил: студент гр. *214.21*

*Гармышев Дмитрий Евгеньевич*

Проверил: преподаватель

Хованец В.А.

Оглавление

[Введение 3](#_Toc74206560)

[Задание 4](#_Toc74206561)

[План разработки проекта 5](#_Toc74206562)

[Набросок тестового проекта 5](#_Toc74206563)

[Разработка новой структуры данных 5](#_Toc74206564)

[Хранение точек в древовидной структуре. 6](#_Toc74206565)

[Хранение точек и фигур в списках, но фигуры состоят из рёбер. 12](#_Toc74206566)

[Хранение точек и фигур в списках, без использования рёбер. 14](#_Toc74206567)

[Разработка и тестирование программы 15](#_Toc74206568)

[Общее описание проекта 16](#_Toc74206569)

[Структура модулей 16](#_Toc74206570)

[Пакет активностей 16](#_Toc74206571)

[Пакет адаптеров 17](#_Toc74206572)

[Пакет компонентов 17](#_Toc74206573)

[Требования к оформлению кода и выбору имен 19](#_Toc74206574)

[Алгоритмы 20](#_Toc74206575)

[Тексты модулей 21](#_Toc74206576)

[Инструкция к приложению 22](#_Toc74206577)

[Заключение 23](#_Toc74206578)

[Список литературы 24](#_Toc74206579)

# Введение

Основная тема данной работы – векторный редактор, который позволяет быстро и удобно создавать изображения, основанные на геометрических фигурах. Его особенность заключается в том, что мы можем легко создавать и редактировать фигуры.

Для создания этого проекта было использовано Объектно-ориентированное программирование (далее ООП). Такой подход позволяет явно представлять объекты (например, объект точки или фигуры), указывать их свойства и функции. Помимо ООП, уделено внимание алгоритмам поиска фигур, точек и их обработки.

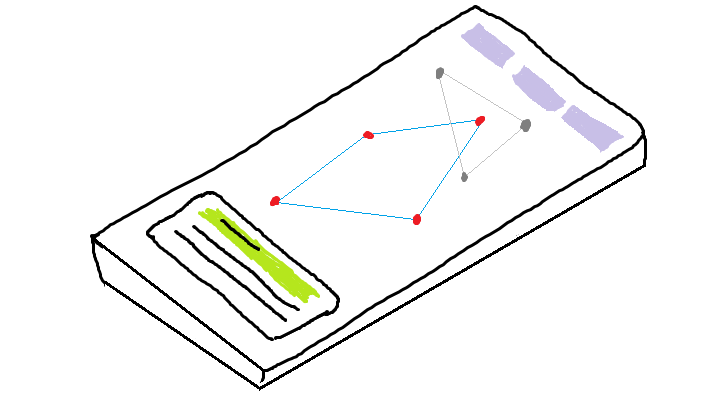
Проект написан для ОС Android, на языке Java, с использованием событийно-управляемого программирования (например, нажатие на экран, кнопки и т.д.).

Используемые инструменты включают в себя IDE Android Studio для написания кода, и браузер для поиска полезной информации в процессе разработки ПО.

Основная цель данного проекта – закрепить знания в области ООП, научиться создавать сложные структуры данных и алгоритмы для их обработки.

Структура данных включает в себя две части. Первая часть включает в себя все точки, которые отображаются на экране, и хранит только их координаты. Вторая часть хранит в себе все данные о фигурах, их шаблонах. Фигуры в данной структуре не содержат в себе объекты точек, а только ссылки на них в виде индексов.

P.S. Мои знания на момент написания программы и на момент написания отчёта очень рознятся, и сейчас я бы сделал всё по другому.



# Задание

1. Создать новую фигуру щелчками мышью – суть этого задания заключается в том, чтобы при нажатии на экран, создавались точки и соединялись линиями, тем самым образуя фигуру.
2. Выбрать нужные фигуры или одну фигуру щелчками мышью на фигуре. По нажатию на экран, и по полученным координатам нажатия мы должны правильно распознать, какую фигуру мы выбрали.
3. Удалить выбранные фигуры. Данное задание вытекает из второго. Просто удаляем выбранные фигуры из списка, а также точки, которые принадлежат этой фигуре.
4. Копировать выбранные и поместить в нужное место. Данное задание вытекает из второго. Мы создаём копии выбранных фигур и их точек, но уже смещаем их в другое место, куда нажал пользователь.
5. Переместить выбранные фигуры. Данное задание вытекает из второго. Мы просто перемещаем выбранные фигуры, туда куда нажал пользователь.

# План разработки проекта

## Набросок тестового проекта

Для того чтобы приблизительно понять, что конкретно требуется разработать в этом проекте, было создано тестовое приложение (черновик), в котором использовались самые примитивные структуры данных (довольно неоптимизированные, и не пригодные для дальнейшего развития проекта), не самые лучшие алгоритмы, но зато примерно было понятно куда двигаться дальше. Из функций, это приложение включало в себя создание одной фигуры, редактирование её точек (простое перемещение и удаление их), и элементы событийно-управляемого программирования.

Пускай на это приложение было потрачено время, зато я смог понять с какими «подводными камнями» я могу столкнуться в процессе разработки основного приложения, смог определить для себя самые важные функции приложения, и продумал дальнейшую архитектуру (не структур данных, а моментов касаемо работы самого приложения), тем самым значительно снизив вероятность глобальных переработок в дальнейшем.

## Разработка новой структуры данных

Данный этап является тоже самым немаловажным. Изначально было очень много вариантов создания структуры данных, основанных на разных принципах.

1. Древовидная структура – в которой точки хранятся в виде дерева по 4 направлениям.
2. Структура, в которой точки хранятся в отдельном списке, но фигура состоит из рёбер.
3. Структура, в которой точки хранятся в отдельном списке, но фигура содержит в себе просто ссылки на рёбра.

Далее будут кратко описаны все эти принципы, и их реализация.

### Хранение точек в древовидной структуре.

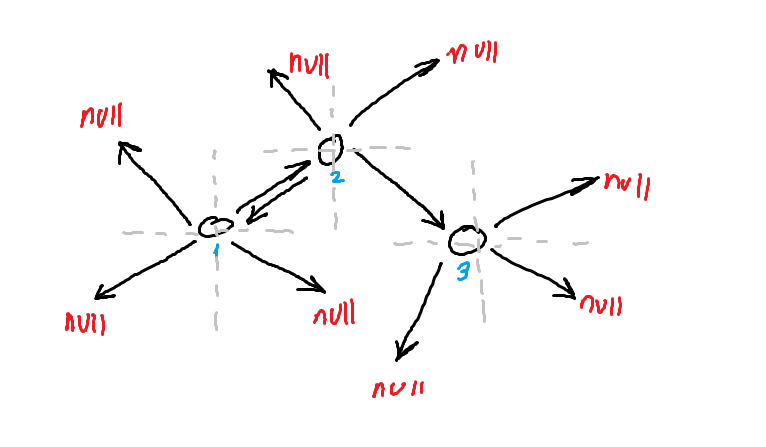


Рис 1. Образное представление древовидной структуры

Основная суть данной структуры заключается в том, что точки хранятся в виде ссылок друг на друга, и учитывается их положение относительно друг друга.

Например, если в структуре уже есть точки, и я хочу добавить новую, то я прохожусь по каждой точке и проверяю, в каком направлении лежит новая точка относительно старой. Если оно правее и выше, то я иду по ветке дальше, и опять сверяю направление. Как только я нахожу пустое место, оно заполняется ссылкой на новую точку. При этом у каждой новой точке остаётся ссылка на родительскую точку.

У этого способа есть свои плюсы и недостатки.

Плюсы:

* Очень быстрый поиск точек. Нам не нужно проходиться по каждой точке в структуре (как например при использовании списков), мы просто идём по ветвям в поисках ближайших координат и затрагиваем только интересующие нас элементы.

Минусы:

* Редактирование точек. Дело в том, что при редактировании (перетаскивании точек), меняется их положение относительно друг друга. И поэтому, при каждой смене координат приходилось удалять точку из списка и создавать новую, но уже с новыми координатами, которая встанет на своё место. Из-за этого сбивались все ссылки на точки в структуре фигур.
* Сложный контроль над точками. Мы не можем в любой момент и быстро обратиться к элементу по индексу (как например в списках или массивах), поэтому постоянно приходилось искать точку снова по всей структуре. (Именно на момент написания программы у меня были некие пробелы в знаниях, и сейчас я понимаю, что можно было просто обращаться к точке по ссылке на неё).
* Использование более сложных алгоритмов для обработки точек и фигур.
* Более сложный алгоритм удаления точек из структуры, т.к. нужно учитывать ссылки дочерних и родительских точек друг на друга.

Реализация данной структуры на Java.

Вот такой наполнение у класса Point (скриншот без процедур и методов).

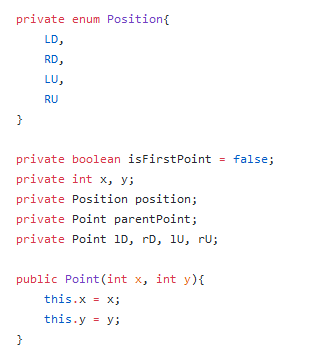


Рис 2. Наполнение класса Point в древовидной структуре.

Тут у нас хранится флаг, дающий знать, какая это точка в структуре, первая или нет, её координаты, текущее положение точки относительно родительской точки, ссылка на родительскую точку, 4 ссылки на дочерние точки, где каждая ссылка отвечает за своё направление.

Давайте рассмотрим самые основные процедуры в этом классе.

Процедура добавления новых точек в структуру:

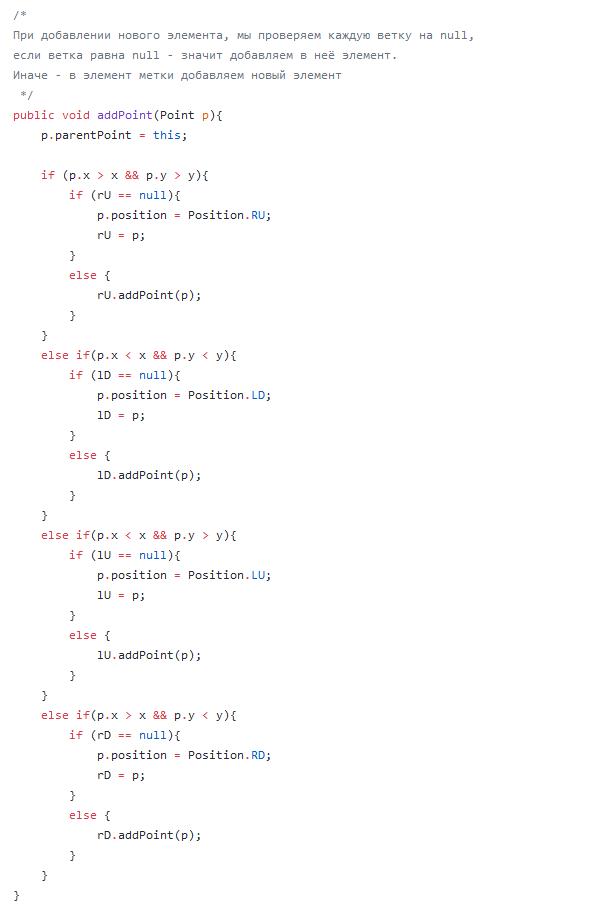


Рис 3. Процедура добавления новых точек в древовидную структуру.

Поиск точек в структуре. То о чём говорилось выше. Мы не просто перебираем каждый элемент, а идём по ветвям и ищем нужный элемент.

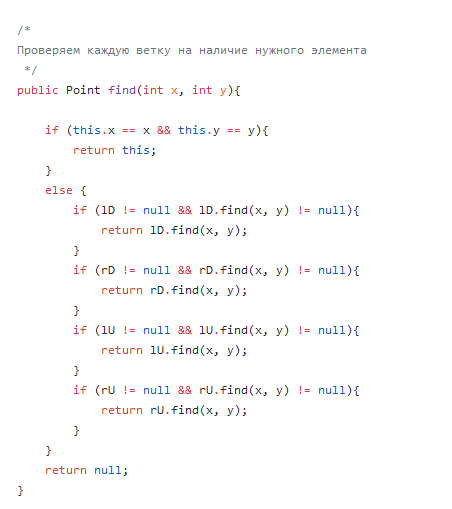


Рис 4. Реализация поиска точек в древовидной структуре.

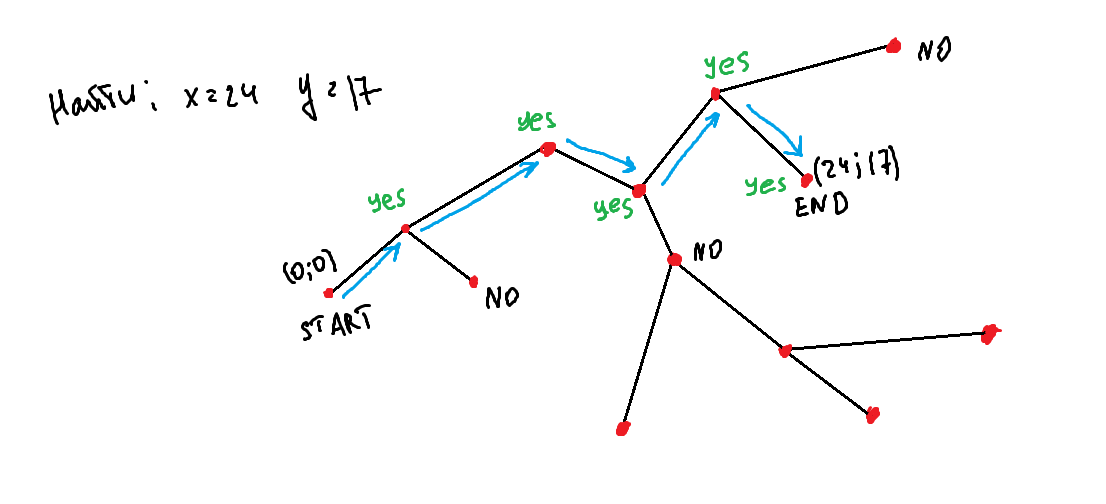


Рис 4. Визуальное представление поиска точек в древовидной структуре.

Удаление точки из структуры. Мы должны перекинуть ссылки с родительских и дочерних элементов друг на друга.

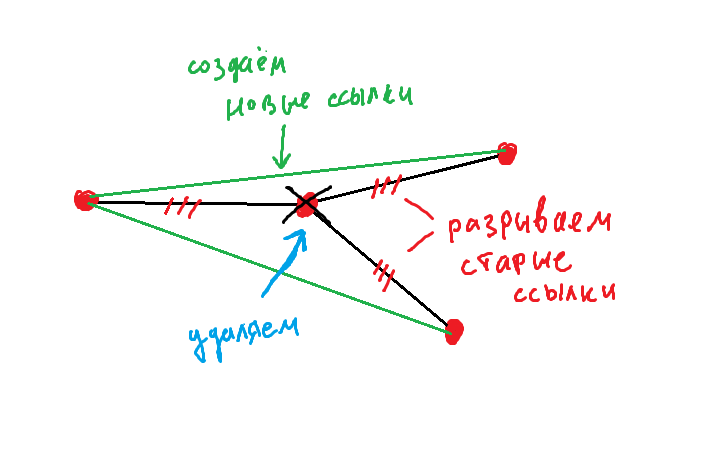
 

Рис 5 и 6. Алгоритм удаления точек из структуры, а также его визуальное представление.

Также стоит отметить то, как именно использовалась данная структура данных, именно на этапе разработки этой структуры.

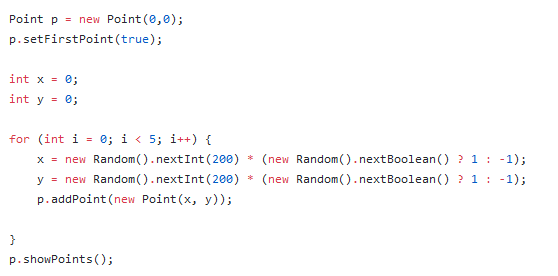


Рис 7. Использование древовидной структуры в программе.

Добавляли новые точки мы в переменную *p,* тем самым создавая целую древовидную структуру в дальнейшем. (Просто закидывали точку в точку).

И на тот момент предполагалось, что одна такая переменная типа *Point* – есть готовая фигура, т.к. она сразу хранит в себе структуру точек. Т.е. мы могли создать список из *Point*’ов, и использовать его как список фигур. Отсюда и вытекали все дальнейшие сложности работы с такой структурой. Правда появлялись они из-за того, что, на тот момент я не знал, как правильно это всё организовать.

По итогу я вынужден был отказаться от такой структуры и пойти по более простому пути. Но я планирую доработать эту структуру и избавиться от всех её недостатков, т.к. считаю, что у неё есть потенциал.

### Хранение точек и фигур в списках, но фигуры состоят из рёбер.

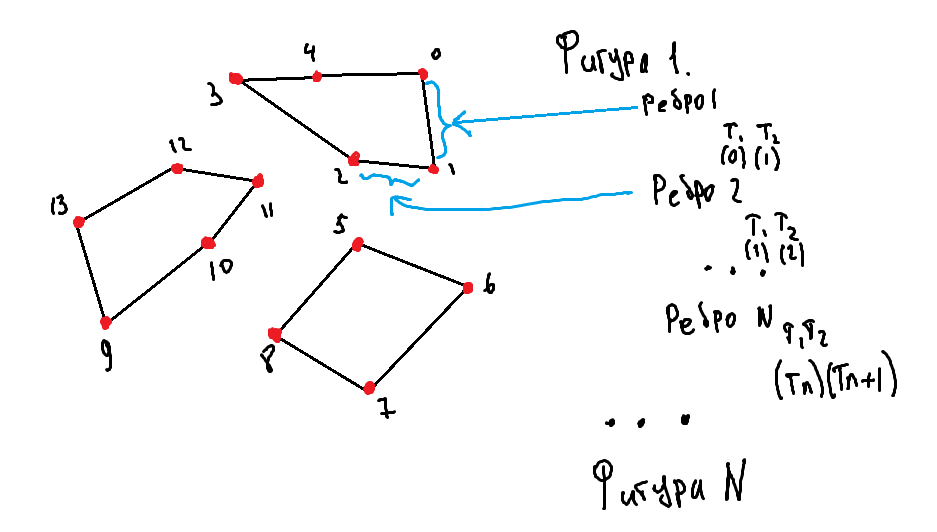


Рис 8. Визуальное представление структуры с рёбрами.

Суть данной структуры заключается в том, что все точки хранятся в отдельном списке, как и фигуры. Однако фигуры состоят из рёбер. Ребро тем самым ссылается на две точки, начало и конец ребра.

С такой структурой работать уже проще, чем с первой. Появляется больше контроля над всей структурой благодаря индексам, становятся проще алгоритмы. Но всё же нашлись свои весомые недостатки.

Плюсы данной структуры:

1. Более просыте алгоритмы поиска, удаления и редактирования точек и фигур в структуре.
2. Больше контроля над всей структурой благодаря индексам.
3. Т.к. фигуры хранятся отдельными объектами, появилось обширное поле для модернизации этих фигур (настройка их параметров, изменение цвета и т.д.).

Минусы:

1. По сравнению с первой структурой поиск точки представляет собой проход по каждой точке, из-за чего скорость поиска более медленная. Однако это спорный вопрос. Ибо если брать маленькое количество точек, то скорость будет мало чем отличаться и особой разницы тут не увидеть.

Но заметное преимущество первой структуры можно увидеть при использовании более миллиона точек (условно), где проход по каждой будет занимать достаточно много времени.

1. Из-за того что использовались рёбра, а в одном ребре хранились две ссылки на точки, то у нас бессмымленно удваивалась работа при редактирвоании фигур и точек.

Реализация данной структуры на Java.

К сожалению, исходного кода той структуры не осталось. Но от неё тоже пришлось отказаться, из-за её несовершенства и использования рёбер, что в какой-то мере бессмысленно удваивало работу.

### Хранение точек и фигур в списках, без использования рёбер.

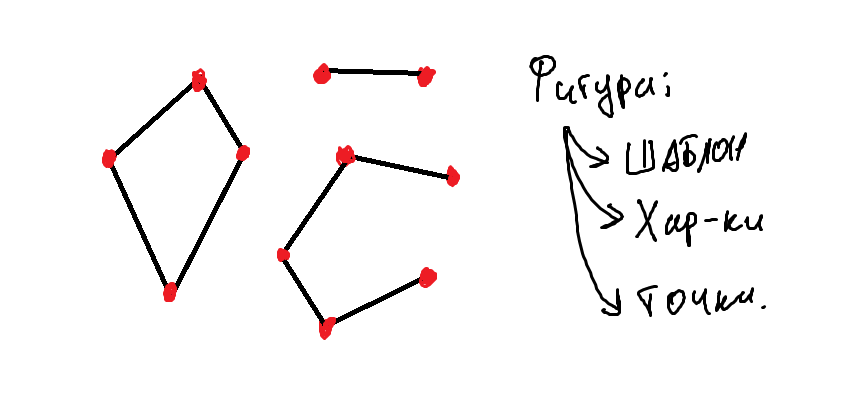


Рис 8. Визуальное представление структуры без рёбер.

Данная структура полностью наследовалась от предыдущей, и всё что в неё поменялось – использование индексов точек вместо рёбер, для построения фигуры. Именно этот шаг упростил дальнейшую работу, и позволил реализовать шаблоны фигур, выполнение каких-либо операций с ними и т.д.

От предыдущей структуры ей передались и плюсы, и минусы (из минусов только один, связанный с производительностью).

Описывать её реализацию в этом разделе смысла нет, потому что всё это ещё будет расписано далее.

## Разработка и тестирование программы

Разработка полноценного приложения началась после выполнения двух предыдущих пунктов, что вполне объяснимо. Сначала была подготовлена графическая часть приложения, затем разработана отдельно структура данных, и по итогу эти две системы слились воедино.

Сначала приложение имело скудный функционал, позволяющий создавать одну фигуру и редактировать её точки, затем, по мере продвижения наращивались новые «фичи». Например, разные режимы работы с фигурами, различные новые окна в приложении т.д.

Тестирование производилось на разных этапах и на разных уровнях.

Т.е. тестировалась как структура данных отдельно, во время разработки, так и графическая часть приложения. И со временем, оно всё переросло в состояние, когда тестирование этих обоих частей происходило одновременно.

Например, добавление точек на экран производится через касание (графическая часть), и через дальнейшее занесение этой точки в структуру данных (часть данных).

Единственное, что не вошло в этот план – построение архитектуры самого приложения. И я очень об этом жалею. Если бы изначально я изучил нюансы разработки этого приложения, продумал как внедрить структуру данных в него, то сейчас это приложение выглядело бы гораздо лучше, как внутри, так и снаружи. Это бесценный опыт для меня, когда я на живом примере увидел, почему важно планирование и проектирование.

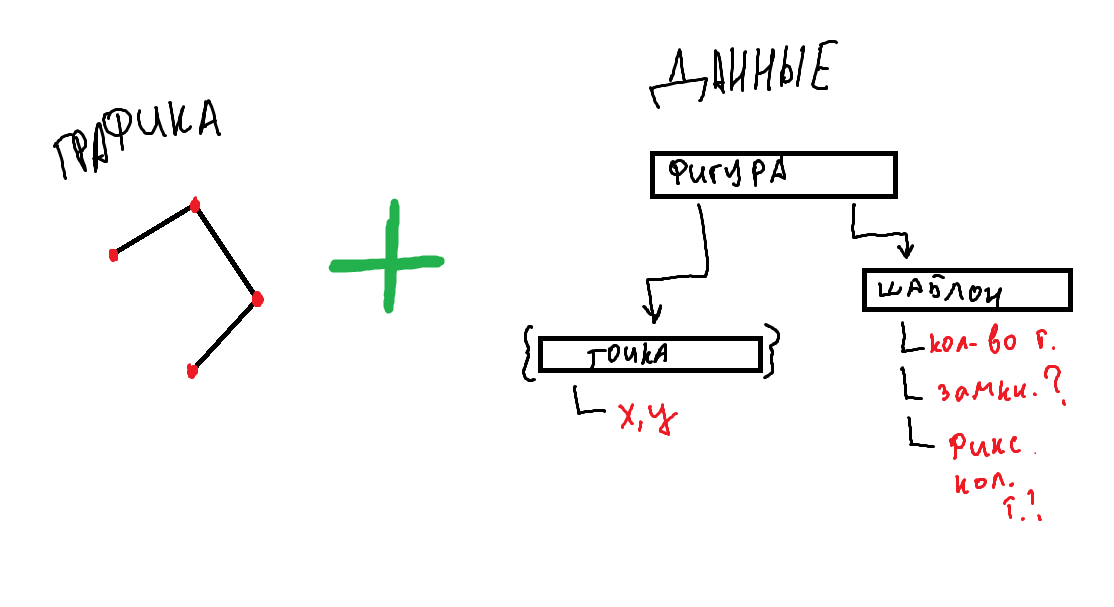


Рис 9. Визуальное представление смешения двух частей приложения.

# Общее описание проекта

## Структура модулей

Этот проект работает на ОС Android, поэтому в структуре классов встречается не мало классов, которые связаны с этой системой. Например, активности, фрагменты и т.д.

Поэтому, сначала я покажу все пакеты, которые содержатся в проекте, а затем разберу каждый из них, включая вложенные в них классы.

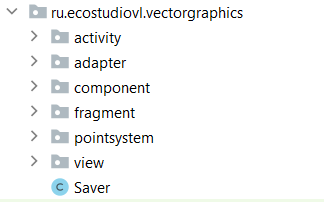


Рис 9. Структура пакетов программы.

### Пакет активностей

В системе Android, приложение состоит из окон, которые могут переключаться между собой, они называются активностями (Activity).

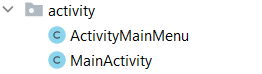


Рис 10. Структура пакета активностей

В данном пакете хранятся рабочий экран (MainActivity), и главное меню (ActivityMainMenu).

Рабочий экран представляет собой область, где мы строим изображения.

Главное меню отвечает за навигацию по приложению.

### Пакет адаптеров

Адаптеры (Adapter, в данном случае используются RecyclerView адаптеры), нужны для того, чтобы преобразовать данные (допустим из списка) в графическое представление, и вывести их на экран.

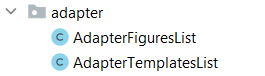


Рис 11. Структура пакета адаптеров

Конкретно в этом пакете хранятся адаптеры для списка фигур, и для списка шаблонов. Адаптеры так-же позволяют реализовать выбор данных, проводить с ними какие-либо операции (например, удаление или изменение данных).

### Пакет компонентов

В данном пакете хранятся глобальные компоненты программы, к которым можно обратиться из любого места программы, записать или взять из них информацию.

Обращение к ним из любого места программы возможно, благодаря тому что, эти классы глобальные, как и их данные, и все эти классы реализуют паттерн Singleton.

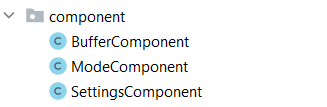


Рис 12. Структура пакета компонентов

BufferComponent – позволяет проводить операции связанные с выбором фигур. Его суть заключается в том, чтобы хранить индексы на выбранные пользователем фигуры, которые мы потом сможем прочитать в любой удобный момент.

ModeComponent – хранит в себе доступные и текущие режимы работы программы, режимы работы с фигурами и т.д.

SettingsComponent – обеспечивает доступ к глобальным настройкам приложения, для того чтобы мы смогли узнать интересующие нас параметры в любой момент работы программы.

1. **Типы данных и имена переменных**

Указываем основные структуры данных и особенности их использования, если они есть. Описываем основные переменные и обязательно глобальные переменные с объяснением того, почему они сделаны глобальными.

1. **Структура меню проекта**

Показываем меню проекта в виде образов экрана. Даем краткое описание пунктов меню и действий в этих пунктах.

1. **Описания основных процедур**

Описываем самостоятельно разработанные в проекте основные процедуры и реализуемые ими алгоритмы.

1. **Примеры графических построений в приложении**

Показываем характерные примеры графических построений в виде образов экранов.

# Требования к оформлению кода и выбору имен

Очень важный раздел. Здесь должны быть приведены основные требования к оформлению кода, способам формирования имен и всё достаточно подробно. Ищите то, что я давал в лекциях и что написано в методических указаниях, ищите информацию в Интернет, чем больше вы напишете, тем лучше. Напишите какие префиксы в именах вы используете, укажите в какой нотации записываются ваши имена в проекте. Также можете объяснить, почему соглашения об оформлении кода и выборе имен так важны.

# Алгоритмы

Здесь размещаются словесные описания алгоритмов для каждого из ваших индивидуальных заданий. Как показывает опыт, написание алгоритмов для вас немалая проблема, поэтому, тренируйтесь, пробуйте, пишите. Обычно пытаются сразу начинать кодирование, а в голове нет даже и понимания, как надо решать задачу. Это важный блок вашего отчета.

# Тексты модулей

Здесь вы должны разместить тексты всех модулей проекта. Предлагается вставлять в отчет тексты модулей в виде копий экрана в приложении Lazarus. Т.е., в редакторе кода выбираете модуль, пролистываете до нужного мета, копируете активное окно при помощи Alt-PrtScr или другим способом (в Win 10, например, есть функция копии в буфер произвольной области экрана) и вставляете копии в отчет так, чтобы не было пропусков в тексте программы. Подсвеченные Лазарусом тексты удобнее читать, да и копии экранов не сканируются Антиплагиатом.

# Инструкция к приложению

Напишите инструкцию, позволяющую освоить разработанный вами векторный редактор.

# Заключение

Сформулируйте, что вы сделали в ходе исполнения проекта. Что разработано, кратко функционал и, по-вашему, какие достоинства имеет этот проект.

# Список литературы

В списке литературы обязательно указываются нижеперечисленные источники. В продолжение списка добавляете ту литературу и те ссылки в Интернете, которые вы использовали в своей работе в дополнение к основным.

1. Хованец В.А. Технология программирования и абстрактные типы данных// Методические указания – Владивосток: Изд-во МГУ им. адм. Г.И. Невельского, 2011. – 36 с.
2. Хованец В.А. Событийно-управляемое программирование// Методические указания – Владивосток: Изд-во МГУ им. адм. Г.И. Невельского, 2011. – 26 с.
3. Хованец В.А. Курс лекций «Программирование» [Текст]: учеб-метод, пособие / В.А. Хованец. – Владивосток: Изд-во МГУ им. адм. Г.И. Невельского, 2014. – 90 с. (электронный вариант).
4. Хованец В.А. Курс «Программирование» в системе электронного обучения МГУ им. адм. Г.И. Невельского, 2 курс ФЭИТ ИВТ, осенний семестр 2019-2020, <http://course.msun.ru/course/view.php?id=10179>.
5. Хованец В.А. Курс «Программирование» в системе электронного обучения МГУ им. адм. Г.И. Невельского, 2 курс ФЭИТ ИВТ, весенний семестр 2019-2020, <http://course.msun.ru/course/view.php?id=10178>.
6. Форум по курсовому проекту «Векторный редактор», <http://course.msun.ru/mod/forum/view.php?id=52933>
7. https://metanit.com/java/tutorial/5.8.php