|  |  |  |
| --- | --- | --- |
|  |  |  |
| МИНОБРНАУКИ РОССИИ | | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |

Институт информационных технологий

Кафедра математического обеспечения и стандартизации информационных технологий

**Отчет по практической работе №9**

по дисциплине

«Разработка мобильных приложений»

|  |  |
| --- | --- |
| **Выполнил:**  Студент группыИКБО-33-21 | Привезенцев Д.А. |
| **Проверил:**  Ассистент | Новичков Д.Е. |

Москва 2023 г.

**СОДЕРЖАНИЕ**

[1. ЗАДАНИЕ 1 3](#_Toc135962886)

[1.1 Условие задания 3](#_Toc135962887)

[1.2 Решение задания 3](#_Toc135962888)

[2. ЗАДАНИЕ 2 5](#_Toc135962889)

[2.1 Условие задания 5](#_Toc135962892)

[2.2 Решение задания 5](#_Toc135962893)

[ВЫВОД 6](#_Toc135962894)

# ЗАДАНИЕ 1

## Условие задания

Обновить структуру разработанного на прошлых практических работах мобильного приложения по выбранной предметной области на основе общих рекомендаций к архитектуре мобильных приложений.

## Решение задания

Для удобства разработки и оптимизации работы приложения существуют паттерны архитектуры приложения, они предполагают у каждого приложения наличие слоя UI, слоя данных и опционального доменного слоя. UI слой — слой с которым взаимодействует пользователь, слой отображающий информацию приложения. При изменении данных слой должен отразить изменения. UI слой подразделяется на два пакета — StateHolders, элементы которые хранят данные, открывают к ним доступ для UI и работают с логикой и Views, элементы которые отображают данные на экране. Слой данных в приложении содержит логику хранения данных*,* а именно правила, по которым приложение создаёт, хранит и изменяет данные. Именно благодаря данной логике приложение имеет ценность. UI слой представлен на рисунке 1, слой данных на рисунке 2. На рисунках 3-4 представлены классы модели и объекта.

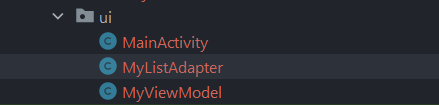


Рисунок 1 – Слой UI

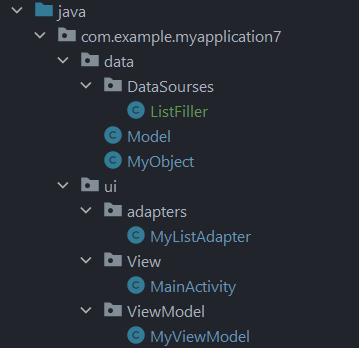


Рисунок 2 – Слой данных

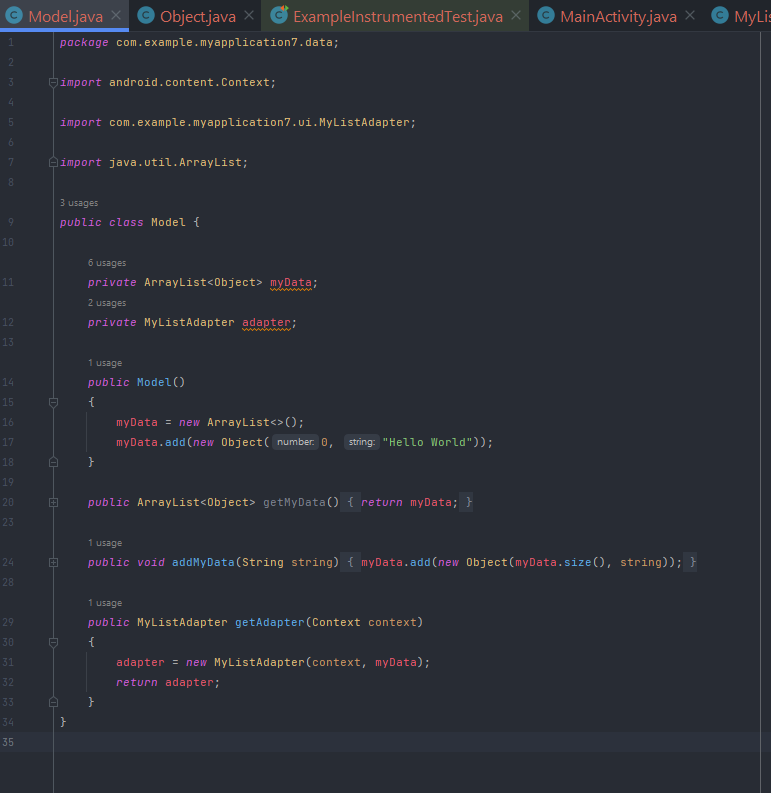


Рисунок 3 – Класс модели в папке data

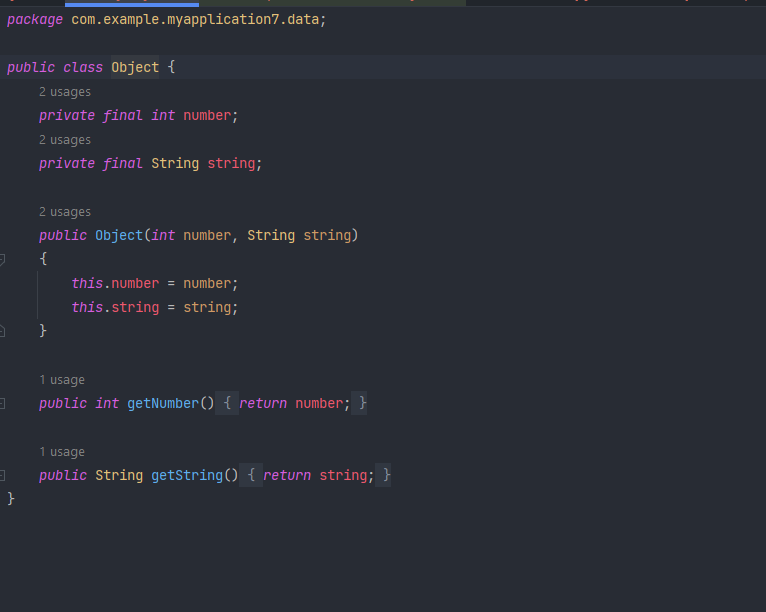


Рисунок 4 – Класс объекта в папке data

# ЗАДАНИЕ 2



## Условие задания

Реализовать архитектуру разрабатываемого приложения по паттерну MVVM с использованием компонентов LiveData в элементах паттерна.

## Решение задания

Model View ViewModel — паттерн позволяющий связывать элементы представления со свойствами и событиями View-модели. Объект ViewModel является связующим звеном между UI слоем и слоем данных, он предоставляет команды для View и привязывает View к Model. Когда Model обновляется, соответствующие View обновляются через связывание данных. Каждый компонент View создается собственный делегат, обрабатывающий и связывающий его с Model. При пересоздании View нет необходимости пересоздавать делегат, так как связь между компонентами паттерна реализуется не на прямую. Помимо этого, положительным фактором в возможность такой логики выступает факт того, что ViewModel имеет собственный жизненный цикл и не зависит от жизненного цикла View. Все ViewModel хранятся и зависят от состояния ViewModelScope, который прекращает свое существование только на момент уничтожения своего ViewModelStoreOwner. Относительно изначального разделения приложения на слои был добавлен класс ListAdapter. Схему паттерна MVVM можно заметить на рисунке 5. Класс ListAdapter показан на рисунке 6.



Рисунок 5 – схема паттерна MVVM

# ВЫВОД

В данной работе были разобраны паттерны проектирования приложений, стало понятно зачем они нужны, как их надо соблюдать и разобран конкретный паттерн MVVM и получены знания по работе с классами ViewModel. Для реализации некоторых задач ViewModel был изучены основы работы с LiveData.