Гопиенко Александр КИ22-17/2Б

Документация к коду.

Данный код представляет собой Android приложение на языке Java. Приложение предназначено для записи задач пользователя, которые он планирует выполнить.

Общая идея приложения заключается в том, чтобы пользователь мог добавлять задачи, к которым добавлять комментарии, дату и время выполнения с возможность удаления или редактирования. При нажатии на выполнение задачи пользователю выдаётся случайная мотивационная фраза.

Из приложения были убраны несколько функций по некоторым причинам. Функция создания подзадач была убрана по причине неэффективности и

не нужности в коде. Функция повтора и добавления приоритета задачи из-за ненадобности. Также была убрана регистрация и авторизация пользователей и админка из-за неэффективности и сложности реализации.

В моём коде используется паттерн "Стратегия" (Strategy). Он нужен для обеспечения механизма случайной выдачи мотивационных фраз, при котором класс MotivationGenerator принимает экземпляры классов CasualMotivationStrategy, GPeopleMotivationStrategy, HumorMotivationStrategy, которые содержат список из мотивационных фраз. В классе ToDoAdapter с помощью экземпляра класса вызывается метод generateMotivation в переменную motivation сохраняем рандомную мотивационную фразу, которую выводим с помощью AlertDialog.

Преимущества использования паттерна "Strategy" в моём коде:

* Паттерн Strategy позволяет скрыть детали реализации алгоритма из класса ToDoAdapter, тем самым не засоряя его.
* Предоставляет возможность замены одного класса другим в процессе выполнения программы.
* Проще модифицировать работу данной функции.

В моём коде паттерн "MVC" используется для получения данных из всплывающего окна, куда пользователь вводит свои данные о задачах. Эти данные добавляются или обновляются в базе данных. В программе класс ToDoModel берёт на себя роль модели, класс AddNewTask берёт на себя роль контроллера и XML файл add\_new\_task роль View.

Преимущества использования паттерна "MVC" в моём коде:

* Единая система концепции. Паттерн реализует единую глобальную архитектуру. Через графический интерфейс пользователь вводит свои данные, которые обрабатываются в контроллере, а затем Model работает с полученными данными.
* Облегчает работу с базой данных.
* Упрощён режим отладки приложения, так как весь процесс визуализации сконцентрирован в одном блоке.

Паттерн "Декоратор" не применим для моего кода:

Этот паттерн применяется, когда необходимо динамически подключить поведение к объекту. Моё приложение, не предусматривает такой реализации объектов класса, поэтому этот паттерн не стал использовать.