Цель ДЗ

В этом домашнем задании вы улучшите ваше приложение со списком, созданное в предыдущем модуле.
Вы упростите работу с View внутри ViewHolder с помощью интерфейса LayoutContainer
Научитесь отображать списки в различных видах, использовать декораторы для создания разделителей и добавлять анимации к элементам списка
Научитесь использовать DiffUtil, позволяющий упростить обновление списка
Примените паттерн Delegate Adapters для неоднородных списков.
В качестве дополнительного задания вы реализуете пагинацию в списке, которая позволит вам подгружать элементы списка пачками по мере того, как пользователь скролит список.

Описание задания

1. Создать корневой фрагмент MainFragment, в котором будут отображаться кнопки для навигации на другие экраны. Добавить кнопку навигации на экран списка элементов *выбранной вами темы в предыдущем модуле*. Обработайте нажатие на эту кнопку. Реализацию экрана списка *Person* можно взять из прошлого модуля.
   Далее в задании используется пример с персонами, изменяйте текст, выделенный курсивом, в соответствии со своей темой.

2. Во ViewHolder'ах списка *Person* избавьтесь от методов findViewById, уберите переменные, которые ссылаются на View. Реализуйте интерфейс LayoutContainer ViewHolder'ом для того, чтобы внутри метода bind обращаться к закэшированным вью с помощью идентификаторов. Убедитесь с помощью декомпиляции байткода, что под капотом внутри метода bind не вызывается метод findViewById.
3. Создать несколько видов списков - горизонтальный, сеткой, неровной сеткой. Для этого используйте LinearLayoutManager, GridLayoutManager, StaggeredGridLayoutManager. Реализуйте пример списка по желанию. Каждый список отобразите в отдельном фрагменте. Добавьте навигацию на эти экраны из MainFragment.
4. Создать собственный разделитель с помощью DividerItemDecoration. Для этого создайте xml drawable и установите его в качестве разделителя в теме. Добавьте разделитель в какой-нибудь из списков.
5. Добавьте оступы на списки с помощью кастомного ItemDecoration. Величина отступа должна задаваться в dp и переводиться в px при установке отступа.
6. Измените анимации по умолчанию для добавления/удаления элементов *Person.*
7. Настройте асинхронный DiffUtil при работе с адаптером списка *Person,* избавьтесь от вызовов notifyItem для оповещения адаптера.
8. Примените подход Delegate Adapters к списку *Person*.
9. *Добавьте пагинацию для списка элементов *Person*. Когда пользователь прокручивает список до конца должна подгружаться новая пачка элементов

типа *Person*. Ограничьте максимальный размер списка, после которого новые пачки загружаться больше не будут.

Рекомендации

Используйте репозиторий **learning_materials / android_basic**
Скачайте изменения в репозитории на локальную машину.
Выполните ДЗ в папке **Lists_2.**
Перед выполнением - скопируйте проект из папки **Lists_1,** после копирования проекта сделайте коммит, выполните домашнее задание и сделайте второй коммит (или несколько коммитов). Отправьте коммиты в удаленный репозиторий.

Удостоверьтесь, что списки не тормозят при быстром скролле с использованием настройки разработчика profile gpu rendering.

Для изменения анимаций элементов списка используйте ItemAnimator из репозитория https://github.com/wasabeef/recyclerview-animators

Для упрощения реализации Delegate Adapters используйте библиотеку https://github.com/sockeqwe/AdapterDelegates

Критерии оценки

1. Код оформлен в соответствии с правилами https://kotlinlang.org/docs/reference/coding-conventions.html.
2. Соблюдён принцип инкапсуляции с помощью модификаторов доступа.
3. Классы являются не финальными (open, abstract) только при необходимости.
4. Текстовые строки не являются захардкоженными и используются из ресурсов.
5. Для хранения списка сущностей используется ArrayList.
6. В качестве значений отступов в разделителе используются dp.
7. Внутри метода bind у любого ViewHolder не должно вызываться методов findViewById ни явно, ни под капотом синтетических свойств.
8. Адаптеры зануляются в методе onDestroyView фрагмента, если они были сохранены в поле.
9. При работе с DiffUtil сущности сравниваются внутри метода areItemsTheSame по id. Внутри метода areContentTheSame - по всем полям сущности.
10. Для каждого вида элемента в списке *Person* создан Delegate Adapter. Основной адаптер списка *PersonAdapter* только указывает с какими Delegate Adapter будет работать и не содержит логики создания / связки ViewHolder.

Дополнительные материалы

https://developer.android.com/jetpack - набор библиотек Android Jetpack, предназначенных для решения часто встречающихся задач

https://guides.codepath.com/android/endless-scrolling-with-adapterviews-and-recyclerview - пример реализации пагинации