Федеральное государственное автономное образовательное учреждение высшего образования «Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

**Отчёт по лабораторной работе №4**

**Дисциплина:** Проектирование мобильных приложений

**Тема:** «RecyclerView»

**Вариант №14**

Выполнил студент гр. 3530901/70202

Д.В.Павлов

*(подпись)*

Преподаватель

А.Н.Кузнецов

*(подпись)*

Санкт-Петербург

2019

# Цели работы

1.1. Ознакомиться с принципами работы adapter-based views;

1.2. Получить практические навыки разработки адаптеров для view.

# 2. Задачи работы

**2.1.**  **Знакомство с библиотекой (*unit test*):**

Ознакомиться со *strict mode* библиотеки, проиллюстрировав его работу *unit*-тестом.

* 1. **Знакомство с *RecyclerView*:**

Написать *Android* приложение, которое выводит все записи из *bibtex* файла (записи имеют разный формат) на экран, используя предложенную библиотеку и *RecyclerView*.

* 1. **Бесконечный список:**

Сделать список из предыдущей задачи бесконечным: после последнего элемента все записи повторяются, начиная с первой.

# 3. Выполнение работы

**3.1. Знакомство с библиотекой (*unit test*):**

3.1.1. Тест *strictModeThrowsException()*

Отличие *strict mode* от *normal mode* заключается в том, что в строгом режиме мы применяем ограничение: в памяти нельзя хранить более чем maxValid (в нашем случае равен *20)* записей одновременно. При извлечении начиная от *maxValid+1й* при доступе к полям кидаются исключения. Это ограничение позволит быстрее выявлять ошибки при работе с *RecyclerView* и адаптерами.

Был написан тест. Первые 20 записей извлекаются безошибочно, а при попытке извлечь 21-ю и т. д. записи, ожидается исключение *IllegalStateException*. Код представлен в Листинге 3.1.1.1.

Листинг 3.1.1.1.

@Test  
public void strictModeThrowsException() {  
 BibConfig cfg = database.getCfg();  
 cfg.strict = true;  
  
 BibEntry firstEn = database.getEntry(0);  
 for (int i = 0; i < cfg.maxValid + 10; i++) {  
 if (i < cfg.maxValid - 1) {  
 Assert.*assertNotNull*("Should not throw any exception @" + i, firstEn.getType());  
 } else {  
 try {  
 firstEn.getType();  
 } catch (IllegalStateException ex) {  
 *assertThat*(ex.getMessage(), *is*("This object has already been invalidated. myOrder=1," +  
 "latestOrder=" + String.*valueOf*(i + 2)));  
 }  
 }  
 }  
}

3.1.2. Тест *shuffleFlag()*

Данный флаг перемешивает прочитанные из файла данные случайным образом. Вероятность, что после перемешивания элементы будут располагаться в точности том же порядке, что и ранее, крайне мала, но существует. Она была найдена и, для её уменьшения, перемешивание производилось 100 раз. Также для тестирования был создан файл *references2.bib* (в файле *references.bib* хранится всего одна запись и перемешивать ёё не имеет смысла).

Код соответствующего теста представлен в Листинге 3.1.2.1.

Листинг 3.1.2.1.

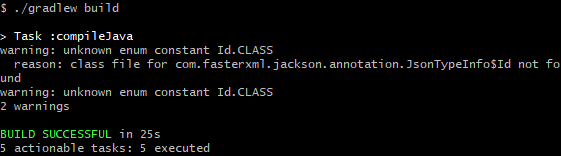
@Test  
public void shuffleFlag() throws IOException {  
 BibConfig cfg = new BibConfig();  
 cfg.shuffle = true;  
   
 int bibEntriesSize;  
 BibDatabase shuffledBibDatabase;  
 List<BibTeXEntry> shuffledEntries;  
   
 try (InputStreamReader streamReader = new InputStreamReader(getClass().getResourceAsStream("/test\_references.bib"))) {  
 shuffledBibDatabase = new BibDatabase(streamReader, cfg);  
 shuffledEntries = shuffledBibDatabase.getEntries();  
 bibEntriesSize = shuffledBibDatabase.getEntriesSize();  
 }  
   
 List<BibTeXEntry> oldEntries = new ArrayList<>(shuffledEntries);  
 double chances = 1 - 1 / (double) factorial(bibEntriesSize);  
 int count = 0;  
 int itNumber = 100;  
   
 for (int i = 0; i < itNumber; i++) {  
 try (InputStreamReader streamReader = new InputStreamReader(getClass().getResourceAsStream("/test\_references.bib"))) {  
 shuffledBibDatabase = new BibDatabase(streamReader, cfg);  
 shuffledEntries = shuffledBibDatabase.getEntries();  
 if (!shuffledEntries.equals(oldEntries)) count++;  
 }  
 }  
 Assert.*assertTrue*((double) (count) / itNumber >= chances);  
}

}

3.1.3. Сборка библиотеки

Используя команду *./gradlew build* был создан *.jar* файл (Листинг 3.1.3.1).

Листинг 3.1.3.1.



Сборка завершилась успешно, т. к. все *unit* тесты выполняются безошибочно.

**3.2. Знакомство с *RecyclerView***

C помощью *RecyclerView* можно отображать прокручивающиеся списки, которые содержат большие объемы данных или могут изменяться. В его моделипроисходит кооперация некоторого количества компонентов для отображения данных. Их воспроизведение в виде списка осуществляется объектами типа *ViewHolder*.

Также у *RecyclerView* имеется адаптер (*RecyclerView.Adapter*), который осуществляет контроль над объектами *ViewHolder*. Он занимается созданием необходимых *ViewHolders* для отображения на экране, плюс нескольких дополнительных, чтобы они сразу визуально были доступны пользователю. Дополнительно мы можем уведомить адаптер об изменении отображаемых элементов, используя *notify()*.

Польза *RecyclerView* заключается в том, что неиспользуемые на данный момент элементы не будут храниться в памяти. Это значит, что, когда происходит прокручивание элементов списка, прокрученные элементы будут удаляться, а новые будут постепенно подгружаться.

Собранная библиотека была подключена как прекомпилированный *jar* файл (Рис.3.2.1, Листинг 3.2.1-2).

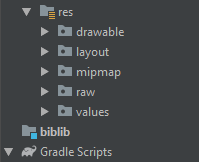
******

Рис.3.2.1.

Листинг 3.2.1.

include ':app', ':biblib'  
rootProject.name='RecView'

Листинг 3.2.2.

implementation project(":biblib")

implementation "androidx.recyclerview:recyclerview:1.1.0-rc01"

В данной задаче требуется отобразить список, в котором хранятся различные элементы. Для каждого формата записи было выбрано несколько полей, чтобы все было кратко и понятно. Результат представлен в Таблице 3.2.1.

Таблица 3.2.1.

|  |  |
| --- | --- |
| **Формат записи** | **Отображаемые поля** |
| article | author, title, journal |
| misc | author, title, volume |
| inproceedings | author, title, pages, year |
| techreport, unpublished | author, title, address |
| book, incollection | author, title, publisher, year |

Пример отображения записей представлен на Рис.3.2.2.

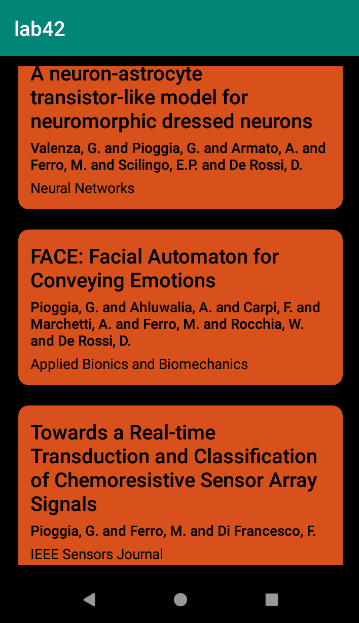
 

Рис.3.2.2.

Решение данной задачи приводится в Листингах 3.2.4-3.2.6.

Листинг 3.2.4. MainActivity

import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import name.ank.lab4.BibDatabase  
import name.ank.lab4.BibEntry  
  
class MainActivity : AppCompatActivity() {  
  
 private lateinit var recyclerView: RecyclerView  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
 recyclerView = findViewById(R.id.*rec\_view*)  
  
 val ourDatabase = BibDatabase(  
 this::class.*java*.getResourceAsStream("/res/raw/publicationsferro.bib")!!.*reader*()  
 )  
   
 var currentEntry = ourDatabase.getEntry(0)  
 val entriesNumber = ourDatabase.*entriesSize* val entriesList = *mutableListOf*<BibEntry>()  
 var counter = 1  
  
 while (currentEntry.*type* != null && entriesNumber > counter) {  
 entriesList.add(currentEntry)  
 currentEntry = ourDatabase.getEntry(counter)  
 counter++  
 }  
   
 recyclerView.*layoutManager* = LinearLayoutManager(this)  
 recyclerView.*adapter* = BibLibAdapter(entriesList)  
 }  
}

Листинг 3.2.5. BibLibViewHolder:

import android.view.View  
import android.widget.TextView  
import androidx.recyclerview.widget.RecyclerView  
import name.ank.lab4.BibEntry  
import name.ank.lab4.Keys  
  
abstract class BibLibViewHolder<T>(view: View) : RecyclerView.ViewHolder(view) {  
 abstract fun bind(entry: T)  
}  
  
class ArticleViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
  
 private val title: TextView = view.findViewById(R.id.*articleTitle*)  
 private val author: TextView = view.findViewById(R.id.*articleAuthor*)  
 private val journal: TextView = view.findViewById(R.id.*articleJournal*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 journal.*text* = entry.getField(Keys.JOURNAL)  
 }  
}  
  
class MiscViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
 private val title: TextView = view.findViewById(R.id.*miscTitle*)  
 private val author: TextView = view.findViewById(R.id.*miscAuthor*)  
 private val volume: TextView = view.findViewById(R.id.*miscVolume*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 volume.*text* = entry.getField(Keys.VOLUME)  
 }  
}  
  
class InproceedingsViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
 private val title: TextView = view.findViewById(R.id.*inproceedingsTitle*)  
 private val author: TextView = view.findViewById(R.id.*inproceedingsAuthor*)  
 private val pages: TextView = view.findViewById(R.id.*inproceedingsPages*)  
 private val year: TextView = view.findViewById(R.id.*inproceedingsYear*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 pages.*text* = entry.getField(Keys.PAGES)  
 year.*text* = entry.getField(Keys.YEAR)  
 }  
}  
  
class TechreportViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
 private val title: TextView = view.findViewById(R.id.*techreportTitle*)  
 private val author: TextView = view.findViewById(R.id.*techreportAuthor*)  
 private val address: TextView = view.findViewById(R.id.*techreportAddress*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 address.*text* = entry.getField(Keys.ADDRESS)  
 }  
}  
  
class UnpublishedViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
 private val title: TextView = view.findViewById(R.id.*unpublishedTitle*)  
 private val author: TextView = view.findViewById(R.id.*unpublishedAuthor*)  
 private val address: TextView = view.findViewById(R.id.*unpublishedAddress*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 address.*text* = entry.getField(Keys.ADDRESS)  
 }  
}  
  
class BookViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
 private val title: TextView = view.findViewById(R.id.*bookTitle*)  
 private val author: TextView = view.findViewById(R.id.*bookAuthor*)  
 private val publisher: TextView = view.findViewById(R.id.*bookPublisher*)  
 private val year: TextView = view.findViewById(R.id.*bookYear*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 publisher.*text* = entry.getField(Keys.PUBLISHER)  
 year.*text* = entry.getField(Keys.YEAR)  
 }  
}  
  
class IncollectionViewHolder(view: View) : BibLibViewHolder<BibEntry>(view) {  
 private val title: TextView = view.findViewById(R.id.*incollectionTitle*)  
 private val author: TextView = view.findViewById(R.id.*incollectionAuthor*)  
 private val publisher: TextView = view.findViewById(R.id.*incollectionPublisher*)  
 private val year: TextView = view.findViewById(R.id.*incollectionYear*)  
  
 override fun bind(entry: BibEntry) {  
 title.*text* = entry.getField(Keys.TITLE)  
 author.*text* = entry.getField(Keys.AUTHOR)  
 publisher.*text* = entry.getField(Keys.PUBLISHER)  
 year.*text* = entry.getField(Keys.YEAR)  
 }  
}

Листинг 3.2.6. BibLibAdapter

import android.view.LayoutInflater  
import android.view.ViewGroup  
import androidx.recyclerview.widget.RecyclerView  
import name.ank.lab4.BibDatabase  
import name.ank.lab4.BibEntry  
import name.ank.lab4.Types  
import java.lang.IllegalArgumentException  
  
class BibLibAdapter( val ourDatabase: BibDatabase) :  
 RecyclerView.Adapter<BibLibViewHolder<\*>>() {  
  
   
 override fun getItemViewType(position: Int): Int {  
 return ourDatabase.getEntry(position % ourDatabase.*entriesSize*).*type*.ordinal  
 }  
  
 override fun getItemCount(): Int {  
 return Integer.*MAX\_VALUE* }  
  
 override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): BibLibViewHolder<\*> {  
 val layoutInflater = LayoutInflater.from(parent.*context*)  
 return when (Types.values()[viewType]) {  
 Types.ARTICLE -> {  
 ArticleViewHolder(  
 layoutInflater.inflate(R.layout.*item\_article*, parent, false)  
 )  
 }  
 Types.MISC -> {  
 MiscViewHolder(  
 layoutInflater.inflate(R.layout.*item\_misc*, parent, false)  
 )  
 }  
 Types.INPROCEEDINGS -> {  
 InproceedingsViewHolder(  
 layoutInflater.inflate(R.layout.*item\_inproceedings*, parent, false)  
 )  
 }  
 Types.TECHREPORT -> {  
 TechreportViewHolder(  
 layoutInflater.inflate(R.layout.*item\_techreport*, parent, false)  
 )  
 }  
 Types.UNPUBLISHED -> {  
 UnpublishedViewHolder(  
 layoutInflater.inflate(R.layout.*item\_unpublished*, parent, false)  
 )  
 }  
 Types.BOOK -> {  
 BookViewHolder(  
 layoutInflater.inflate(R.layout.*item\_book*, parent, false)  
 )  
 }  
 Types.INCOLLECTION -> {  
 IncollectionViewHolder(  
 layoutInflater.inflate(R.layout.*item\_incollection*, parent, false)  
 )  
 }  
 else -> {  
 throw IllegalArgumentException("Illegal type")  
 }  
 }  
 }  
  
 override fun onBindViewHolder(holder: BibLibViewHolder<\*>, position: Int) {  
 val currentEntry = ourDatabase.getEntry(position % ourDatabase.*entriesSize*)  
 when (currentEntry.*type*) {  
 Types.ARTICLE -> (holder as ArticleViewHolder).bind(currentEntry)  
 Types.MISC -> (holder as MiscViewHolder).bind(currentEntry)  
 Types.INPROCEEDINGS -> (holder as InproceedingsViewHolder).bind(currentEntry)  
 Types.TECHREPORT -> (holder as TechreportViewHolder).bind(currentEntry)  
 Types.UNPUBLISHED -> (holder as UnpublishedViewHolder).bind(currentEntry)  
 Types.BOOK -> (holder as BookViewHolder).bind(currentEntry)  
 Types.INCOLLECTION -> (holder as IncollectionViewHolder).bind(currentEntry)  
 else -> throw IllegalArgumentException("Illegal type")  
 }  
 }  
}

**3.3. Бесконечный список**

Для достижения цели в классе *BibLibAdapter* было изменено всего лишь 3 момента:

1. **getItemCount()** теперь возвращает ***Integer.MAX\_VALUE*** (вместо настоящего количества записей);
2. В **getItemViewType()** индекс теперь вычисляется по модулю (*position* ***% entriesNumber***);
3. В **onBindViewHolder()** индекс текущей записи теперь вычисляется по модулю (*position* ***% entries***);

Корректность приложения была проверена путем прокрутки вверх-вниз с ***true/false*** *strict mode* библиотеки. Результат получился успешный.

# 4. Выводы

В данной лабораторной работе требовалось ознакомиться с принципами работы *adapter-based views.* Для этого:

1)Для библиотеки *BibLib* были написаны тесты, проверяющие два используемых флага – *strict (*режим, при котором в памяти может храниться ограниченное количество записей из файла) и *shuffle* (режим, при котором происходит перемешивание извлеченных элементов). После достигнутой корректности тестов удалось собрать объектные файлы для последующего их использования в проектах в качестве библиотеки, что требовалось во 2 пункте данной работы.

2)Было разработано приложение, выводящее все записи из *bibtex* файла на экран, используя *RecyclerView* и библиотеку *BibLib.*

3)Был разработан адаптер *BibLibAdapter* для отображения записей различного родав виде списка. Далее он был модифицирован так, чтобы список прокручивался бесконечно, повторяя себя.