12. ПАНЕЛЬ ИНСТРУМЕНТОВ

В данном разделе для приложения BookDepository будет создано меню, которое станет отображаться на панели инструментов. В этом меню будет присутствовать элемент действия (action item) для добавления новой книги. Также будет обеспечена работа кнопки Up на панели инструментов (рисунок 12.1).

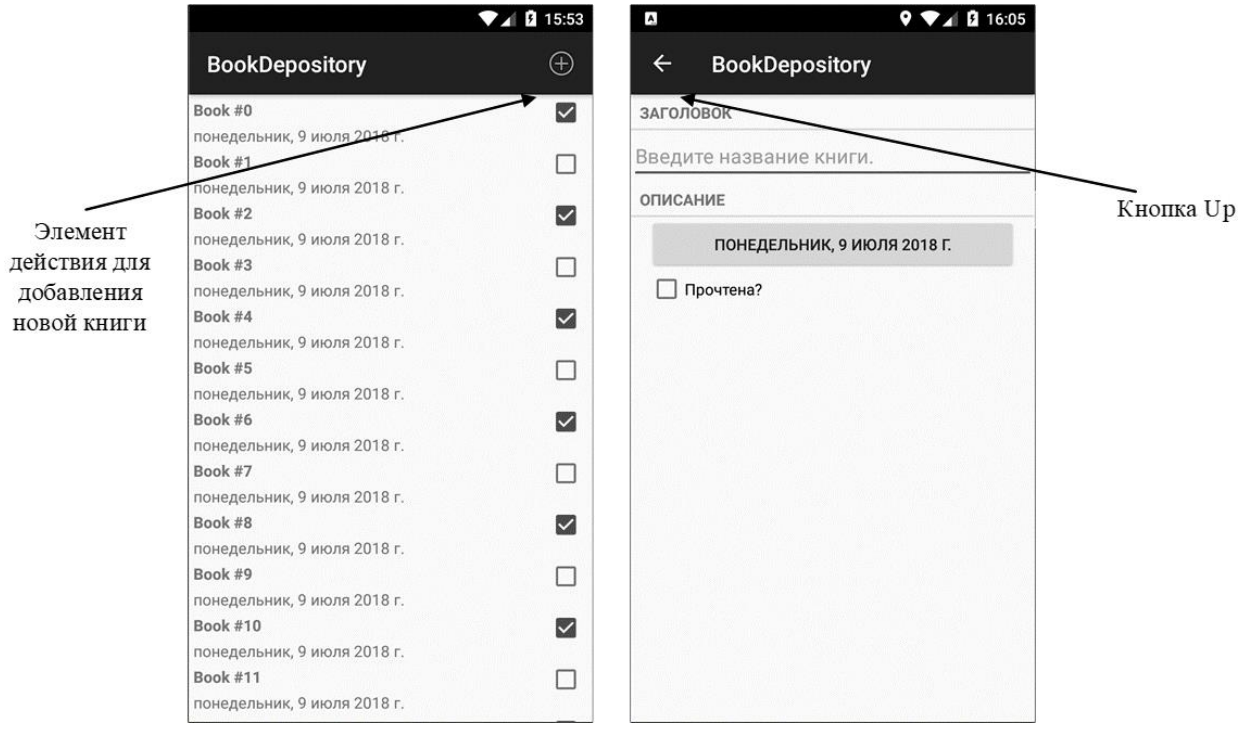

Рисунок 12.1 – Панель инструментов BookDepository

Панель инструментов (toolbar) является ключевым компонентом любого хорошо спроектированного приложения Android. Панель инструментов содержит действия, которые могут выполняться пользователем, и новые средства навигации, а также обеспечивает единство дизайна и фирменного стиля.

Использование библиотеки AppCompat

В предыдущем разделе в приложение BookDepository была добавлена зависимость для библиотеки AppCompat. Полная интеграция с библиотекой AppCompat требует ряда дополнительных шагов. Возможно, некоторые из этих действий уже выполнены.

Для использования библиотеки AppCompat необходимо:

- добавить зависимость AppCompat;
- использовать одну из тем AppCompat;
- проследить за тем, чтобы все активности были субклассами AppCompatActivity.

Обновление темы

Так как зависимость для AppCompat уже добавлена, пора сделать следующий шаг — убедиться в том, что используется одна из тем (themes) AppCompat. Библиотека AppCompat включает три темы:

- `Theme.AppCompat` — темная;
- `Theme.AppCompat.Light` — светлая;
- `Theme.AppCompat.Light.DarkActionBar` — светлая с темной панелью инструментов.

Тема приложения задается на уровне приложения; также существует необязательная возможность назначения темы на уровне активностей в файле AndroidManifest.xml. Откройте файл AndroidManifest.xml и найдите тег application. Следует обратить внимание на атрибут `android:theme`. Он выглядит примерно так, как показано в листинге 12.1.

Листинг 12.1 – Стандартный манифест (AndroidManifest.xml)

```
...
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme" >
...
```

AppTheme определяется в файле res/values/styles.xml. В зависимости от того, как создавался исходный проект, он может содержать несколько версий AppTheme в нескольких файлах styles.xml. Надо убедиться в том, что атрибут parent элемента AppTheme соответствует выделенной части в листинге 12.2.

Листинг 12.2 – Использование темы AppCompat (res/values/styles.xml)

```
<resources>
    <style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
    </style>
</resources>
```

Использование AppCompatActivity

Последним шагом становится преобразование всех активностей BookDepository в субклассы AppCompatActivity. До настоящего момента все активности были субклассами FragmentActivity, что позволяло использовать реализацию фрагментов из библиотеки поддержки.

Класс AppCompatActivity сам является субклассом FragmentActivity. Это означает, что по-прежнему можно использовать поддержку фрагментов в AppCompatActivity, что упрощает необходимые изменения в BookDepository.

Преобразовать SingleFragmentActivity и BookPagerActivity в субклассы AppCompatActivity.

Листинг 12.3 – Преобразование в AppCompatActivity (SingleFragmentActivity.java)

```
public abstract class SingleFragmentActivity extends FragmentActivity
        AppCompatActivity {
    ...
}
```

Листинг 12.4 – Преобразование в AppCompatActivity (BookPagerActivity.java)

```
public class BookPagerActivity extends FragmentActivity
        AppCompatActivity {
    ...
}
```

Запустить BookDepository и убедиться в том, что запуск происходит без сбоев. Приложение должно выглядеть примерно так, как показано на рисунке 12.2.

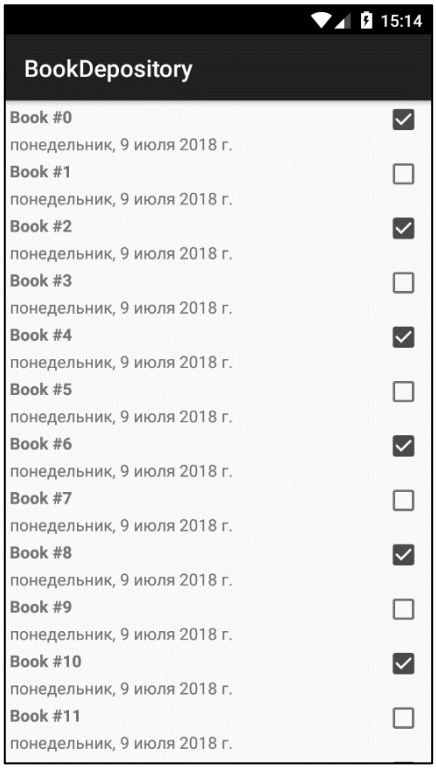

Рисунок 12.2 – Новая панель инструментов

Теперь, когда в приложении BookDepository используется панель инструментов AppCompat, можно добавлять действия на панель инструментов.

Меню

Правая верхняя часть панели инструментов зарезервирована для меню. Меню состоит из *элементов действий* (иногда также называемых *элементами меню*), выполняющих действия на текущем экране или в приложении в целом. Далее будет добавлен элемент действия, при помощи которого пользователь сможет создать описание новой книги.

Для работы меню потребуется несколько строковых ресурсов. Добавить их в файл strings.xml (листинг 12.5). Пока эти строки выглядят довольно загадочно, но лучше решить эту проблему сразу. Когда они позднее понадобятся, они уже будут на своем месте.

Листинг 12.5 – Добавление строк для меню (res/values/strings.xml)

```
<resources>
    ...
    <string name="date_picker_title">Дата начала прочтения:</string>
    <string name="new_book">Новая книга</string>
    <string name="show_subtitle">Показать подзаголовок</string>
    <string name="hide_subtitle">Скрыть подзаголовок</string>
    <string name="subtitle_format">%1$d книг</string>
</resources>
```

Определение меню в XML

Меню определяются такими же ресурсами, как и макеты. Создается описание меню в XML и файл помещается в каталог res/menu проекта. Android генерирует идентификатор ресурса для файла меню, который затем используется для заполнения меню в коде.

В окне инструментов Project щёлкнуть правой кнопкой мыши на каталоге res и выбрать команду New→Android resource File. Выбрать тип ресурса Menu, присвойте ресурсу меню имя fragment_book_list и щёлкнуть на кнопке OK. Android Studio сгенерирует файл res/menu/fragment_book_list.xml (рисунок 12.3).

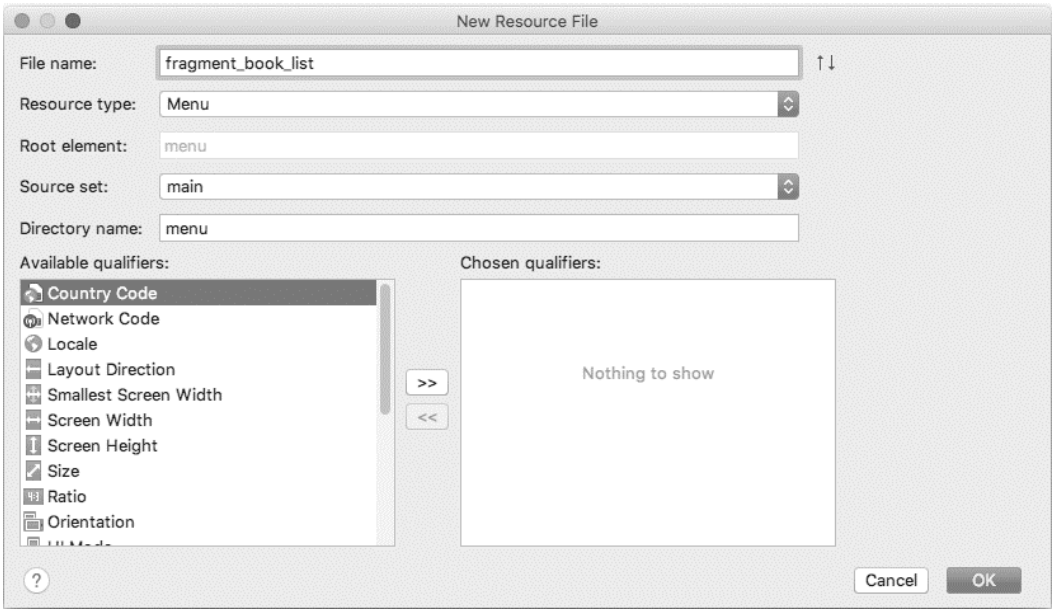

Рисунок 12.3 – Создание файла меню

Добавить в новый файл fragment_book_list.xml элемент item (листинг 12.6).

Листинг 12.6 – Создание ресурса меню BookListFragment (fragment_book_list.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/menu_item_new_book"
        android:icon="@android:drawable/ic_menu_add"
        android:title="@string/new_book"
        app:showAsAction="ifRoom|withText"/>
</menu>
```

Атрибут showAsAction устанавливает, должна ли команда меню отображаться на самой панели инструментов или в *дополнительном меню* (overflow menu). В данном случае два значения объединиены, ifRoom и withText, чтобы при наличии свободного места на панели инструментов отображался значок и текст команды. Если на панели не хватает места для текста, то отображается только значок. Если места нет ни для того, ни для другого, команда перемещается в дополнительное меню. Дополнительное меню вызывается значком в виде трех точек в правой части панели инструментов.

Android Asset Studio

В атрибуте `android:icon` значение `@android:drawable/ic_menu_add` ссылается на *системный значок* (system icon). Системные значки находятся на устройстве, а не в ресурсах проекта.

В прототипе приложения ссылки на системные значки работают нормально. Однако в приложении, готовом к выпуску, лучше быть уверенным в том, что именно пользователь увидит на экране. Системные значки могут сильно различаться между устройствами и версиями ОС, а на некоторых устройствах системные значки могут не соответствовать дизайну приложения.

Одно из возможных решений — найти системные значки, соответствующие потребностям приложения, и скопировать их прямо в графические ресурсы проекта.

Системные значки находятся в каталоге Android SDK. Для получения местонахождения SDK можно открыть окно Project Structure и выбрать категорию SDK Location. В каталоге SDK можно найти разные ресурсы Android, включая ic_menu_add. Эти ресурсы находятся в каталоге /platforms/android-26/data/res, где 26 — уровень API Android-версии.

Самый простой вариант — использовать программу **Android Asset Studio**, включенную в Android Studio. Asset Studio позволяет создать и настроить изображение для использования на панели инструментов.

Щёлкнуть правой кнопкой мыши на каталоге drawable в окне инструментов Project и выбрать команду New→Image Asset. На экране появляется Asset Studio (рисунок 12.4).

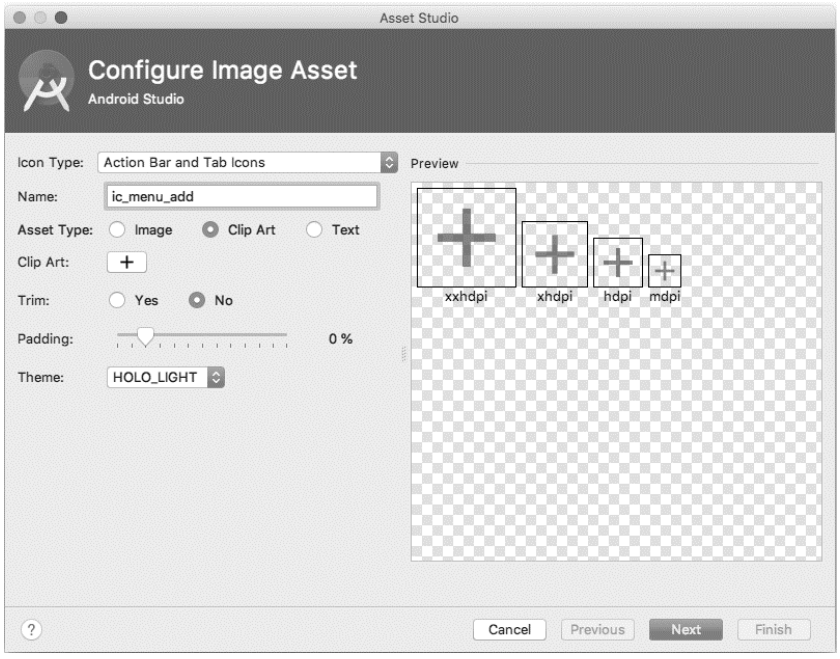

Рисунок 12.4 – Asset Studio

В Asset Studio можно генерировать значки нескольких разных типов. Выбрать в поле Asset Type: вариант Action Bar and Tab Icons. Затем в группе Foreground выбрать переключатель Clipart и нажать кнопку Choose, чтобы выбрать графическую заготовку.

В открывшемся окне выбрать изображение, напоминающее знак + (рисунок 12.5).

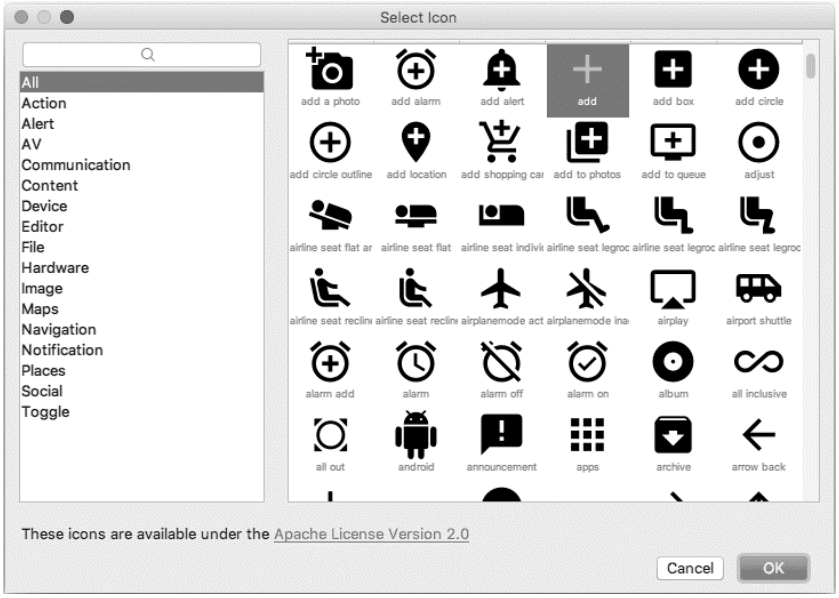

Рисунок 12.5 – Галерея графических заготовок

Наконец, ввести имя ic_menu_add и нажать кнопку Next (рисунок 12.6).

Asset Studio предлагает выбрать модуль и каталог для добавления изображения. Оставить значения по умолчанию, чтобы добавить изображение в модуль app. В окне также выводится план работы, которую выполнит Asset Studio. Значки mdpi, hdpi, xhdpi и xxhdpi будут созданы автоматически.

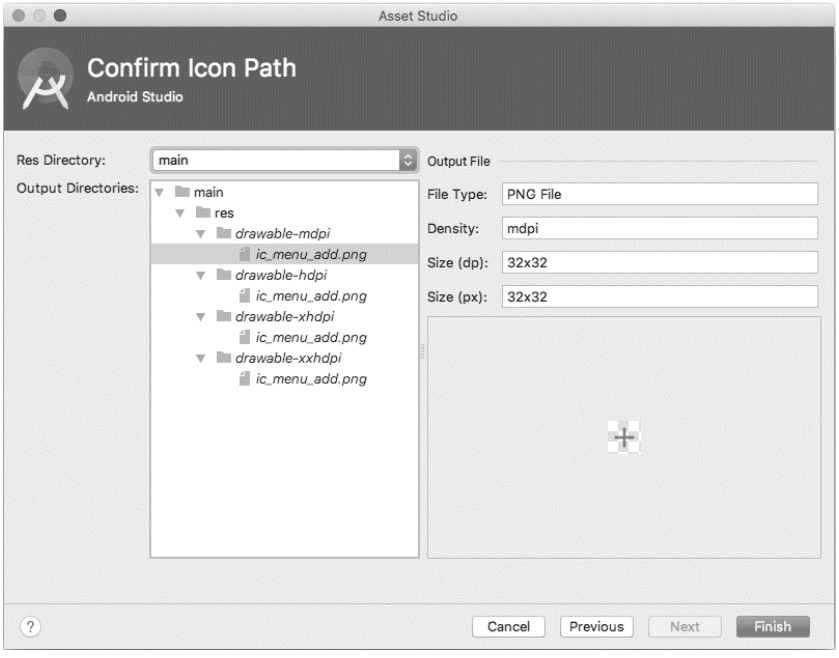

Рисунок 12.6 – Файлы, сгенерированные в Asset Studio

Щёлкнуть на кнопке **Finish**, чтобы сгенерировать изображения.

Затем в файле макета изменить атрибут icon и включить в него ссылку на новый ресурс из данного проекта.

Листинг 12.7 – Ссылка на локальный ресурс (menu/fragment_book_list.xml)

```
<item
    android:id="@+id/menu_item_new_book"
    android:icon="@android:drawable/ic_menu_add"
    android:icon="@drawable/ic_menu_add"
    android:title="@string/new_book"
    app:showAsAction="ifRoom|withText"/>
```

Создание меню

Для управления меню в коде используются методы обратного вызова класса Activity. Когда возникает необходимость в меню, Android вызывает метод Activity с именем `onCreateOptionsMenu(Menu)`.

Однако архитектура приложения требует, чтобы реализация находилась в фрагменте, а не в активности. Класс Fragment содержит собственный набор методов обратного вызова для командных меню, которые будут реализованы в BookListFragment. Для создания меню и обработки выбранных команд используются следующие методы:

```
public void onCreateOptionsMenu(Menu menu,
        MenuInflater inflater)
public boolean onOptionsItemSelected(MenuItem item)
```

В файле BookListFragment.java переопределить метод `onCreateOptionsMenu(Menu, MenuInflater)` так, чтобы он заполнял меню, определенное в файле fragment_book_list.xml.

Листинг 12.8 – Заполнение ресурса меню (BookListFragment.java)

```
@Override
public void onResume() {
    super.onResume();
    updateUI();
}
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater);
    inflater.inflate(R.menu.fragment_book_list, menu);
}
```

В этом методе вызывается метод MenuInflater.inflate(int, Menu) и передается идентификатор ресурса файла меню. Вызов заполняет экземпляр Menu командами, определенными в файле.

FragmentManager отвечает за вызов `Fragment.onCreateOptionsMenu(Menu, MenuInflater)` при получении активностью обратного вызова `onCreateOptionsMenu(…)` от ОС. Поэтому необходимо явно указать FragmentManager, что фрагмент должен получить вызов `onCreateOptionsMenu(…)`. Для этого вызывается следующий метод:

```
public void setHasOptionsMenu(boolean hasMenu)
```

В методе `BookListFragment.onCreate(…)` следует сообщить FragmentManager, что экземпляр BookListFragment должен получать обратные вызовы командного меню.

Листинг 12.9 – Получение обратных вызовов (BookListFragment.java)

```
...
private RecyclerView mBookRecyclerView;
private BookAdapter mAdapter;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setHasOptionsMenu(true);
}

@Override
public View onCreateView(LayoutInflater inflater, ...){
    ...
}
```

В приложении BookDepository появляется командное меню (рисунок 12.7). У большинства телефонов в книжной ориентации хватает места только для значка. Текст команды открывается долгим нажатием на значке на панели инструментов (рисунок 12.8).

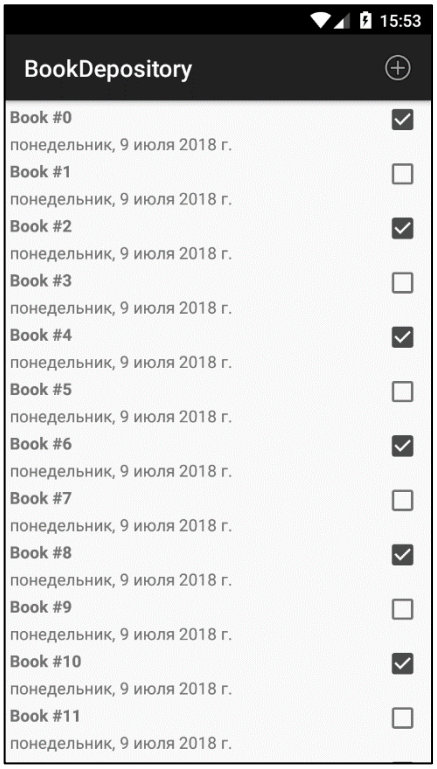

Рисунок 12.7 – Значок команды меню на панели инструментов

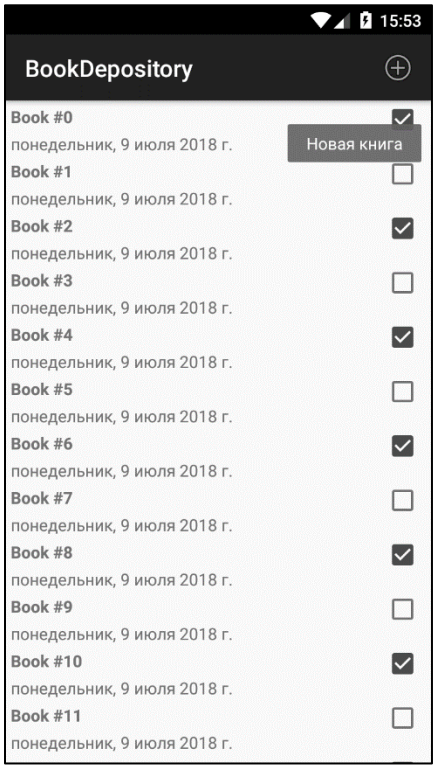

Рисунок 12.8 – Долгое нажатие на значке на панели инструментов выводит текст команды

В альбомной ориентации на панели инструментов хватает места как для значка, так и для текста (рисунок 12.9).

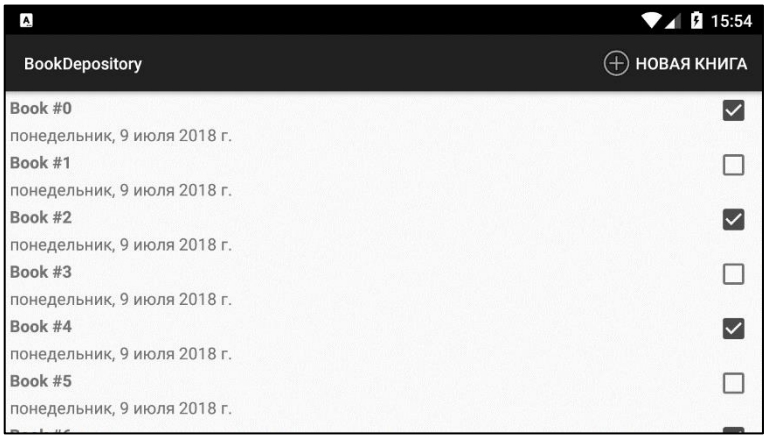

Рисунок 12.9 – Значок и текст на панели инструментов

Реакция на выбор команд

Чтобы отреагировать на выбор пользователем команды *New Book*, понадобится механизм добавления нового объекта Book в список. Включить в файл BookLab.java следующий метод.

Листинг 12.10 – Добавление нового объекта Book (BookLab.java)

```
...
public void addBook(Book b) {
    mBooks.add(b);
}
public List<Book> getBooks() {
    return mBooks;
}
...
```

Теперь, когда есть возможность вводить описания книг самостоятельно, программное генерирование 100 объектов становится лишним. В файле BookLab.java удалить код, генерирующий эти книги.

Листинг 12.11 – Долой случайные книги! (BookLab.java)

```
private BookLab(Context context) {
    mBooks = new ArrayList<>();
    for (int i = 0; i < 100; i++) {
        Book book = new Book();
        book.setTitle("Book #" + i);
        book.setReaded(i % 2 == 0);
        mBooks.add(book);
    }
}
```

Когда пользователь выбирает команду в командном меню, фрагмент получает обратный вызов метода `onOptionsItemSelected(MenuItem)`. Этот метод получает экземпляр MenuItem, описывающий выбор пользователя.

И хотя сознанное меню состоит всего из одной команды, в реальных меню их обычно больше. Чтобы определить, какая команда меню была выбрана, надо проверить идентификатор команды меню и отреагировать соответствующим образом. Этот идентификатор соответствует идентификатору, назначенному команде в файле меню.

В файле BookListFragment.java реализуйте метод `onOptionsItemSelected(MenuItem)`, реагирующий на выбор команды меню. Реализация создает новый объект Book, добавляет его в BookLab и запускает экземпляр BookPagerActivity для редактирования нового объекта Book.

Листинг 12.12 – Реакция на выбор команды меню (BookListFragment.java)

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    ...
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
```

```
    switch (item.getItemId()) {
        case R.id.menu_item_new_book:
            Book book = new Book();
            BookLab.get(getActivity()).addBook(book);
            Intent intent = BookPagerActivity
                .newIntent(getActivity(), book.getId());
            startActivity(intent);
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Метод возвращает логическое значение. После того как команда меню будет обработана, он вернет true; тем самым сообщая, что дальнейшая обработка не нужна. Секция default вызывает реализацию суперкласса, если идентификатор команды не известен в данной реализации.

Запустить приложение BookDepository и опробовать новую команду. Добавить несколько книг и отредактировать их.

Иерархическая навигация

До настоящего момента приложение BookDepository использует кнопку Back для навигации по приложению. Кнопка Back возвращает приложение к предыдущему состоянию. С другой стороны, *иерархическая навигация* осуществляет перемещение по иерархии приложения.

В иерархической навигации пользователь переходит на один уровень «наверх» к родителю текущей активности при помощи кнопки Up в левой части панели инструментов. Чтобы включить иерархическую навигацию в приложение BookDepository, добавить атрибут parentActivityName в файл AndroidManifest.xml.

Листинг 12.13 – Включение кнопки Up (AndroidManifest.xml)

```
...
<activity
    android:name=".BookPagerActivity"
    android:label="@string/app_name"
    android:parentActivityName=".BookListActivity">
</activity>
...
```

Запустить приложение BookDepository и добавить новую книгу. Обратить внимание на появление кнопки Up (рисунок 12.10). Нажатие кнопки Up переводит приложение на один уровень вверх в иерархии BookDepository к активности BookListActivity.

Рисунок 12.10 – Кнопка Up в BookPagerActivity

Альтернативная команда меню

Далее будет добавлена команда меню, скрывающая и отображающая подзаголовок в панели инструментов BookListActivity.

В файле res/menu/fragment_book_list.xml добавить элемент действия *Show Subtitle*, который будет отображаться на панели инструментов при наличии свободного места.

Листинг 12.14 – Добавление элемента действия Show Subtitle (res/menu/fragment_book_list.xml)

```
...
    <item
    android:id="@+id/menu_item_new_book"
    ...
    app:showAsAction="ifRoom|withText"/>
    <item
        android:id="@+id/menu_item_show_subtitle"
        android:title="@string/show_subtitle"
        app:showAsAction="ifRoom"/>
</menu>
```

Команда отображает подзаголовок с количеством книг в BookDepository. Создать новый метод `updateSubtitle()`, который будет задавать подзаголовок панели инструментов.

Листинг 12.15 – Назначение подзаголовка панели инструментов (BookListFragment.java)

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    ...
}
private void updateSubtitle() {
    BookLab bookLab = BookLab.get(getActivity());
    int bookCount = bookLab.getBooks().size();
    String subtitle = getString(R.string.subtitle_format, bookCount);
    AppCompatActivity activity = (AppCompatActivity) getActivity();
    activity.getSupportActionBar().setSubtitle(subtitle);
}
```

Метод updateSubtitle генерирует строку подзаголовка при помощи метода `getString(int resId, Object… formatArgs)`, который получает значения, подставляемые на место заполнителей в строковом ресурсе.

Затем активность, являющаяся хостом для BookListFragment, преобразуется в AppCompatActivity. BookDepository использует библиотеку AppCompat, поэтому все активности должны быть субклассами AppCompatActivity, чтобы панель инструментов была доступной для приложения. По историческим причинам панель инструментов во многих местах библиотеки AppCompat называется «панелью действий».

Итак, теперь метод updateSubtitle определен, и его можно вызвать при нажатии нового элемента действий.

Листинг 12.16 – Обработка элемента действия Show Subtitle (BookListFragment.java)

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_new_book:
        ...
        case R.id.menu_item_show_subtitle:
            updateSubtitle();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Запустить BookDepository, нажать элемент *Показать подзаголовок* и убедиться в том, что в подзаголовке отображается количество книг.

Переключение текста команды

Теперь подзаголовок отображается, но текст команды меню остался неизменным: *Показать подзаголовок*. Было бы лучше, если бы текст команды и функциональность команды меню изменялись в зависимости от текущего состояния подзаголовка.

При вызове `onOptionsItemSelected(…)` в параметре передается объект MenuItem для элемента действия, нажатого пользователем. Текст элемента *Показать подзаголовок* можно было бы обновить в этом методе, но изменения будут потеряны при повороте устройства и повторном создании панели инструментов.

Другое, более правильное решение — обновить объект MenuItem для *Show Subtitle* в `onCreateOptionsMenu(…)` и инициировать повторное создание панели инструментов при нажатии на элементе действия. Это позволит заново использовать код обновления элемента действия как при выборе элемента действия пользователем, так и при повторном создании панели инструментов.

Сначала добавить переменную для хранения признака видимости подзаголовка.

Листинг 12.17 – Хранение признака видимости подзаголовка (BookListFragment.java)

```
public class BookListFragment extends Fragment {
    private RecyclerView mBookRecyclerView;
    private BookAdapter mAdapter;
    private boolean mSubtitleVisible;
    ...
```

Затем изменить подзаголовок в `onCreateOptionsMenu(…)` и инициировать повторное создание элементов действий при нажатии элемента действия *Показать подзаголовок*.

Листинг 12.18 – Обновление MenuItem (BookListFragment.java)

```
@Override
public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
    super.onCreateOptionsMenu(menu, inflater);
    inflater.inflate(R.menu.fragment_book_list, menu);

    MenuItem subtitleItem = menu.findItem(R.id.menu_item_show_subtitle);
    if (mSubtitleVisible) {
        subtitleItem.setTitle(R.string.hide_subtitle);
    } else {
        subtitleItem.setTitle(R.string.show_subtitle);
    }
}
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.menu_item_new_book:
            ...
        case R.id.menu_item_show_subtitle:
            mSubtitleVisible = !mSubtitleVisible;
            getActivity().invalidateOptionsMenu();
            updateSubtitle();
            return true;
        default:
            return super.onOptionsItemSelected(item);
    }
}
```

Наконец, проверить переменную mSubtitleVisible при отображении или сокрытии подзаголовка панели инструментов.

Листинг 12.19 – Отображение или сокрытие подзаголовка (BookListFragment.java)

```
private void updateSubtitle() {
```

```
    BookLab bookLab = BookLab.get(getActivity());
    int bookCount = bookLab.getBooks().size();
    String subtitle = getString(R.string.subtitle_format, bookCount);
    if (!mSubtitleVisible) {
        subtitle = null;
    }
    AppCompatActivity activity = (AppCompatActivity) getActivity();
    activity.getSupportActionBar().setSubtitle(subtitle);
}
```

Запустить приложение BookDepository и убедиться в том, что подзаголовок успешно скрывается и отображается. Следует обратить внимание на изменение текста команды в зависимости от состояния подзаголовка.

Однако в приложении присутствуют две проблемы. Во-первых, при создании новой книги и последующем возвращении к BookListActivity кнопкой Back содержимое подзаголовка не соответствует новому количеству книг. Во-вторых, если отобразить подзаголовок, а потом повернуть устройство, подзаголовок исчезнет.

Начать следует с первой проблемы. Она решается обновлением текста подзаголовка при возвращении к BookListActivity. Инициировать вызов updateSubtitle в onResume. Метод updateUI уже вызывается в onResume и onCreate; добавить вызов updateSubtitle в метод updateUI.

Листинг 12.20 – Вывод обновленного состояния (BookListFragment.java)

```
private void updateUI() {
    BookLab bookLab = BookLab.get(getActivity());
    List<Book> books = bookLab.getBooks();
    ...
    updateSubtitle();
}
```

Запустить BookDepository, отобразить подзаголовок, создайть новую книгу и нажать на устройстве кнопку Back, чтобы вернуться к BookListActivity. На этот раз на панели инструментов будет отображаться правильное количество.

Повторите эти действия, но вместо кнопки Back можно воспользоваться кнопкой Up. Подзаголовок снова становится невидимым. Это происходит, потому что у реализации иерархической навигации в Android имеется неприятный побочный эффект: активность, к которой осуществляется переход кнопкой Up, полностью создается заново, с нуля.

Возможное решение — переопределение механизма перехода. В BookDepository можно вызвать метод `finish()` для `BookPagerActivity`,

чтобы вернуться к предыдущей активности. Реализовать этот механизм САМОСТОЯТЕЛЬНО, для этого в файле BookPagerActivity.java переопределить метод `onOptionsItemSelected(MenuItem)`, реагирующий на выбор команды меню Up. После этого в подзаголовке всегда будет отображаться правильное количество книг.

Для решения проблемы с поворотом следует сохранить переменную экземпляра mSubtitleVisible между поворотами при помощи механизма сохранения состояния экземпляров.

Листинг 12.21 – Сохранение признака видимости подзаголовка (BookListFragment.java)

```
public class BookListFragment extends Fragment {
    private static final String SAVED_SUBTITLE_VISIBLE = "subtitle";
    ...
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
                container, Bundle savedInstanceState) {
        ...
        if (savedInstanceState != null) {
            mSubtitleVisible =
                savedInstanceState.getBoolean(SAVED_SUBTITLE_VISIBLE);
        }
        updateUI();
        return view;
    }
    @Override
    public void onResume() {
        ...
    }
    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);
        outState.putBoolean(SAVED_SUBTITLE_VISIBLE, mSubtitleVisible);
    }
}
```

Запустить приложение BookDepository. Отобразите подзаголовок, поверните устройство. Подзаголовок должен появиться в воссозданном представлении, как и ожидалось.

Самостоятельные задания.

Задание 1. Удаление книг

Приложение BookDepository дает возможность создать новые книги, но не предоставляет средств для их удаления из списка. В этом задании

добавить в BookFragment новый элемент действия для удаления текущей книги. После того как пользователь нажимает элемент удаления, необходимо вернуть его к предыдущей активности вызовом метода `finish()` для активности-хоста BookFragment.

Задание 2. Множественное число в строках

Если список содержит одну книгу, подзаголовок «1 книг» становится грамматически неправильным. В этом задании необходимо исправить текст подзаголовка.

Можно создать несколько разных строк и выбрать нужную в коде, но такое решение быстро разваливается при попытке локализовать приложение для разных языков. Лучше использовать строковые ресурсы (иногда называемые *количественными строками*) во множественном числе.

Сначала определить в файле strings.xml элемент `plurals`:

```
<plurals name="subtitle_plural">
   <item quantity="one">%1$d книга</item>
   <item quantity="two-four">%1$d книги</item>
   <item quantity="other">%1$d книг</item>
</plurals>
```

Затем использовать метод `getQuantityString(…)` для правильного формирования множественного числа строки:

```
int bookSize = bookLab.getBooks().size();
String subtitle = getResources()
      .getQuantityString(R.plurals.subtitle_plural, bookSize, bookSize);
```

Задание 3. Пустое представление для списка

В настоящее время при запуске BookDepository отображает пустой виджет RecyclerView — большую черную пустоту. Пользователям необходимо предоставить что-то для взаимодействия при отсутствии элементов в списке. В этом задании необходимо в пустом представлении выводить сообщение (например, «Список пуст»).