11. ДИАЛОГОВЫЕ ОКНА

В данном разделе будет добавлено диалоговое окно, в котором пользователь может изменить дату начала прочтения книги. Диалоговые окна требуют от пользователя внимания и ввода данных. Обычно они используются для принятия решений или отображения важной информации.

При нажатии кнопки даты в BookFragment будет открываться диалоговое окно, показанное на рисунке 11.1.

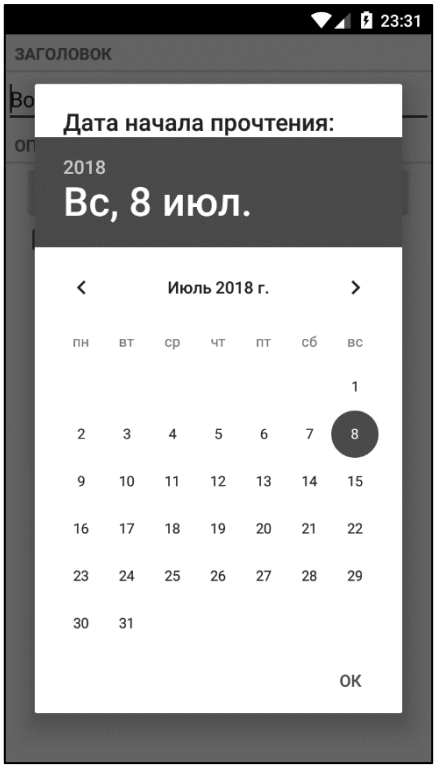

Рисунок 11.1 – Диалоговое окно для выбора даты

Диалоговое окно на рисунке 11.1 является экземпляром **AlertDialog** — субкласса **Dialog**. Именно этот многоцелевой субкласс Dialog будет чаще всего использован в программах.

Начиная с версии Lollipop, диалоговые окна прошли визуальную переработку. AlertDialog в Lollipop и выше автоматически используют новый стиль. В более ранних версиях Android окна AlertDialog возвращаются к старому стилю. Для того чтобы диалоговые окна всегда отображались в новом стиле независимо от версии Android на устройстве пользователя, необходимо использовать библиотеку **AppCompat**.

AppCompat — библиотека совместимости, разработанная компанией Google, которая реализует некоторые возможности новых версий Android на старых устройствах. В приложении библиотека AppCompat будет использована для создания стабильного оформления диалоговых окон во всех поддерживаемых версиях Android.

Библиотека AppCompat

Чтобы использовать библиотеку AppCompat, сначала необходимо добавить ее в число зависимостей. Возможно, она там уже есть; это зависит от того, как создавался проект.

Открыть окно Project Structure (File→Project Structure...), выбрать модуль app и щёлкнуть на вкладке Dependencies. Если библиотека AppCompat не входит в список, добавить ее — щёлкнуть на кнопке + и выбрать зависимость appcompat-v7 в списке (рисунок 11.2).

Рисунок 11.2 – Выбор зависимости AppCompat

Библиотека AppCompat содержит собственный класс AlertDialog, который будет использоваться в приложении. Эта версия AlertDialog очень похожа на ту, которая включена в ОС Android. Чтобы использовать именно ту версию, которая нужна, необходимо импортировать правильную версию AlertDialog. В приложении будет использоваться `android.support.v7.app.AlertDialog`.

Использование DialogFragment

Создание DialogFragment

При использовании объекта AlertDialog обычно удобно упаковать его в экземпляр DialogFragment — субкласса Fragment. Вообще говоря, экземпляр AlertDialog может отображаться и без DialogFragment, но Android так поступать не рекомендует. Управление диалоговым окном из FragmentManager открывает больше возможностей для его отображения.

Кроме того, «минимальный» экземпляр AlertDialog исчезнет при повороте устройства. С другой стороны, если экземпляр AlertDialog упакован во фрагмент, после поворота диалоговое окно будет создано заново и появится на экране.

Для приложения BookDepository будет создан субкласс DialogFragment с именем **DatePickerFragment**. В коде DatePickerFragment создается и

настраивается экземпляр AlertDialog, отображающий виджет DatePicker. В качестве хоста DatePickerFragment используется экземпляр BookPagerActivity.

На рисунке 11.3 изображена схема этих отношений.

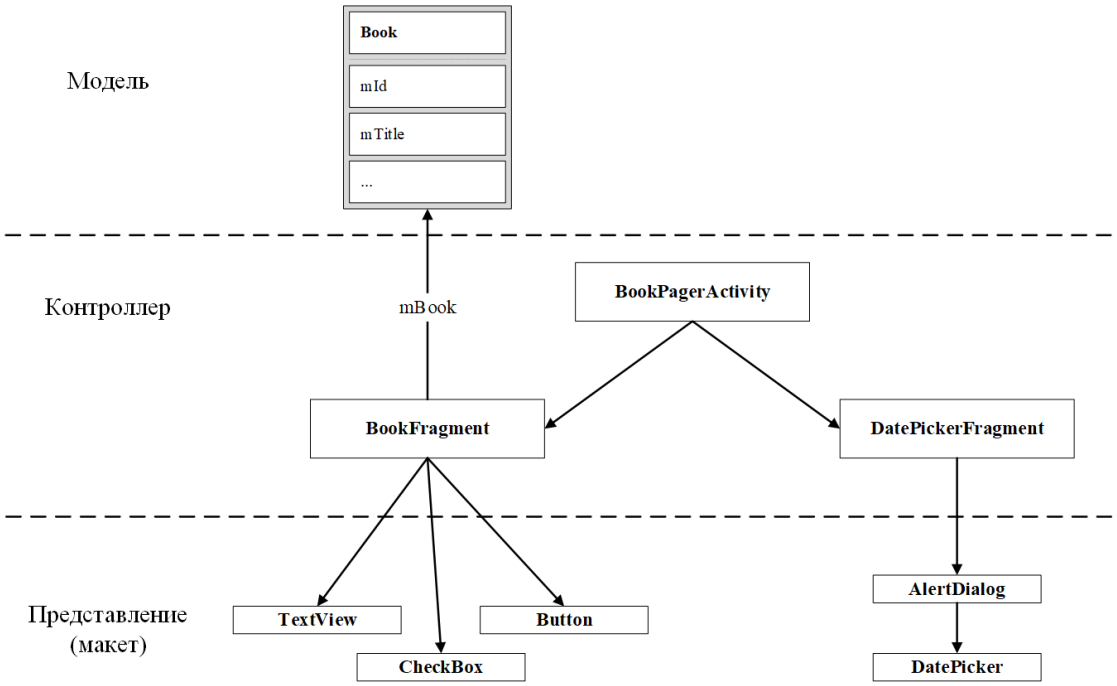

Рисунок 11.3 – Диаграмма объектов для двух фрагментов с хостом BookPagerActivity

В этом разделе предстоит реализовать следующие задачи:

- создать класс DatePickerFragment;
- построить AlertDialog;
- вывести диалоговое окно на экран с использованием FragmentManager.

Перед началом работы, добавить строковый ресурс (листинг 11.1).

Листинг 11.1 – Добавление строки заголовка диалогового окна (values/strings.xml)

```
<resources>
    ...
    <string name="book_readed_label">Прочтена?</string>
    <string name="date_picker_title">Дата начала прочтения:</string>
</resources>
```

Создать новый класс с именем **DatePickerFragment** и назначить его суперклассом **DialogFragment**. Обязательно выбрать версию DialogFragment из библиотеки поддержки: `android.support.v4.app.DialogFragment`.

Класс DialogFragment содержит следующий метод:

```
public Dialog onCreateDialog(Bundle savedInstanceState)
```

Экземпляр FragmentManager активности-хоста вызывает этот метод в процессе вывода DialogFragment на экран.

Добавить в файл DatePickerFragment.java реализацию `onCreateDialog(…)`, которая создает AlertDialog с заголовком и одной кнопкой OK. (Виджет DatePicker будет добавлен позднее.)

Проследите за тем, чтобы импортировалась версия AlertDialog из AppCompat: `android.support.v7.app.AlertDialog`.

Листинг 11.2 – Создание DialogFragment (DatePickerFragment.java)

```
public class DatePickerFragment extends DialogFragment {
   @Override
   public Dialog onCreateDialog(Bundle savedInstanceState) {
      return new AlertDialog.Builder(getActivity())
      .setTitle(R.string.date_picker_title)
      .setPositiveButton(android.R.string.ok, null)
      .create();
   }
}
```

В этой реализации используется класс AlertDialog.Builder, предоставляющий динамичный интерфейс для конструирования экземпляров AlertDialog.

Сначала передается объект Context конструктору AlertDialog.Builder, который возвращает экземпляр AlertDialog.Builder.

Затем вызываются два метода AlertDialog.Builder для настройки диалогового окна:

```
public AlertDialog.Builder setTitle(int titleId)
public AlertDialog.Builder setPositiveButton(int textId,
     DialogInterface.OnClickListener listener)
```

Метод `setPositiveButton(…)` получает строковый ресурс и объект, реализующий DialogInterface.OnClickListener. В листинге 11.2 передается константа Android для кнопки OK и null вместо слушателя. Слушатель будет реализован позднее.

Положительная кнопка (Positive) нажимается пользователем для подтверждения информации в диалоговом окне. В AlertDialog также можно добавить еще две кнопки: отрицательную (Negative) и нейтральную (Neutral). Эти обозначения определяют позицию кнопок в диалоговом окне (если их несколько).

Построение диалогового окна завершается вызовом `AlertDialog.Builder.create()`, который возвращает настроенный экземпляр AlertDialog.

Этим возможности AlertDialog и AlertDialog.Builder не исчерпываются; подробности достаточно хорошо изложены в документации разработчика.

Отображение DialogFragment

Как и все фрагменты, экземпляры DialogFragment находятся под управлением экземпляра FragmentManager активности-хоста.

Для добавления экземпляра DialogFragment в FragmentManager и вывода его на экран используются следующие методы экземпляра фрагмента:

`public void show(FragmentManager manager, String tag)`

`public void show(FragmentTransaction transaction, String tag)`

Строковый параметр однозначно идентифицирует DialogFragment в списке FragmentManager. Выбор версии (с FragmentManager или FragmentTransaction) зависит только от разработчика: если передать FragmentTransaction, за создание и закрепление транзакции отвечает разработчик. При передаче FragmentManager транзакция автоматически создается и закрепляется для разработчика.

Добавить в BookFragment константу для метки DatePickerFragment. Затем в методе `onCreateView(…)` удалить код, блокирующий кнопку даты, и назначить слушателя `View.OnClickListener`, который отображает DatePickerFragment при нажатии кнопки даты.

Листинг 11.3 – Отображение DialogFragment (BookFragment.java)

```
public class BookFragment extends Fragment {
   private static final String ARG_BOOK_ID = "book_id";
   private static final String DIALOG_DATE = "DialogDate";
   ...
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup
               container, Bundle savedInstanceState) {
      ...
      mDateButton.setEnabled(false);
      mDateButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
              FragmentManager manager = getFragmentManager();
              DatePickerFragment dialog = new DatePickerFragment();
              dialog.show(manager, DIALOG_DATE);
         }
      });
      mReadedCheckBox = (CheckBox) v.findViewById(R.id.book_readed);
      ...
      return v;
```

```
}
...
```

Запустить приложение BookDepository и нажать кнопку даты, чтобы диалоговое окно появилось на экране (рисунок 11.4).

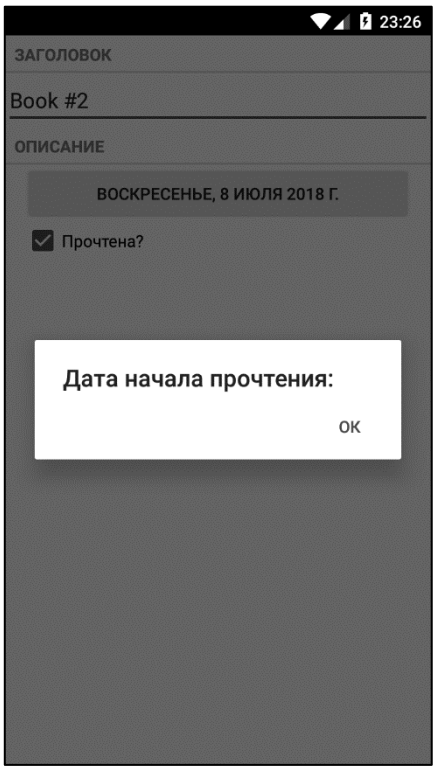

Рисунок 11.4 – AlertDialog с заголовком и кнопкой

Далее будет включен в AlertDialog виджет DatePicker при помощи метода AlertDialog.Builder:

```
public AlertDialog.Builder setView(View view)
```

Метод настраивает диалоговое окно для отображения переданного объекта View между заголовком и кнопкой(-ами).

В окне инструментов Project создать новый файл макета с именем dialog_date.xml и назначить его корневым элементом DatePicker. Макет будет состоять из одного объекта View (DatePicker), который будет заполнен и передан `setView(…)`.

Настройть макет DatePicker так, как показано на рисунке 11.5.

DatePicker

```
xmlns:android="http://schemas.android.com/apk/res/android"
android:id="@+id/dialog_date_date_picker"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:calendarViewShown="false"
```

Рисунок 11.5 – Макет DatePicker (layout/dialog_date.xml)

В методе `DatePickerFragment.onCreateDialog(…)` заполнить представление и назначить его диалоговому окну.

Листинг 11.4 – Включение DatePicker в AlertDialog (DatePickerFragment.java)

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    View v = LayoutInflater.from(getActivity())
        .inflate(R.layout.dialog_date, null);

    return new AlertDialog.Builder(getActivity())
        .setView(v)
        .setTitle(R.string.date_picker_title)
        .setPositiveButton(android.R.string.ok, null)
        .create();
}
```

Запустить приложение BookDepository. Нажать кнопку даты и убедиться в том, что в диалоговом окне теперь отображается DatePicker. На устройствах с версией Lollipop и выше отображается календарный виджет (рисунок 11.6).

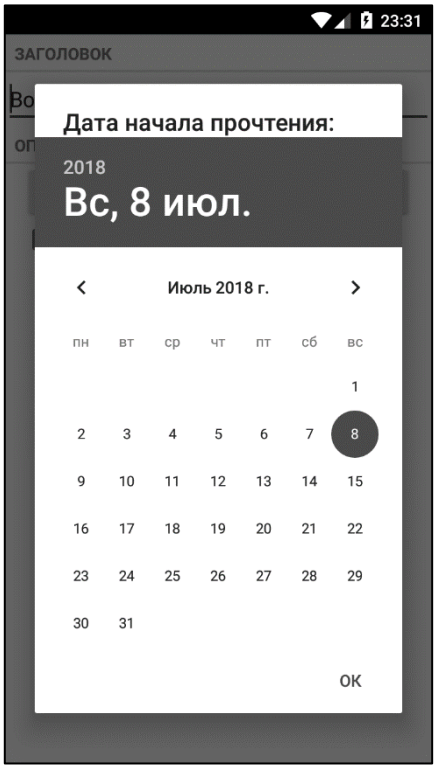

Рисунок 11.6 – DatePicker в Lollipop

Передача данных между фрагментами

В предыдущих разделах осуществлялась передача данных между двумя активностями, между двумя фрагментными активностями, а теперь необходимо передать данные между двумя фрагментами, хостом которых

является одна активность, — BookFragment и DatePickerFragment (рисунок 11.7).

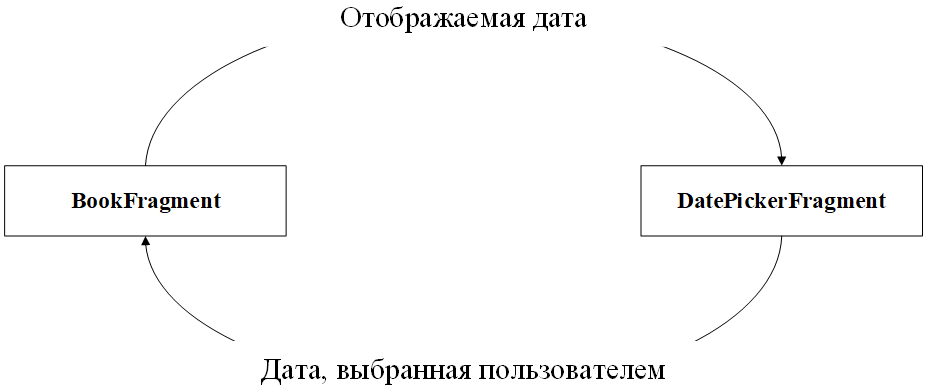

Рисунок 11.7 – Взаимодействие между BookFragment и DatePickerFragment

Чтобы передать дату начала прочтения книги DatePickerFragment, будет создан метод `newInstance(Date)` и объект Date сделан аргументом фрагмента.

Чтобы вернуть новую дату фрагменту BookFragment для обновления уровня модели и его собственного представления, необходимо упаковать ее как дополнение объекта Intent и передать этот объект Intent в вызове `BookFragment.onActivityResult(…)` (рисунок 11.8).

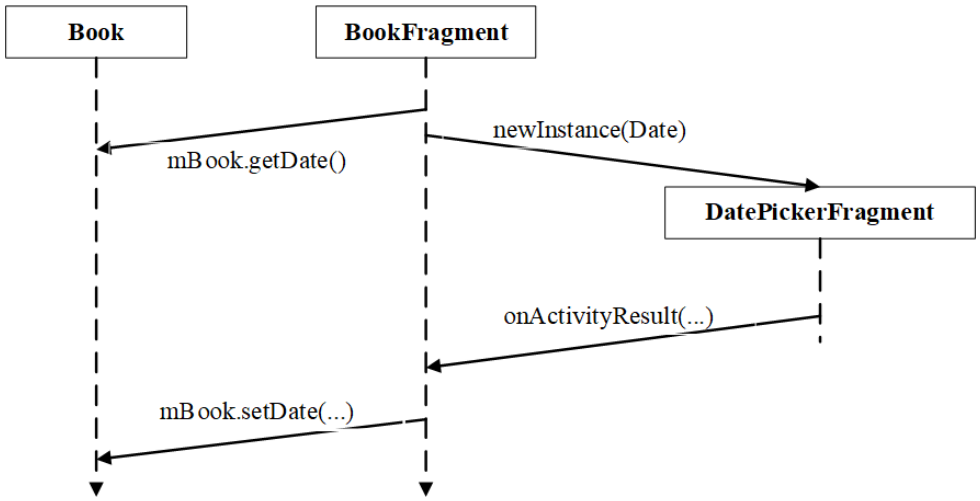

Рисунок 11.8 – Последовательность событий взаимодействия между BookFragment и DatePickerFragment

Передача данных DatePickerFragment

Чтобы получить данные в DatePickerFragment, дата будет сохраняться в пакете аргументов DatePickerFragment, где DatePickerFragment сможет обратиться к ней.

Создание аргументов фрагмента и присваивание им значений обычно выполняется в методе `newInstance()`, заменяющем конструктор фрагмента. Добавить в файл DatePickerFragment.java метод `newInstance(Date)`.

Листинг 11.5 – Добавление метода newInstance(Date) (DatePickerFragment.java)

```
public class DatePickerFragment extends DialogFragment {
    private static final String ARG_DATE = "date";
    private DatePicker mDatePicker;
    public static DatePickerFragment newInstance(Date date) {
        Bundle args = new Bundle();
        args.putSerializable(ARG_DATE, date);

        DatePickerFragment fragment = new DatePickerFragment();
        fragment.setArguments(args);
        return fragment;
    }
    ...
}
```

В классе BookFragment удалить вызов конструктора DatePickerFragment и заменить его вызовом `DatePickerFragment.newInstance(Date)`.

Листинг 11.6 – Добавление вызова newInstance() (BookFragment.java)

```
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
                container, Bundle savedInstanceState) {
        ...
        mDateButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentManager manager = getFragmentManager();
                DatePickerFragment dialog = new DatePickerFragment();
                DatePickerFragment dialog = DatePickerFragment
                        .newInstance(mBook.getDate());
                dialog.show(manager, DIALOG_DATE);
            }
        });
        ...
        return v;
    }
```

Экземпляр DatePickerFragment должен инициализировать DatePicker по информации, хранящейся в Date. Однако для инициализации DatePicker необходимо иметь целочисленные значения месяца, дня и года. Объект Date больше напоминает временную метку и не может предоставить нужные целые значения напрямую.

Чтобы получить нужные значения, следует создать объект Calendar и использовать Date для определения его конфигурации. После этого можно получить нужную информацию из Calendar.

В методе `onCreateDialog(…)` получить объект Date из аргументов и использовать его с Calendar для инициализации DatePicker.

Листинг 11.7 – Извлечение даты и инициализация DatePicker (DatePickerFragment.java)

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    Date date = (Date) getArguments().getSerializable(ARG_DATE);
    Calendar calendar = Calendar.getInstance();
    calendar.setTime(date);
    int year = calendar.get(Calendar.YEAR);
    int month = calendar.get(Calendar.MONTH);
    int day = calendar.get(Calendar.DAY_OF_MONTH);
    View v = LayoutInflater.from(getActivity())
            .inflate(R.layout.dialog_date, null);
    mDatePicker = (DatePicker)
                  v.findViewById(R.id.dialog_date_date_picker);
    mDatePicker.init(year, month, day, null);

    return new AlertDialog.Builder(getActivity())
            .setView(v)
            .setTitle(R.string.date_picker_title)
            .setPositiveButton(android.R.string.ok, null)
            .create();
}
```

Теперь BookFragment успешно сообщает DatePickerFragment, какую дату следует отобразить. Запустить приложение BookDepository и убедиться в том, что все работает так же, как прежде.

Возвращение данных BookFragment

Чтобы экземпляр BookFragment получал данные от DatePickerFragment, необходимо каким-то образом отслеживать отношения между двумя фрагментами.

С активностями вызывается `startActivityForResult(…)`, а ActivityManager отслеживает отношения между родительской и дочерней активностью. Когда дочерняя активность прекращает существование, ActivityManager знает, какая активность должна получить результат.

Для создания аналогичной связи можно назначить BookFragment *целевым фрагментом* (target fragment) для DatePickerFragment. Эта связь будет автоматически восстановлена после того, как и BookFragment, и

DatePickerFragment уничтожаются и заново создаются ОС. Для этого вызывается следующий метод Fragment:

```
public void setTargetFragment(Fragment fragment,
     int requestCode)
```

Метод получает фрагмент, который станет целевым, и код запроса, аналогичный передаваемому `startActivityForResult(…)`. По коду запроса целевой фрагмент позднее может определить, какой фрагмент возвращает информацию.

FragmentManager сохраняет целевой фрагмент и код запроса. Чтобы получить их, надо вызвать `getTargetFragment()` и `getTargetRequestCode()` для фрагмента, назначившего целевой фрагмент.

В файле BookFragment.java создать константу для кода запроса, а затем назначьте BookFragment целевым фрагментом экземпляра DatePickerFragment.

Листинг 11.8 – Назначение целевого фрагмента (BookFragment.java)

```
public class BookFragment extends Fragment {
   private static final String ARG_BOOK_ID = "book_id";
   private static final String DIALOG_DATE = "DialogDate";
   private static final int REQUEST_DATE = 0;
   ...
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup
                container, Bundle savedInstanceState) {
      ...
      mDateButton.setOnClickListener(new View.OnClickListener() {
         @Override
         public void onClick(View v) {
              FragmentManager manager = getFragmentManager();
              DatePickerFragment dialog = DatePickerFragment
                      .newInstance(mBook.getDate());
              dialog.setTargetFragment(BookFragment.this,
                           REQUEST_DATE);
              dialog.show(manager, DIALOG_DATE);
         }
      });
      mReadedCheckBox = (CheckBox) v.findViewById(R.id.book_readed);
      ...
      return v;
   }
   ...
}
```

Передача данных целевому фрагменту

Итак, связь между BookFragment и DatePickerFragment создана, и теперь нужно вернуть дату BookFragment. Дата будет включена в объект Intent как дополнение.

В классе DatePickerFragment создать закрытый метод, который создает интент, помещает в него дату как дополнение, а затем вызывает `BookFragment.onActivityResult(…)`.

Листинг 11.9 – Обратный вызов целевого фрагмента (DatePickerFragment.java)

```
public class DatePickerFragment extends DialogFragment {
    public static final String EXTRA_DATE =
            "ru.rsue.android.bookdepository.date";
    private static final String ARG_DATE = "date";
    ...
    @Override
    public Dialog onCreateDialog(Bundle savedInstanceState) {
            ...
    }
    private void sendResult(int resultCode, Date date) {
            if (getTargetFragment() == null) {
                  return;
            }
            Intent intent = new Intent();
            intent.putExtra(EXTRA_DATE, date);
            getTargetFragment()
                  .onActivityResult(getTargetRequestCode(), resultCode,
                        intent);
    }
}
```

Пришло время воспользоваться новым методом sendResult. Когда пользователь нажимает кнопку положительного ответа в диалоговом окне, приложение должно получить дату из DatePicker и отправить результат BookFragment. В коде `onCreateDialog(…)` заменить параметр null вызова `setPositiveButton(…)` реализацией `DialogInterface.OnClickListener`, которая возвращает выбранную дату и вызывает `sendResult`.

Листинг 11.10 – Передача информации (DatePickerFragment.java)

```
@Override
public Dialog onCreateDialog(Bundle savedInstanceState) {
    ...
    return new AlertDialog.Builder(getActivity())
        .setView(v)
        .setTitle(R.string.date_picker_title)
        .setPositiveButton(android.R.string.ok, null);
```

```
        .setPositiveButton(android.R.string.ok,
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                int year = mDatePicker.getYear();
                int month = mDatePicker.getMonth();
                int day = mDatePicker.getDayOfMonth();
                Date date = new GregorianCalendar(year, month, day).
                          getTime();
                sendResult(Activity.RESULT_OK, date);
            }
        }).create();
}
```

В классе BookFragment переопределить метод `onActivityResult(…)`, чтобы он возвращал дополнение, задавал дату в Book и обновлял текст кнопки даты.

Листинг 11.11 – Реакция на получение данных от диалогового окна (BookFragment.java)

```
public class BookFragment extends Fragment {
    ...
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
                container, Bundle savedInstanceState) {
            ...
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode,
        Intent data) {
            if (resultCode != Activity.RESULT_OK) {
                    return;
            }
            if (requestCode == REQUEST_DATE) {
                Date date = (Date) data
                        .getSerializableExtra(DatePickerFragment.EXTRA_DATE
      );
                mBook.setDate(date);
                mDateButton.setText(mBook.getDate().toString());
            }
    }
}
```

Код, задающий текст кнопки, идентичен коду из `onCreateView(…)`. Чтобы избежать задания текста в двух местах, необходимо инкапсулировать этот код в закрытом методе `updateDate()`, а затем вызвать его в `onCreateView(…)` и `onActivityResult(…)`.

Это можно сделать вручную или поручить работу Android Studio. Выделить всю строку кода, которая задает текст mDateButton, щёлкнуть на ней правой кнопкой мыши и выбрать команду Refactor→Extract→Method... (рисунок 11.9).

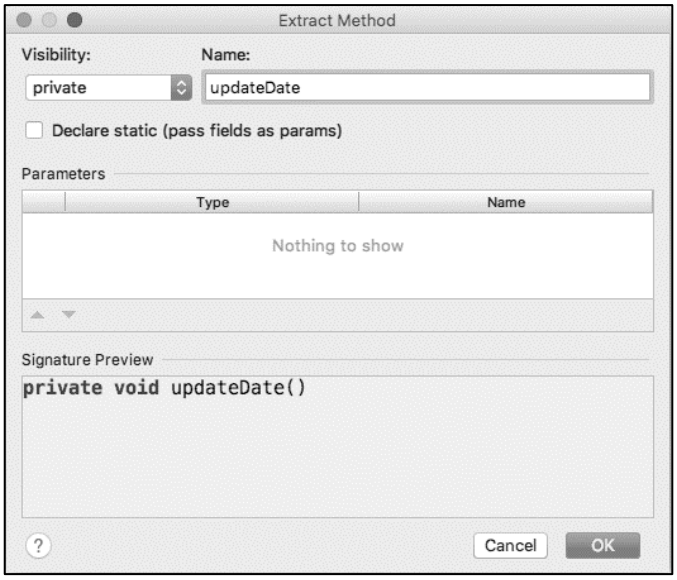

Рисунок 11.9 – Извлечение метода в Android Studio

Выбрать закрытый уровень видимости метода и ввести имя **updateDate**. Щёлкнуть на кнопке OK; среда Android Studio сообщает, что ей удалось найти еще одно место, в котором использовалась эта строка кода. Щёлкнуть на кнопке Yes, чтобы разрешить Android Studio обновить второе вхождение. Убедиться в том, что код был выделен в метод updateDate (листинг 11.12).

Листинг 11.12 – Выделение кода в метод updateDate() (BookFragment.java)

```
public class BookFragment extends Fragment {
    ...
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
                container, Bundle savedInstanceState) {
       View v = inflater.inflate(R.layout.fragment_book, container,
                false);
       ...
       mDateButton = (Button) v.findViewById(R.id.book_date);
       updateDate();
       ...
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent
                data)
```

```
    {
        if (resultCode != Activity.RESULT_OK) {
                  return;
        }
        if (requestCode == REQUEST_DATE) {
             Date date = (Date) data
             .getSerializableExtra(DatePickerFragment.EXTRA_DATE);
             mBook.setDate(date);
             updateDate();
        }
    }
    private void updateDate() {
        mDateButton.setText(mBook.getDate().toString());
    }
}
```

Запустить приложение BookDepository и убедиться в том, что пользователь действительно может управлять датой. Изменить дату Book; новая дата должна появиться в представлении BookFragment. Вернуться к списку книг, проверить дату Book и убедиться в том, что уровень модели действительно обновлен.

Самостоятельные задания.

Задание 1. Новые диалоговые окна

Написать еще один диалоговый фрагмент TimePickerFragment для выбора времени начала прочтения книги. Использовать виджет TimePicker, Добавить в BookFragment еще одну кнопку для отображения TimePickerFragment.

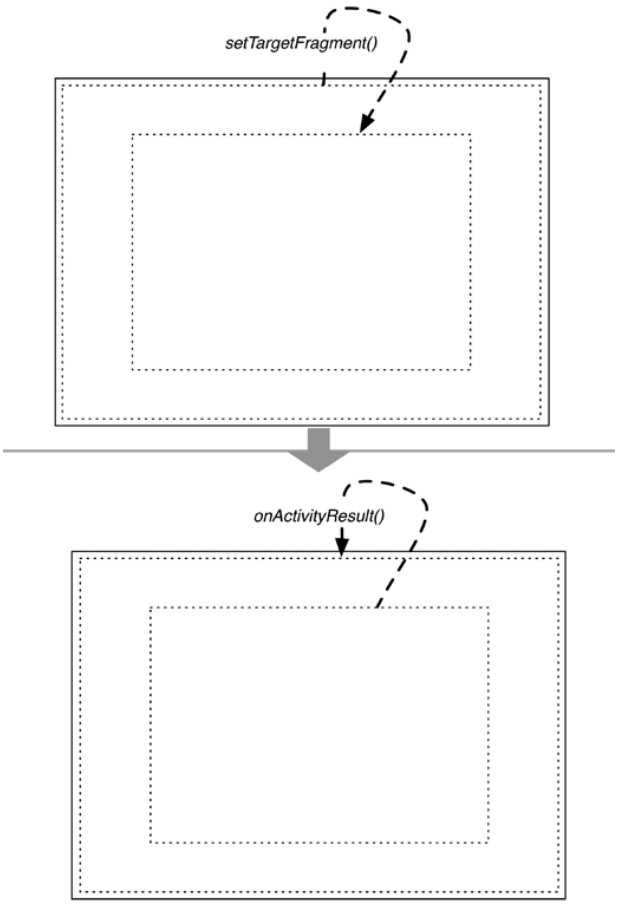

Рисунок 11.10 – Взаимодействие между фрагментами на планшетах

Задание 2. DialogFragment

Необходимо измененить представление DatePickerFragment.

Первая часть — представить представление DatePickerFragment с переопределением onCreateView вместо onCreateDialog. При таком способе создания DialogFragment не будет отображаться со встроенными областями заголовка и кнопок в верхней и нижней части диалогового окна. Необходимо самостоятельно создать кнопку OK в dialog_date.xml.

После того как представление DatePickerFragment будет создано в onCreateView, можно отобразить фрагмент DatePickerFragment как диалоговое окно или встроить его в активность.

Во второй части — создать новый субкласс SingleFragmentActivity и сделать эту активность хостом для DatePickerFragment. При таком представлении DatePickerFragment будет использоваться механизм startActivityForResult для возвращения даты BookFragment. В DatePickerFragment, если целевой фрагмент не существует, использовать метод `setResult(int, intent)` активности-хоста для возвращения даты фрагменту.