



*Белорусский государственный университет  
информатики и радиоэлектроники*

---

## **Эргономика мобильных приложений**

**Преподаватель: Меженная Марина Михайловна**

К.Т.Н., доцент,

доцент кафедры инженерной психологии и эргономики  
а 609-2

[mezhennaya@bsuir.by](mailto:mezhennaya@bsuir.by)



---

*Кафедра инженерной психологии и эргономики*

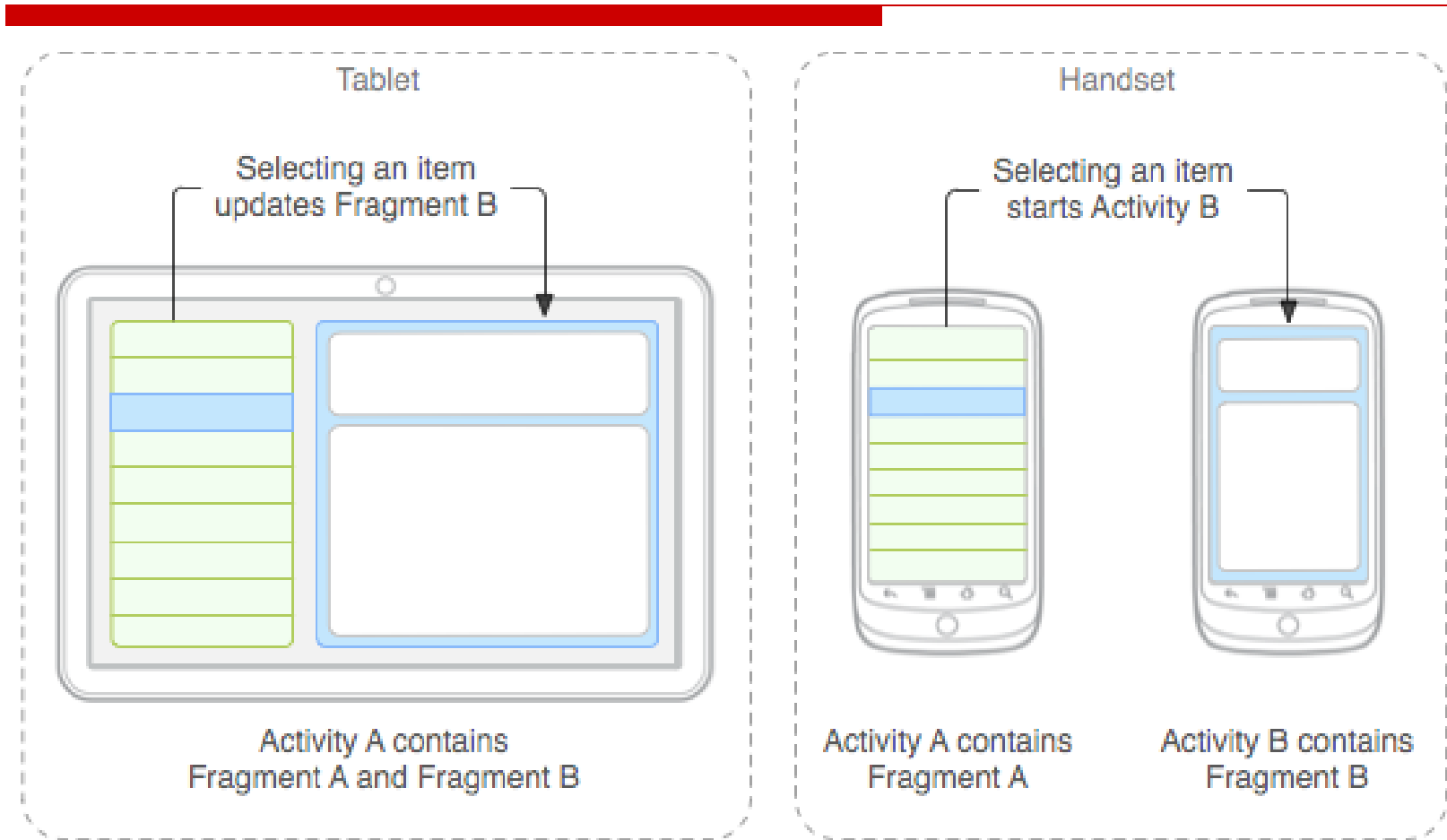
# Лекция 9: Фрагменты

---

## План лекции:

1. Что такое и зачем нужны фрагменты?
2. Создание фрагмента и его жизненный цикл.
3. Статическое подключение фрагмента к Activity.
4. Динамическое подключение фрагмента к Activity. Основные методы динамической работы с фрагментом.
5. Пример динамической работы с фрагментами (со сменой ориентации экрана, подключением списка данных из SQLite в фоновом режиме, интерфейсами в обратной связи).

# Что такое и зачем нужны фрагменты?



# Что такое и зачем нужны фрагменты?

---

Fragment — модульная часть Activity, у которой свой жизненный цикл и свои обработчики различных событий.

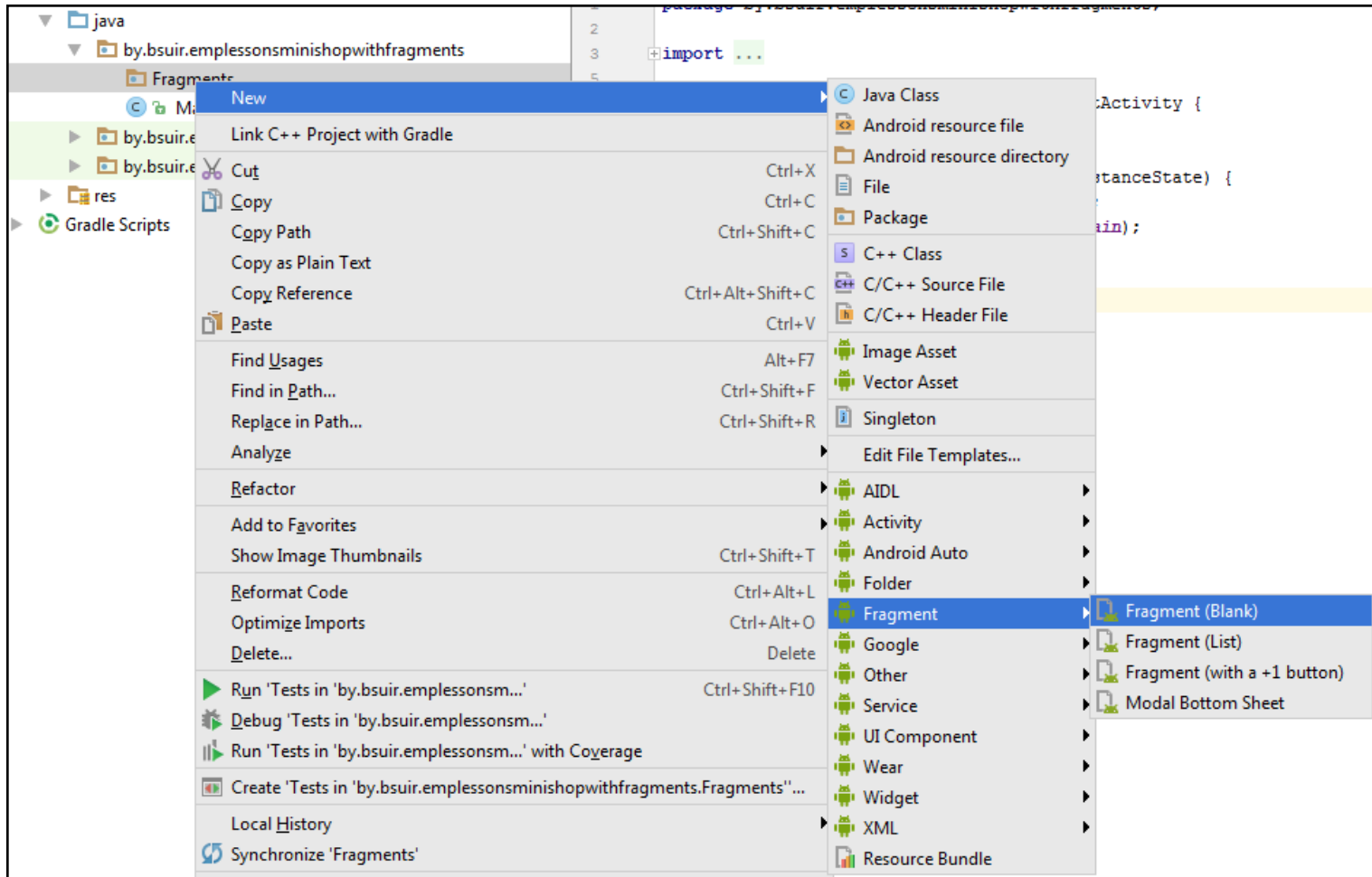
Android добавил фрагменты с API 11, для того, чтобы разработчики могли разрабатывать более гибкие пользовательские интерфейсы на больших экранах, таких как экраны планшетов.

Плюсы по сравнению с использованием нескольких Activity:

- С помощью них можно легко сделать дизайн, адаптивный под планшеты.
- Разделение кода на функциональные модули, а следовательно, поддержка кода обходится дешевле.


# Создание фрагмента

## 1. Создаем класс `public class Fragment1 extends Fragment`



# Создание фрагмента

New Android Component

 **Configure Component**  
Android Studio

Creates a blank fragment that is compatible back to API level 4.


Fragment Name:

☒ Create layout XML?

Fragment Layout Name:

☐ Include fragment factory methods?

☐ Include interface callbacks?



The name of the layout to create

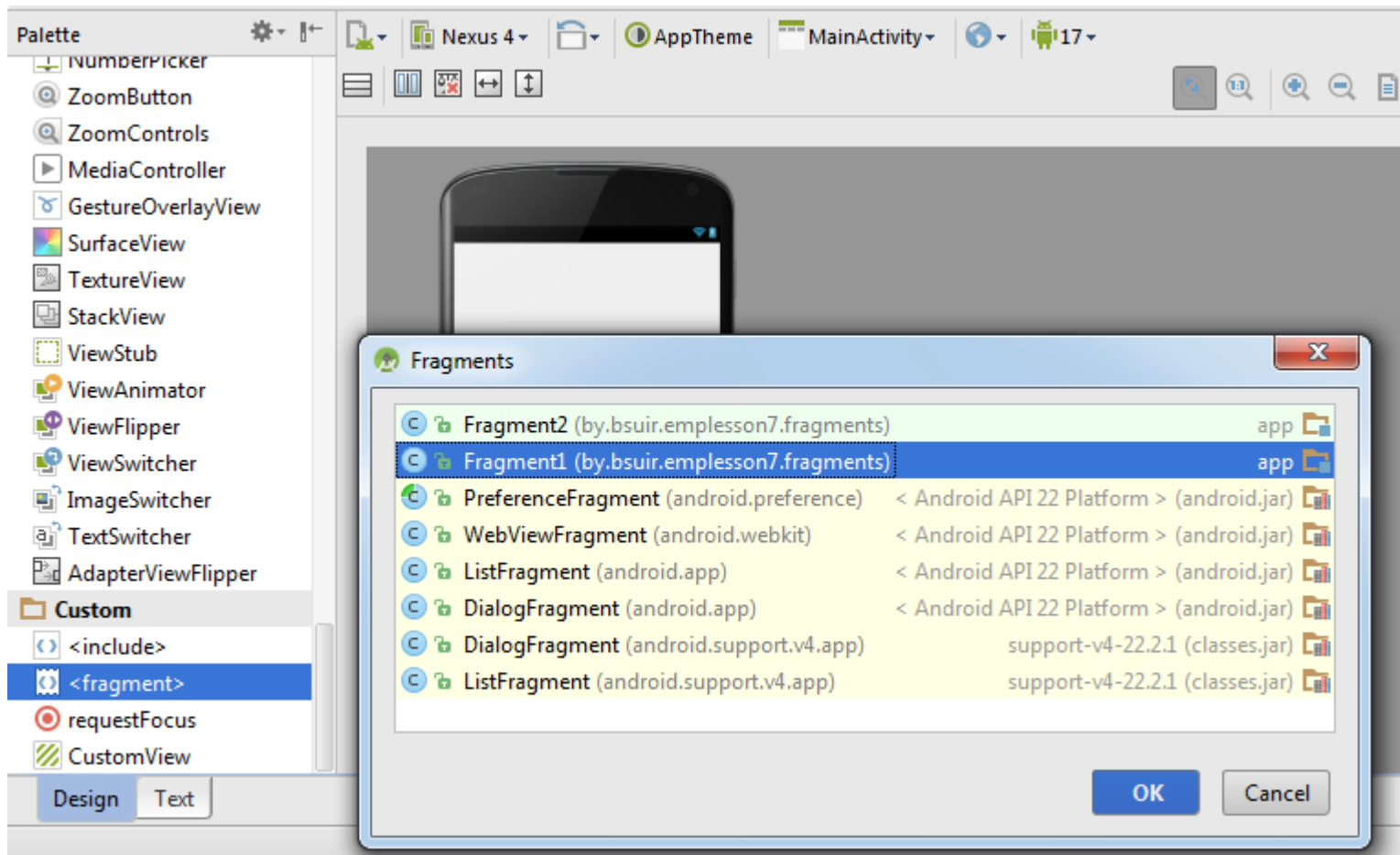
# Создание фрагмента

---

2. Наполняем xml-файл разметки.
3. Добавляем фрагмент в Activity одним из двух способов.
  - 1) В activity\_main.xml фрагмент добавляется как элемент `<fragment>` (статическое добавление фрагмента).
  - 2) или можно добавить его в существующий объект `ViewGroup` непосредственно в коде приложения (динамическое добавление фрагмента).

# Статическое добавление фрагмента

## 3.1.1 К activity\_main.xml добавляем Fragment1.

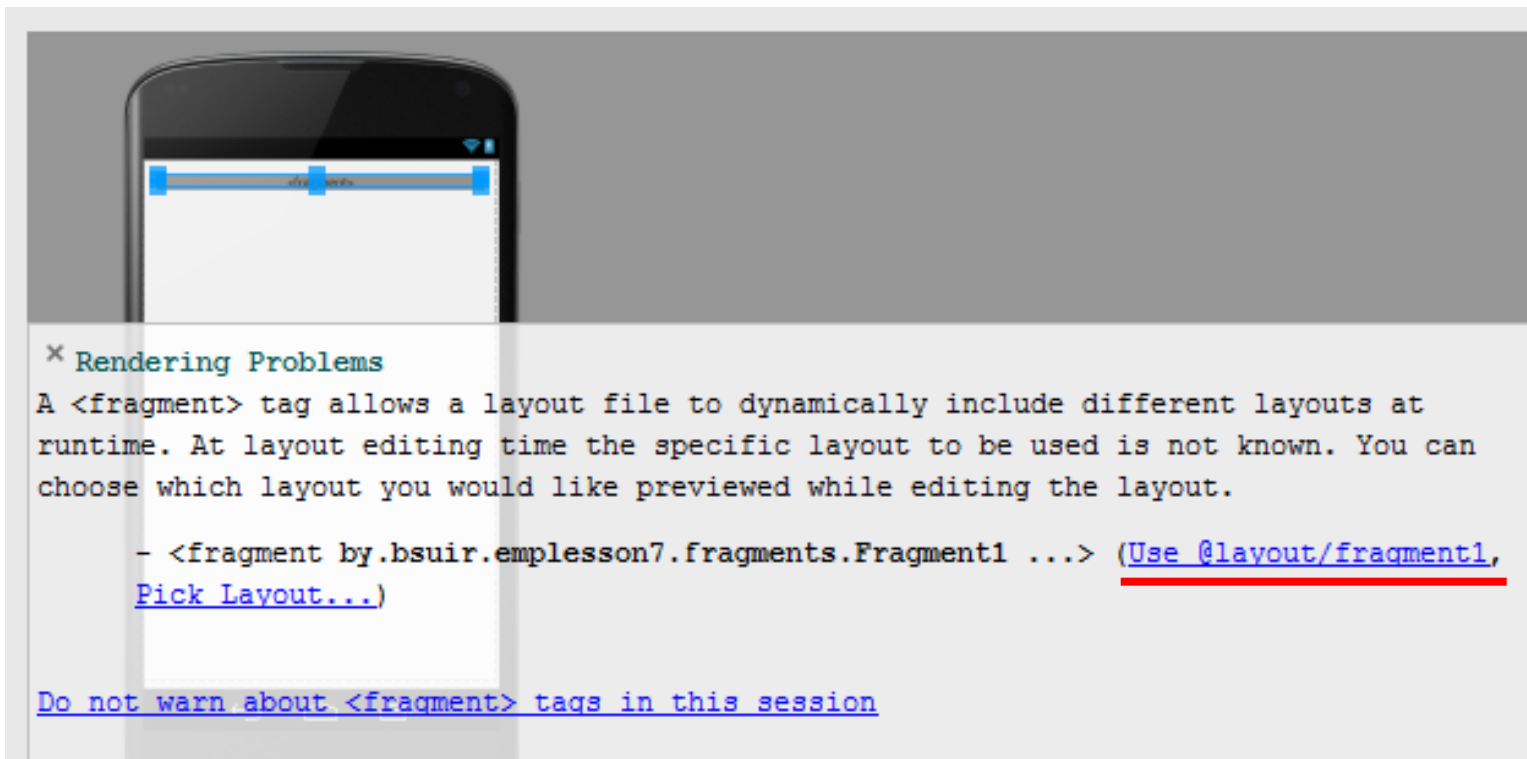




# Статическое добавление фрагмента

---

## 3.1.2 Активизируем разметку Fragment1.



## Статическое подключение фрагмента к Activity: activity\_main.xml

---

```
<LinearLayout
```

```
...
```

```
>
```

```
    <fragment
```

```
        android:layout_width="match_parent "
```

```
        android:layout_height="match_parent"
```

```
        android:name="by.bsuir.emplesson7.fragments.Fragment1"
```

```
        android:id="@+id/fragment"
```

```
        android:layout_gravity="center_horizontal"
```

```
        tools:layout="@layout/fragment1"/>
```

```
</LinearLayout>
```

---

# Динамическое подключение фрагмента к Activity: activity\_main.xml

3.2.1 К activity\_main.xml добавляем FrameLayout.

```
<LinearLayout
```

```
...
```

```
>
```

```
    <FrameLayout
```

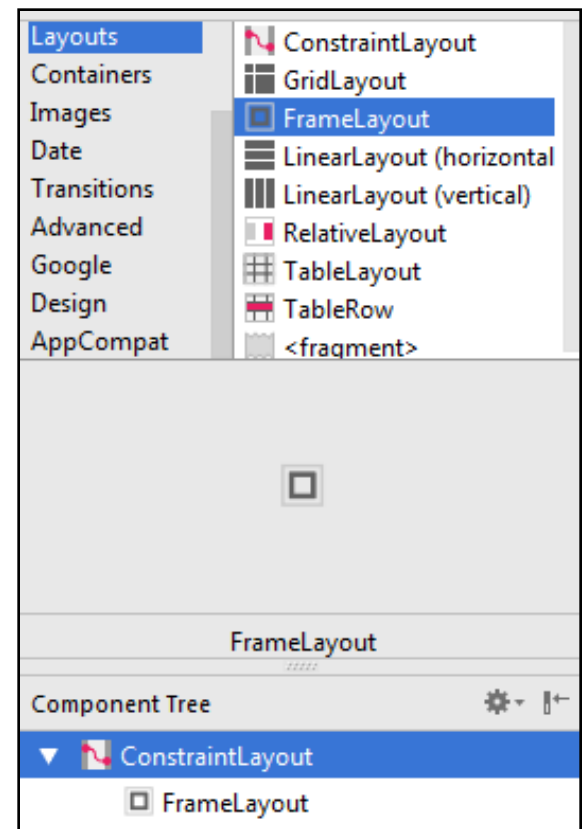
```
        android:id="@+id/fragment_container"
```

```
        android:layout_width="match_parent"
```

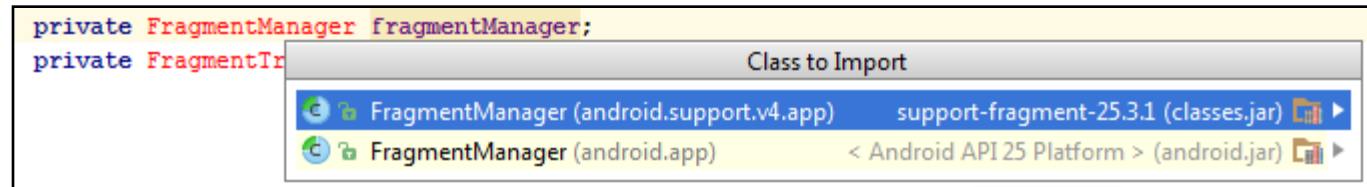
```
        android:layout_height="match_parent">
```

```
    </FrameLayout>
```

```
</LinearLayout>
```



## Динамическое подключение фрагмента к Activity: class MainActivity



3.2.2 Добавляем фрагмент к контейнеру в коде:

```
import android.support.v4.app.FragmentManager;  
import android.support.v4.app.FragmentTransaction;  
import android.support.v7.app.AppCompatActivity;  
import android.os.Bundle;
```

```
public class MainActivity extends ActionBarActivity {
```

```
    private Fragment1 fragment1;  
    private FragmentManager fragmentManager;  
    private FragmentTransaction fragmentTransaction;
```

## Динамическое подключение фрагмента к Activity:

### class MainActivity

@Override

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    initFragments();  
}
```

```
private void initFragments() {  
    fragment1 = new Fragment1();  
    fragmentManager = getSupportFragmentManager();  
    fragmentTransaction = fragmentManager.beginTransaction();  
    fragmentTransaction.add(R.id.fragment_container, fragment1);  
    fragmentTransaction.commit();  
}
```

## Динамическое подключение фрагмента к Activity:

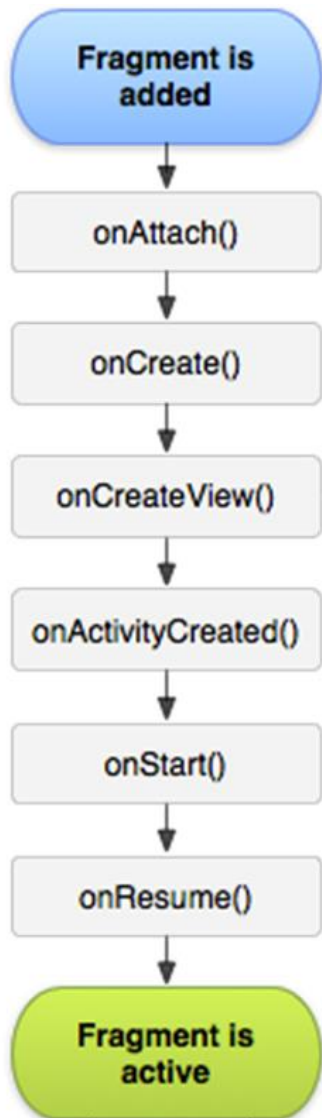
### class MainActivity

---

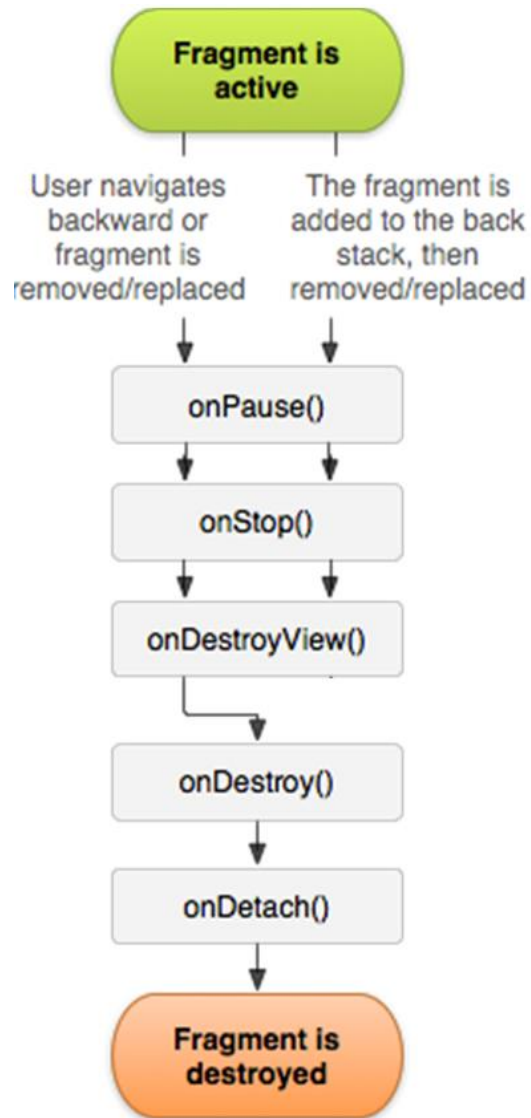
Класс **FragmentManager** служит для управления фрагментами. Чтобы получить его, следует вызвать метод `getFragmentManager()` из кода операции.

Класс **FragmentTransaction** позволяет выполнять транзакции с фрагментами, например, добавление и удаление.

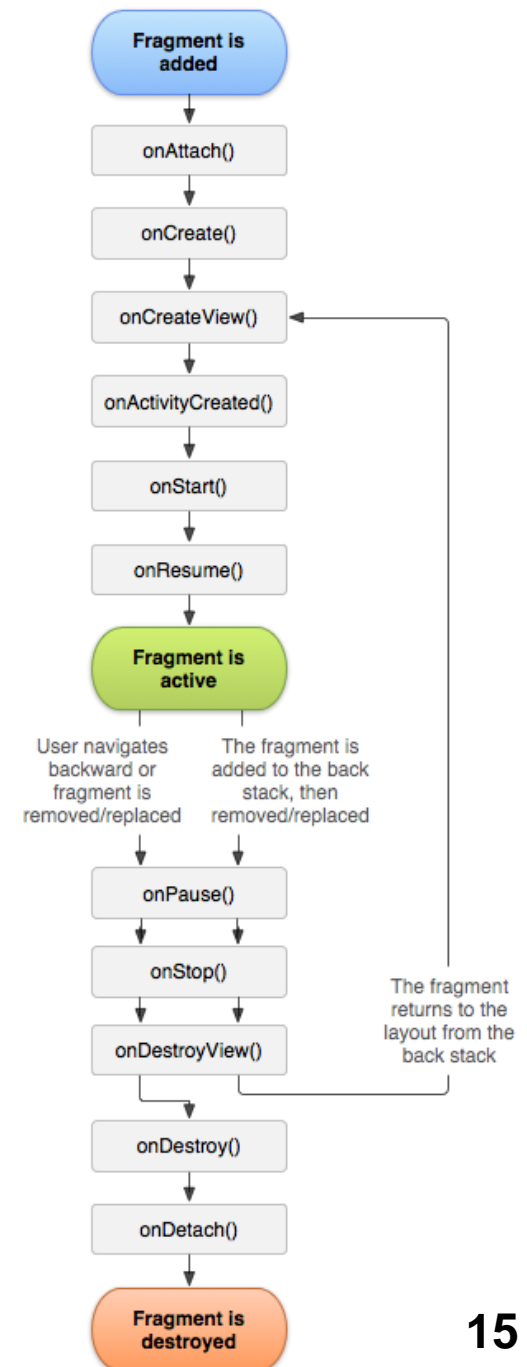
# Жизненный цикл фрагмента



+



=



# Жизненный цикл фрагмента

Чтобы отследить жизненный цикл фрагмента необходимо добавить логи в следующие методы класса Fragment1:

1. `public void onAttach(Activity activity)`
2. `public void onCreate(Bundle savedInstanceState)`
3. `public View onCreateView(...)`
4. `public void onActivityCreated(Bundle savedInstanceState)`
5. `public void onStart()`
6. `public void onResume()`
7. `public void onPause()`
8. `public void onStop()`
9. `public void onDestroyView()`
10. `public void onDestroy()`
11. `public void onDetach()`



# Жизненный цикл фрагмента

---

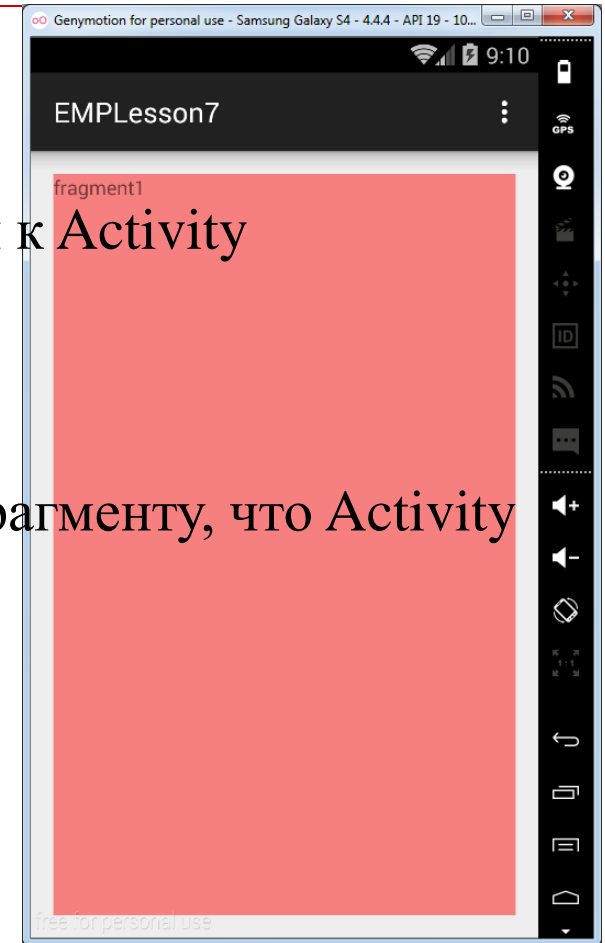
Чтобы отследить жизненный цикл Activity с фрагментом необходимо добавить логи в следующие методы MainActivity.java:

1. `public void onCreate(Bundle savedInstanceState)`
2. `public void onStart()`
3. `public void onResume()`
4. `public void onPause()`
5. `public void onStop()`
6. `public void onDestroy()`

# Жизненный цикл фрагмента

Запускаем:

Fragment1 **onAttach** // фрагмент прикреплен к Activity  
Fragment1 **onCreate**  
Fragment1 **onCreateView**  
MainActivity **onCreate**  
Fragment1 **onActivityCreated** // сообщает фрагменту, что Activity  
создано  
MainActivity **onStart**  
Fragment1 **onStart**  
MainActivity **onResume**  
Fragment1 **onResume**



# Жизненный цикл фрагмента

---

Закрываем:

Fragment1 onPause

MainActivity onPause

Fragment1 onStop

MainActivity onStop

Fragment1 onDestroyView

Fragment1 **onDestroy**

Fragment1 onDetach

MainActivity onDestroy

# Динамическое подключение фрагмента к Activity:

## методы

---

`add()` — добавление фрагмента

`remove()` — удаление фрагмента

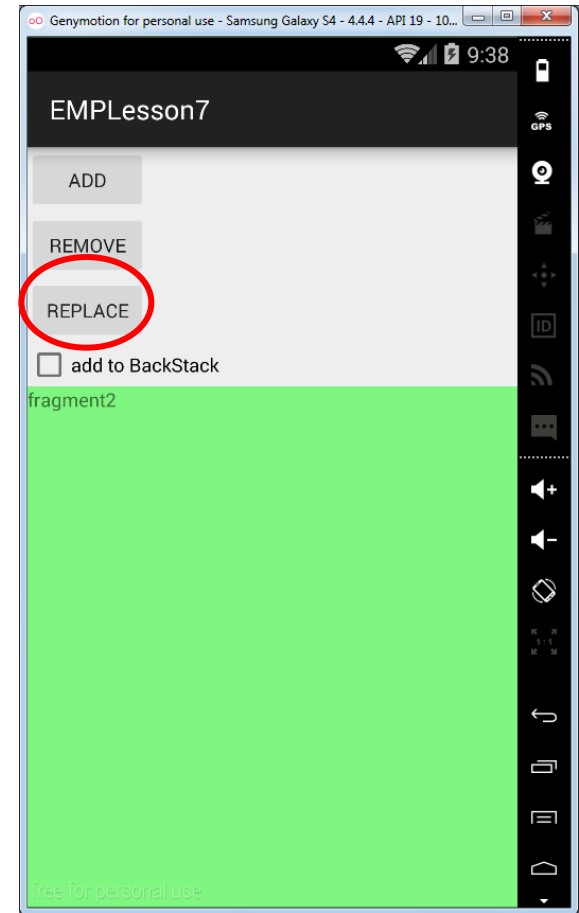
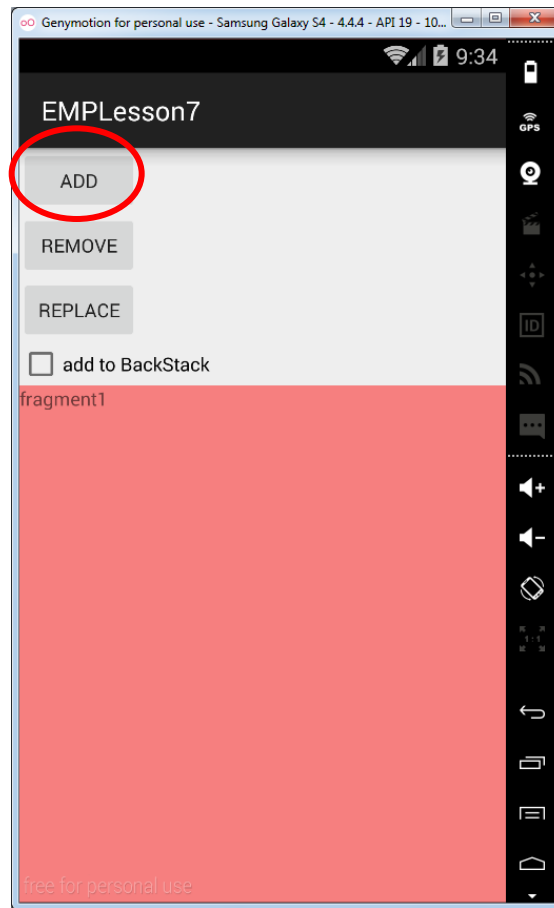
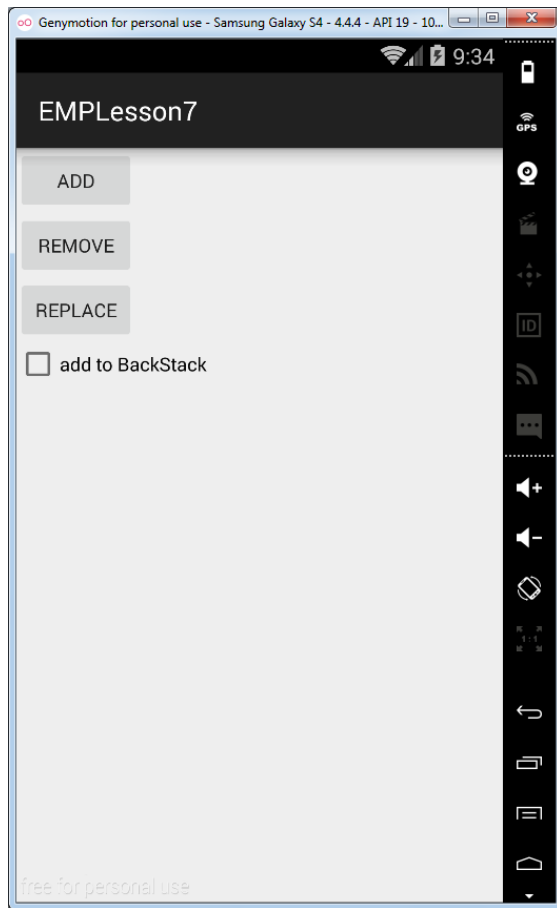
`replace()` — замена фрагмента

`hide()` — делает фрагмент невидимым

`show()` — отображает фрагмент

Чтобы добавлять наши транзакции в стек, как это происходит по умолчанию с активностями, можно использовать `addToBackStack(String)`.

# Динамическое подключение фрагмента к Activity: activity\_main.xml



# Динамическое подключение фрагмента к Activity:

## методы

---

```
public class MainActivity extends ActionBarActivity implements
View.OnClickListener{

    private Fragment1 frag1;
    private Fragment2 frag2;
    private FragmentTransaction fTrans;
    private Button btnAdd, btnRemove, btnReplace;
    private CheckBox chbStack;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initView();
        initFragments();
    }
}
```

---

# Динамическое подключение фрагмента к Activity:

## методы

---

```
private void initView(){
    btnAdd = (Button) findViewById(R.id.btnAdd);
    btnAdd.setOnClickListener(this);
    btnRemove = (Button) findViewById(R.id.btnRemove);
    btnRemove.setOnClickListener(this);
    btnReplace = (Button) findViewById(R.id.btnReplace);
    btnReplace.setOnClickListener(this);
    chbStack = (CheckBox) findViewById(R.id.chbStack);
}

private void initFragments() {
    frag1 = new Fragment1();
    frag2 = new Fragment2();
}
```

# Динамическое подключение фрагмента к Activity:

## методы

---

```
public void onClick(View v) {  
    fTrans = getSupportFragmentManager().beginTransaction();  
    switch (v.getId()) {  
        case R.id.btnAdd:  
            fTrans.add(R.id.fragmentContainer, frag1);  
            break;  
        case R.id.btnRemove:  
            fTrans.remove(frag1);  
            break;  
        case R.id.btnReplace:  
            fTrans.replace(R.id.fragmentContainer, frag2);  
        default:  
            break;  
    }  
    if (chbStack.isChecked()) fTrans.addToBackStack(null);  
    fTrans.commit();  
}
```

---



# Доступ к фрагменту из Activity: метод `findFragmentById`

---

## **MainActivity.java:**

```
public void onClick(View v) {  
  
    // для статических фрагментов: указываем id фрагмента  
    Fragment frag1 = getFragmentManager().findFragmentById(R.id.fragment1);  
    TextView tv_frag1 = (TextView) frag1.getView().findViewById(R.id.tv_frag1);  
    tv_frag1.setText("Access to Fragment 1 from Activity");  
  
    // для динамических фрагментов: указываем id контейнера  
    Fragment frag2 = getFragmentManager().findFragmentById(R.id.fragment_container);  
    TextView tv_frag2 = (TextView) frag2.getView().findViewById(R.id.tv_frag2);  
    tv_frag2.setText("Access to Fragment 2 from Activity");  
  
    // иногда сокращенно пишут так:  
    // ((TextView) frag2.getView().findViewById(R.id.tv_frag2)).setText("Access to Fragment 2  
    from Activity");  
}
```

---

# Доступ к Activity из фрагмента: метод getActivity

---

## Fragment1.java:

```
public void onClick(View v) {  
  
    public void onClick(View v) {  
        TextView tv_activity = getActivity().findViewById(R.id.tv_activity);  
        tv_activity.setText("Access from Fragment1");  
  
        // иногда сокращенно пишут так:  
        //      ((TextView)getActivity().findViewById(R.id.tv_activity)).setText("Access from  
        Fragment1");  
    }  
}
```

**Обработка в Activity (или в другом фрагменте) события из фрагмента...**

---

**Выполняется посредством интерфейса в качестве прослойки для отслеживания событий.**

**Нельзя напрямую связываться из одного фрагмента с другим!**

## Что еще нужно знать о фрагментах?

---

Чтобы из фрагмента «достучаться» до Activity используется метод:

**getActivity()**

Чтобы из фрагмента достать Context используется тот же метод:

**getActivity()**

## Рассмотрим более сложный пример: *портретная ориентация*

MiniShopWithFragments

Goods

1	My good №1
2	My good №2
3	My good №3
4	My good №4
5	My good №5
6	My good №6
7	My good №7
8	My good №8
9	My good №9
10	My good №10
11	My good №11
12	My good №12
13	My good №13

Count of goods = 3

SHOW CHECKED ITEMS

MiniShopWithFragments

12	My good №12	<input type="checkbox"/>
13	My good №13	<input type="checkbox"/>
14	My good №14	<input type="checkbox"/>
15	My good №15	<input type="checkbox"/>
16	My good №16	<input type="checkbox"/>
17	My good №17	<input type="checkbox"/>
18	My good №18	<input checked="" type="checkbox"/>
19	My good №19	<input type="checkbox"/>
20	My good №20	<input type="checkbox"/>
21	My good №21	<input checked="" type="checkbox"/>
22	My good №22	<input checked="" type="checkbox"/>
23	My good №23	<input type="checkbox"/>
24	My good №24	<input type="checkbox"/>
25	My good №25	<input type="checkbox"/>

**MainActivity** с одним контейнером  
в портретной ориентации  
(в контейнере FragmentWithAllGoods)

**OrderActivity**  
с одним контейнером  
в портретной ориентации  
(в контейнере FragmentWithOrder)

MiniShopWithFragments

My Order

Total cost = 61

18	My good №18
21	My good №21
22	My good №22

ORDER

for personal use

## Рассмотрим более сложный пример: *книжная ориентация*

### OrderActivity

с не имеет отдельной книжной разметки за ненадобностью...

MiniShopWithFragments

Goods		My Order Total cost = 61	
1	My good №1 <input type="checkbox"/>	18	My good №18
2	My good №2 <input type="checkbox"/>	21	My good №21
3	My good №3 <input type="checkbox"/>	22	My good №22
4	My good №4 <input type="checkbox"/>		
5	My good №5 <input type="checkbox"/>		
6	My good №6 <input type="checkbox"/>		

ORDER

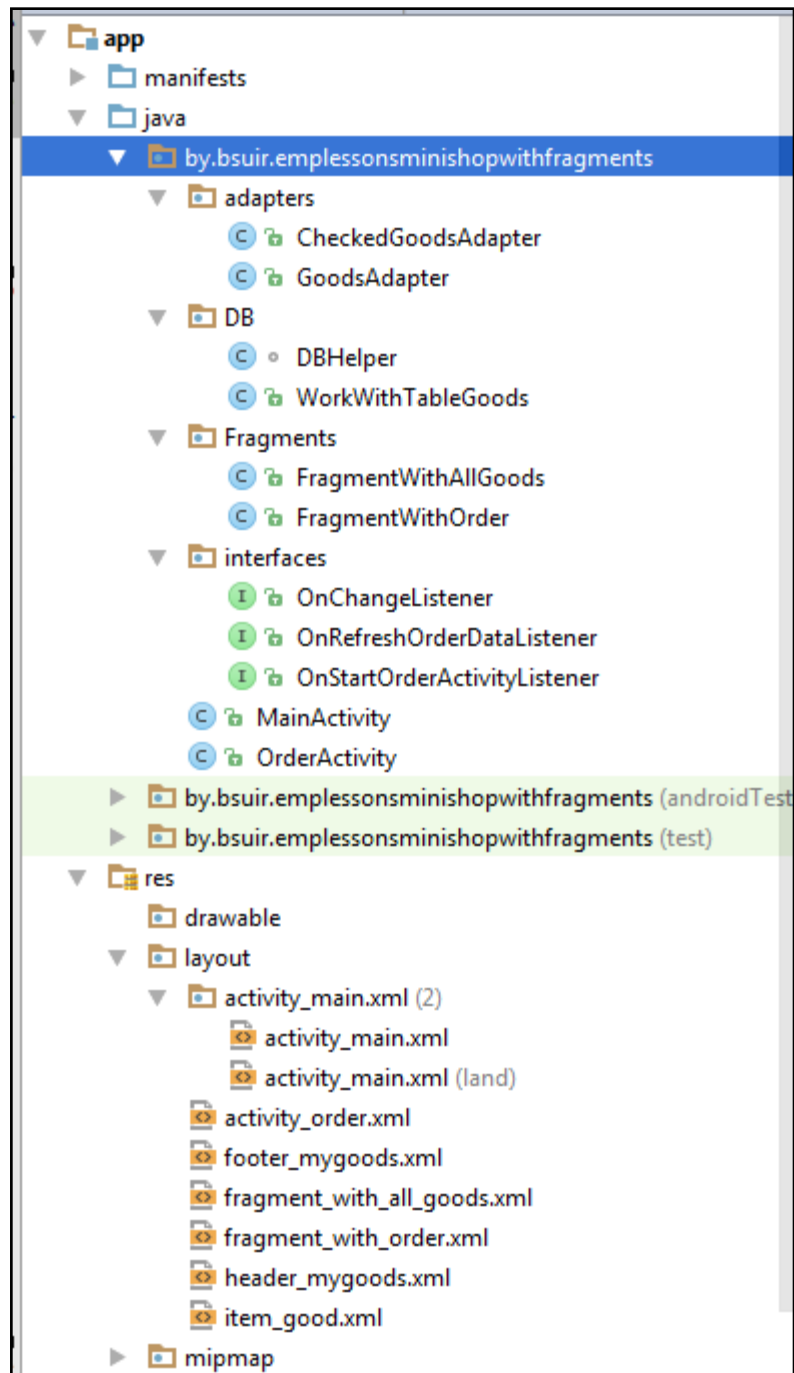
**MainActivity** с двумя контейнерами  
в книжной ориентации (в контейнерах  
FragmentWithAllGoods и FragmentWithOrder)

## Рассмотрим более сложный пример: *книжная ориентация*

MiniShopWithFragments		
		My Order
		Total cost = 110
19 My good №19	<input type="checkbox"/>	18 My good №18
20 My good №20	<input type="checkbox"/>	21 My good №21
21 My good №21	<input checked="" type="checkbox"/>	22 My good №22
22 My good №22	<input checked="" type="checkbox"/>	24 My good №24
23 My good №23	<input type="checkbox"/>	25 My good №25
24 My good №24	<input checked="" type="checkbox"/>	
25 My good №25	<input checked="" type="checkbox"/>	
Count of goods = 5		ORDER

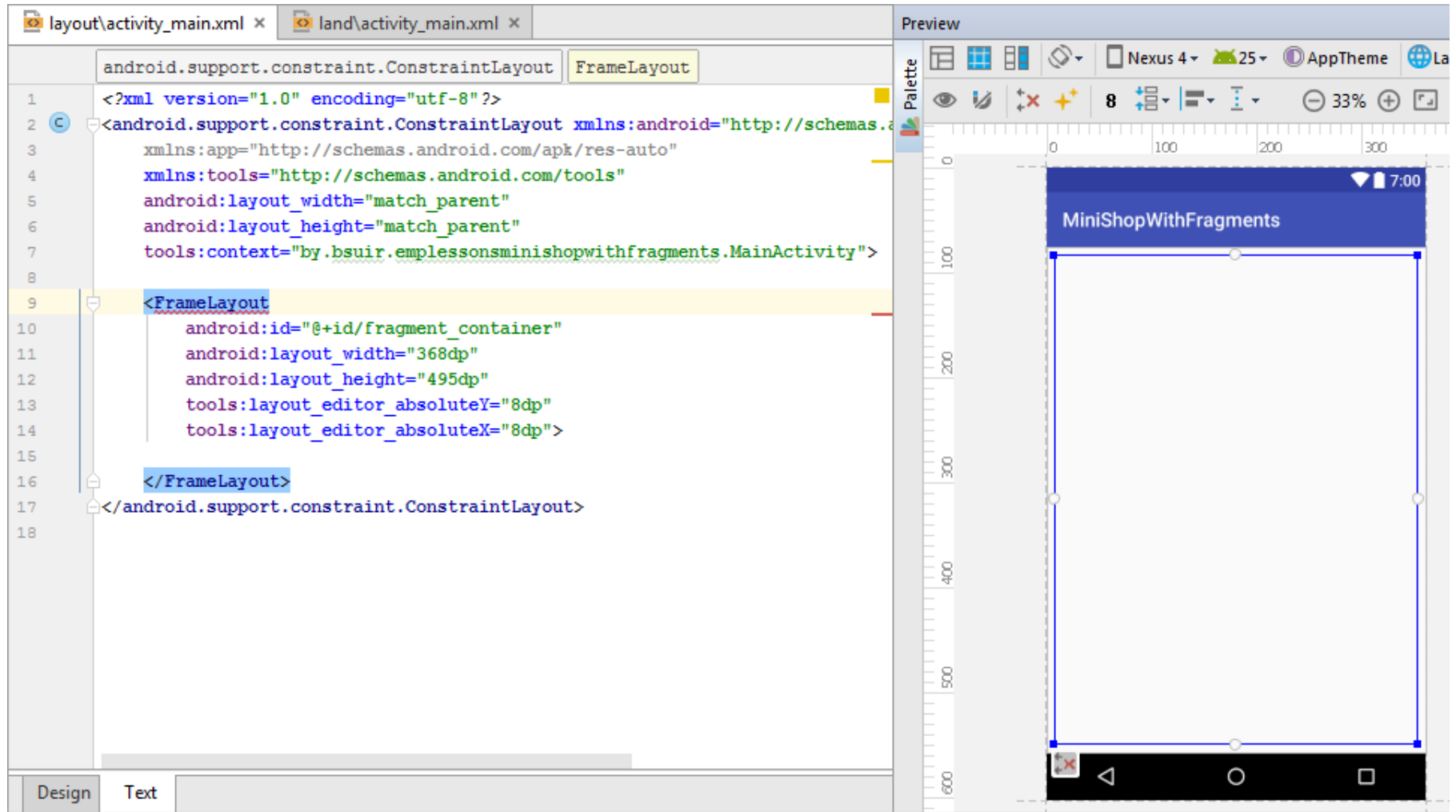
Кнопка Show checked items сделана невидимой,  
выбранные товары в динамическом режиме  
отображаются в корзине справа

# Структура проекта





# activity\_main.xml



# activity\_main.xml (land)

The image shows the Android Studio interface with the XML editor on the left and the Design preview on the right.

**XML Editor (Left):**

- File: layout\activity\_main.xml
- Tab: land\activity\_main.xml
- Palette: android.support.constraint.ConstraintLayout, FrameLayout
- Code (lines 1-27):

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="by.bsuir.emplelessonsminishopwithfragments.M"
    >
    <FrameLayout
        android:id="@+id/fragment_container"
        android:layout_width="0dp"
        android:layout_height="295dp"
        android:layout_marginTop="8dp"
        app:layout_constraintHorizontal_weight="1"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/fragment_ma"
        app:layout_constraintTop_toTopOf="parent">
    </FrameLayout>
    <FrameLayout
        android:id="@+id/fragment main order container"
        android:layout_width="0dp"
        android:layout_height="295dp"
        android:layout_marginTop="8dp"
        app:layout_constraintHorizontal_weight="1"
        app:layout_constraintLeft_toRightOf="@+id/fragment co
```

**Design Preview (Right):**

- Device: Nexus 4
- API Level: 25
- Theme: AppTheme
- Language: English
- Zoom: 33%
- Visual elements: A blue header bar labeled "MiniShopWithFragments" at the top. Below it is a large white rectangular area representing the main content container. The interface is shown in landscape orientation with a black navigation bar on the right side.

# fragment\_with\_all\_goods.xml

The screenshot displays the Android Studio IDE. The left pane shows the XML layout file `fragment_with_all_goods.xml` in Text mode. The XML code defines a `ConstraintLayout` containing a `ListView` with 8 items. The `ListView` is configured with `android:layout_width="wrap_content"`, `android:layout_height="wrap_content"`, and `android:id="@+id/listView"`. A TODO comment indicates the need to update the blank fragment layout. The right pane shows the Design mode preview of the layout on a Nexus 4 device. The preview displays a blue header titled "MiniShopWithFragments" and a list of 8 items, each with a sub-item. The list items are: Item 1 (Sub Item 1), Item 2 (Sub Item 2), Item 3 (Sub Item 3), Item 4 (Sub Item 4), Item 5 (Sub Item 5), Item 6 (Sub Item 6), Item 7 (Sub Item 7), and Item 8 (Sub Item 8). The device status bar at the top shows the time as 7:00 and the battery level at 33%.

```
1 <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
2   xmlns:tools="http://schemas.android.com/tools"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   tools:context="by.bsuir.emplelessonsminishopwithfragments.F
6
7   <!-- TODO: Update blank fragment layout -->
8
9   <ListView
10     android:layout_width="wrap_content"
11     android:layout_height="wrap_content"
12     android:id="@+id/listView" />
13 </android.support.constraint.ConstraintLayout>
14
```

# fragment\_with\_order.xml

The image shows the Android Studio interface with the XML editor on the left and the Design preview on the right.

**XML Code (fragment\_with\_order.xml):**

```
29 </LinearLayout>
30
31
32 <LinearLayout
33     android:orientation="vertical"
34     android:layout_width="match_parent"
35     android:layout_height="0dp"
36     android:layout_weight="5">
37
38     <ListView
39         android:layout_width="wrap_content"
40         android:layout_height="wrap_content"
41         android:id="@+id/second_listView" />
42
43 </LinearLayout>
44
45 <LinearLayout xmlns:android="http://schemas.android.com/a
46     android:layout_width="match_parent"
47     android:layout_height="0dp"
48     android:orientation="horizontal"
49     android:background="@android:color/holo_orange_dark"
50     android:layout_weight="1">
51
52     <Button
53         android:layout_width="310dp"
54         android:layout_height="60dp"
55         android:text="ORDER"
```

**Design Preview:**

The preview shows a mobile app interface for "MiniShopWithFragments". It features a blue header bar with the title "MiniShopWithFragments" and a green bar below it labeled "My Order". Below the green bar is a list of six items, each with a title and a subtitle:

- Item 1  
Sub Item 1
- Item 2  
Sub Item 2
- Item 3  
Sub Item 3
- Item 4  
Sub Item 4
- Item 5  
Sub Item 5
- Item 6  
Sub Item 6

At the bottom of the list is an orange bar containing a grey button labeled "ORDER". The interface is displayed on a Nexus 4 device with a 7:00 time and 33% zoom.

# MainActivity.java

```
package by.bsuir.emplelessonsminishopwithfragments;
```

```
import android.content.Intent;
```

```
import android.content.res.Configuration;
```

```
import android.support.v4.app.FragmentManager;
```

```
import android.support.v4.app.FragmentTransaction;
```

```
import android.support.v7.app.AppCompatActivity;
```

```
import android.os.Bundle;
```

```
import by.bsuir.emplelessonsminishopwithfragments.Fragments.FragmentWithAllGoods;
```

```
import by.bsuir.emplelessonsminishopwithfragments.Fragments.FragmentWithOrder;
```

```
import by.bsuir.emplelessonsminishopwithfragments.interfaces.OnRefreshOrderDataListener;
```

```
import by.bsuir.emplelessonsminishopwithfragments.interfaces.OnStartOrderActivityListener;
```

## MainActivity.java

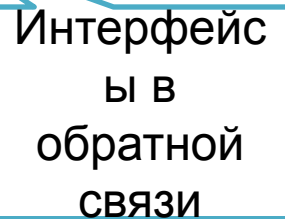
```
public class MainActivity extends AppCompatActivity implements  
OnStartOrderActivityListener, OnRefreshOrderDataListener {
```

```
    private FragmentWithAllGoods fragmentWithAllGoods;  
    private FragmentWithOrder fragmentWithOrder;
```

```
    private FragmentManager fragmentManager;  
    private FragmentTransaction fragmentTransaction;
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        initFragments();  
    }
```



Интерфейс  
ы в  
обратной  
связи

```

private void initFragments() {
    fragmentWithAllGoods = new FragmentWithAllGoods();
    fragmentWithOrder = new FragmentWithOrder();
    fragmentManager = getSupportFragmentManager();
    fragmentTransaction = fragmentManager.beginTransaction();
    if (getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_PORTRAIT) {
        fragmentTransaction.add(R.id.fragment_container,
fragmentWithAllGoods);
    }
    if (getResources().getConfiguration().orientation ==
Configuration.ORIENTATION_LANDSCAPE) {
        fragmentTransaction.add(R.id.fragment_container,
fragmentWithAllGoods);
        fragmentTransaction.add(R.id.fragment_main_order_container,
fragmentWithOrder);
    }
    fragmentTransaction.commit();
}

```

## MainActivity.java

@Override

public void **onStartOrderActivity()** {

Intent intent = new Intent(this, OrderActivity.class);

startActivity(intent);

}

@Override

public void **onRefreshOrderData()** {

if (getResources().getConfiguration().orientation ==

Configuration.ORIENTATION\_LANDSCAPE) {

fragmentWithOrder.refreshMyListView();

}

}

}

Интерфейсы в обратной связи:  
обработка в MainActivity события из  
fragmentWithAllGoods –  
нажатия на кнопку Show checked  
items

Интерфейсы в обратной связи:  
обработка в MainActivity события из  
fragmentWithAllGoods –  
активации/деактивации чекбоксов –  
закрывающаяся в обновлении  
данных корзины товаров (при  
гориз. ориентации)



## FragmentWithAllGoods.java

Интерфейс в обратной связи для отображения фрагментом общего количества выбранных товаров из адаптера

```
public class FragmentWithAllGoods extends Fragment implements  
OnChangeListener, View.OnClickListener,  
LoaderManager.LoaderCallbacks<Cursor>
```

Интерфейс, предоставляемый API Android, для фоновой загрузки данных из SQLite

```
private ListView listView;  
private WorkWithTableGoods tableGoods;  
private Cursor cursor;  
private View view_fragment, view_header, view_footer;  
private LayoutInflater inflater;  
private Button btnShow;  
private TextView tv_count;  
private GoodsAdapter goodsAdapter;  
private int count_checked_goods;
```

Реализация интерфейсов в обратной связи во фрагменте

```
private onStartOrderActivityListener mOnStartOrderActivityListener;  
private onRefreshOrderDataListener mOnRefreshOrderDataListener;
```

## FragmentWithAllGoods.java

@Override

```
public void onCreate(@Nullable Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setRetainInstance(true);  
}
```

Сохранение фрагмента  
при повороте экрана

@Override

```
public void onViewStateRestored(@Nullable Bundle  
savedInstanceState) {  
    super.onViewStateRestored(savedInstanceState);  
    if (savedInstanceState == null) {  
        count_checked_goods = 0;  
    } else {  
        count_checked_goods = savedInstanceState.getInt("counter", 0);  
    }  
};
```

Если не помогло  
сохранение  
фрагмента при  
повороте экрана ☹:  
восстанавливаем  
информацию после  
поворота экрана

## FragmentWithAllGoods.java

@Override

```
public void onSaveInstanceState(Bundle outState) {  
    super.onSaveInstanceState(outState);  
    outState.putInt("counter", count_checked_goods);  
}
```

Если не помогло  
сохранение фрагмента  
при повороте экрана ☹:  
сохраняем информацию  
перед поворотом экрана

@Override

```
public View onCreateView(LayoutInflater inflater, ViewGroup  
container, Bundle savedInstanceState) {  
    view_fragment = inflater.inflate(R.layout.fragment_with_all_goods,  
container, false);  
    initView();  
    initDB();  
    createMyListView();  
    return view_fragment;  
}
```

Самый важный метод во  
фрагменте: подключаем  
разметку, разворачиваем  
логику работы

## FragmentWithAllGoods.java

Инициализируем объекты от интерфейсов в методе `onActivityCreated` (на этом этапе ЖЦ фрагмента активности сообщает фрагменту о подключении; и можно наконец добраться до активности (`getActivity()`) с имплементированными интерфейсами

`@Override`

```
public void onActivityCreated(@Nullable Bundle savedInstanceState) {  
    super.onActivityCreated(savedInstanceState);  
    mOnStartOrderActivityListener =  
(OnStartOrderActivityListener) getActivity();  
    mOnRefreshOrderDataListener =  
(OnRefreshOrderDataListener) getActivity();  
}  
  
private void initView() {  
    listView = (ListView) view_fragment.findViewById(R.id.listView);  
}
```

## **FragmentWithAllGoods.java**

```
private void initDB() {  
    // открываем подключение к БД  
    tableGoods = new WorkWithTableGoods(getActivity());  
    tableGoods.open();  
    //tableGoods.delAll();  
    // tableGoods.write();  
}
```

## FragmentWithAllGoods.java

```
private void createMyListView() {  
    inflater = LayoutInflater.from(getActivity());  
    view_header = inflater.inflate(R.layout.header_mygoods, null);  
    view_footer = inflater.inflate(R.layout.footer_mygoods, null);  
    btnShow = (Button) view_footer.findViewById(R.id.btnShow);  
    btnShow.setOnClickListener(this);  
    tv_count = (TextView) view_footer.findViewById(R.id.tv_count);  
    tv_count.setText("Count of goods = " + count_checked_goods + "");  
  
    if (getActivity().getResources().getConfiguration().orientation ==  
Configuration.ORIENTATION_LANDSCAPE) {  
        btnShow.setVisibility(View.INVISIBLE);  
    }  
    getActivity().getSupportFragmentManager().initLoader(0, null, this);  
}
```

Кнопка Show checked  
items становится  
невидимой в  
горизонтальной  
ориентации

Запускаем загрузчик в  
фоновом потоке

## FragmentWithAllGoods.java

@Override

```
public void onDestroy() {  
    super.onDestroy();  
    tableGoods.close();  
}
```

Закрываем поток работы с БД

@Override

```
public void onClick(View v) {  
    mOnStartOrderActivityListener.onStartOrderActivity();  
}
```

Нажатие кнопки Show checked items приводит к запуску в MainActivity метода onStartOrderActivity()

## FragmentWithAllGoods.java

@Override

```
public void onDataChange(int id, String name, int price, int count) {  
    tableGoods.update(id, name, price, count);  
    getActivity().getSupportLoaderManager().restartLoader(0, null,  
this);
```

Выгружаем заново  
обновленные данные из  
БД в фоновом режиме

```
    if (count==0) {  
        count_checked_goods--;  
    }  
    else {  
        count_checked_goods++;  
    }
```

```
    tv_count.setText("Count of goods = " + count_checked_goods + "");
```

Активация/деактивация  
чекбоксов приводит к  
запуску в MainActivity  
метода  
onRefreshOrderData()

```
mOnRefreshOrderDataListener.onRefreshOrderData();
```

```
}
```



## FragmentWithAllGoods.java

Фоновый загрузчик  
информации из SQLite

@Override

```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {  
    return new MyCursorLoader(getActivity(), tableGoods);  
}
```

@Override

```
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {  
    if (goodsAdapter == null) {  
        goodsAdapter = new GoodsAdapter(getActivity(), data, this);  
        listView.addHeaderView(view_header);  
        listView.addFooterView(view_footer);  
        listView.setAdapter(goodsAdapter);  
    } else {  
        goodsAdapter.refreshCursor(data);  
    }  
}
```

Отдаем данные из SQLite  
в виде объекта Cursor  
data адаптеру, если он  
еще не создавался ранее,  
или обновляем data, если  
адаптер уже есть

## **FragmentWithAllGoods.java**

```
@Override  
public void onLoaderReset(Loader<Cursor> loader) {  
  
}
```

## FragmentWithAllGoods.java

Программная  
реализация фонового  
загрузчика

```
static class MyCursorLoader extends CursorLoader {  
    WorkWithTableGoods tableGoods;  
  
    public MyCursorLoader(Context context, WorkWithTableGoods  
tableGoods) {  
        super(context);  
        this.tableGoods = tableGoods;  
    }  
  
    @Override  
    public Cursor loadInBackground() {  
        Cursor cursor = tableGoods.getAllData();  
        return cursor;  
    }  
}
```

Ключевой метод  
загрузчика, выполняемый в  
фоновом потоке:  
назначаем ему выгрузку  
всех данных из БД в курсор

## **GoodsAdapter.java**

```
public class GoodsAdapter extends BaseAdapter implements  
CompoundButton.OnCheckedChangeListener {
```

```
    private Context context;  
    private LayoutInflater inflater;
```

```
    private Cursor cursor;  
    private int idColIndex;  
    private int nameColIndex;  
    private int priceColIndex;  
    private int countColIndex;
```

```
    private OnChangeListener onChangeListener;
```

## GoodsAdapter.java

```
public GoodsAdapter(Context context, Cursor cursor,  
OnChangeListener onChangeListener) {  
    this.context = context;  
    this.inflater = LayoutInflater.from(context);  
    this.onChangeListener = onChangeListener;  
    this.cursor = cursor;  
    idColIndex = cursor.getColumnIndex("id");  
    nameColIndex = cursor.getColumnIndex("name");  
    priceColIndex = cursor.getColumnIndex("price");  
    countColIndex = cursor.getColumnIndex("count");  
}
```

## **GoodsAdapter.java**

@Override

```
public int getCount() {  
    return cursor.getCount();  
}
```

@Override

```
public Cursor getItem(int i) {  
    cursor.moveToPosition(i);  
    return cursor;  
}
```

@Override

```
public long getItemId(int i) {  
    return i;  
}
```

## GoodsAdapter.java

@Override

```
public View getView(int position, View view, ViewGroup parent) {  
    if (view == null) {  
        view = inflater.inflate(R.layout.item_good, null);  
    }  
    cursor.moveToPosition(position);  
    ViewHolder vh = new ViewHolder();  
    vh.initViewHolder(view);  
    vh.tv_goodPrice.setText(cursor.getInt(priceColIndex)+"");  
    vh.tv_goodName.setText(cursor.getString(nameColIndex)+"");  
    if (cursor.getInt(countColIndex) == 0) {  
        vh.cb_good.setChecked(false);  
    } else vh.cb_good.setChecked(true);  
    vh.cb_good.setOnCheckedChangeListener(this);  
    vh.cb_good.setTag(position);  
    return view;  
}
```

## GoodsAdapter.java

@Override

```
public void onCheckedChanged(CompoundButton compoundButton,
boolean isChecked) {
    if (compoundButton.isShown()) {
        int i = (int) compoundButton.getTag();
        cursor.moveToPosition(i);
        int id = cursor.getInt(idColIndex);
        String name = cursor.getString(nameColIndex);
        int price = cursor.getInt(priceColIndex);
        int check = 0;
        if (isChecked){
            check = 1;
        }
        onChangeListener.onDataChanged(id, name, price, check);
        //notifyDataSetChanged(); // иначе – не работает ☹
    }
}
```



## **GoodsAdapter.java**

```
public void refreshCursor(Cursor data) {  
    cursor = data;  
}
```

## GoodsAdapter.java

```
public class ViewHolder {
```

```
    private TextView tv_goodPrice;  
    private TextView tv_goodName;  
    private CheckBox cb_good;
```

```
    public ViewHolder() {  
    }
```

```
    public void initViewHolder(View view) {  
        tv_goodPrice = (TextView) view.findViewById(R.id.tv_goodPrice);  
        tv_goodName = (TextView)  
view.findViewById(R.id.tv_goodName);  
        cb_good = (CheckBox) view.findViewById(R.id.cb_good);  
    }  
}  
}
```

Класс ViewHolder является внутренним по отношению к GoodsAdapter и позволяет вынести логику обнаружения графических элементов одного пункта списка

# Пояснение к примеру: как работает загрузчик CursorLoader

---

**CursorLoader** асинхронно читает данные из SQLite и возвращает Cursor.

Как его использовать в проекте:

1. В классе, реализующем выгрузку данных из БД, имплементируем интерфейс **LoaderManager.LoaderCallbacks<Cursor>** с вспомогательными методами для работы с загрузчиком.

2. Добавляем и переопределяем методы этого интерфейса:

**public Loader<Cursor> onCreateLoader(int id, Bundle args)** – создает объект загрузчика

**public void onLoadFinished(Loader<Cursor> loader, Cursor data)** – выполняет действия после окончания желаемого фонового процесса

**public void onLoaderReset(Loader<Cursor> loader)** – срабатывает, если что-то при выполнении фоновой задачи пошло не так...

---

## Пояснение к примеру: как работает загрузчик CursorLoader

@Override

```
public Loader<Cursor> onCreateLoader(int id, Bundle args) {  
    return new MyCursorLoader(getActivity(), tableGoods);  
}
```

@Override

```
public void onLoadFinished(Loader<Cursor> loader, Cursor data) {  
    if (goodsAdapter == null) {  
        goodsAdapter = new GoodsAdapter(getActivity(), data, this);  
        listView.addHeaderView(view_header);  
        listView.addFooterView(view_footer);  
        listView.setAdapter(goodsAdapter);  
    } else {  
        goodsAdapter.refreshCursor(data);  
    }  
}
```

@Override

```
public void onLoaderReset(Loader<Cursor> loader) {  
}
```

## Пояснение к примеру: как работает загрузчик CursorLoader

3. Создаем свой собственный внутренний класс с описанием фоновой задачи:

```
static class MyCursorLoader extends CursorLoader {  
    WorkWithTableGoods tableGoods;  
  
    public MyCursorLoader(Context context, WorkWithTableGoods  
tableGoods) {  
        super(context);  
        this.tableGoods = tableGoods;  
    }  
    @Override  
    public Cursor loadInBackground() {  
        Cursor cursor = tableGoods.getAllData();  
        return cursor;  
    }  
}
```

# Пояснение к примеру: как работает загрузчик CursorLoader

---

В этом внутреннем классе прописываем:

- наследование (**extends CursorLoader**)

- переменную класса — объект таблицы с методами (**WorkWithTableGoods tableGoods**)

- переопределяем конструктор, добавляя в него инициализацию переменной класса:

```
public MyCursorLoader(Context context, WorkWithTableGoods  
tableGoods) {  
    super(context);  
    this.tableGoods = tableGoods;  
}
```

- прописываем логику метода `loadInBackground()` — именно в нем и выполняется фоновый процесс, отделенный от UI-потока

---

# Пояснение к примеру: как работает загрузчик CursorLoader

---

```
@Override  
    public Cursor loadInBackground() {  
        Cursor cursor = tableGoods.getAllData();  
        return cursor;  
    }
```

## Пояснение к примеру: как работает загрузчик CursorLoader

---

4. В коде основного класса, в котором имплементирован интерфейс `LoaderManager.LoaderCallbacks<Cursor>` и прописан внутренний класс `MyCursorLoader` вызываем метод:

**`getActivity().getSupportLoaderManager().initLoader(0, null, this);`**

**`public Loader<Cursor> onCreateLoader(int id, Bundle args)`**

**`public Cursor loadInBackground()`**

**`public void onLoadFinished(Loader<Cursor> loader, Cursor data)`**



## Пояснение к примеру: как работает загрузчик CursorLoader

---

5. В коде основного класса, в котором имплементирован интерфейс `LoaderManager.LoaderCallbacks<Cursor>` и прописан внутренний класс `MyCursorLoader` при необходимости обновления данных в курсоре вызываем метод:

```
getActivity().getSupportLoaderManager().restartLoader(0, null, this);
```



The diagram illustrates the flow of data from the `restartLoader` method call to the `loadInBackground` method. A large red arrow points downwards from the `restartLoader` method to the `loadInBackground` method. A curved arrow points from the `loadInBackground` method back to the `restartLoader` method, indicating a return path or callback.

```
public Cursor loadInBackground()
```

```
public void onLoadFinished(Loader<Cursor> loader, Cursor data)
```