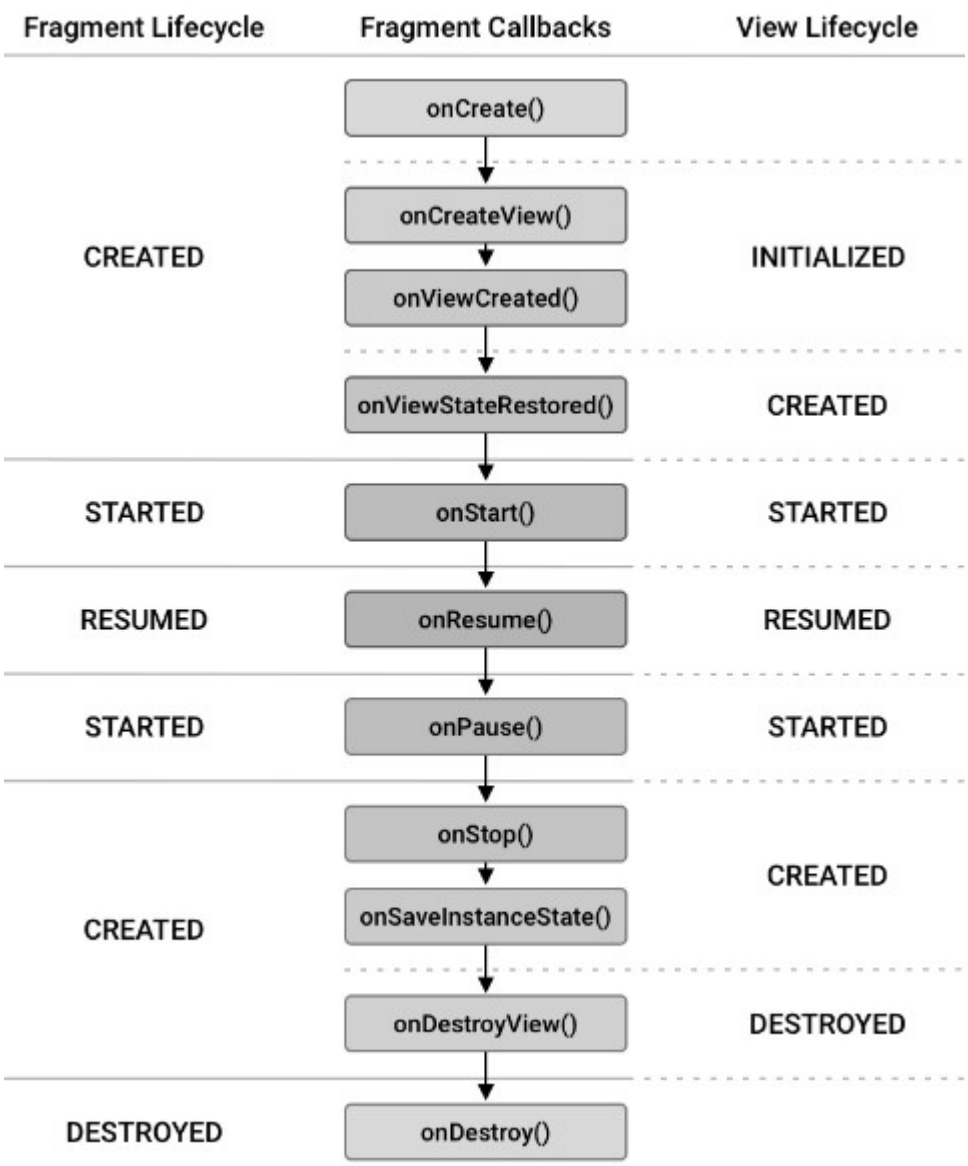
Лекция #26. Жизненный цикл фрагментов

Каждый класс фрагмента наследуется от базового класса Fragment и имеет свой жизненный цикл, состоящий из ряда этапов:



Каждый этап жизненного цикла описывается одной из констант перечисления **Lifecycle.State**:

- **INITIALIZED**
- **CREATED**
- **STARTED**
- **RESUMED**
- **DESTROYED**

Стоит отметить, что представление фрагмента (его визуальный интерфейс или View) имеет отдельный жизненный цикл.

- При создании фрагмент находится в состоянии **INITIALIZED**. Чтобы фрагмент прошел все остальные этапы жизненного цикла, фрагмент необходимо передать в объект **FragmentManager**, который далее определяет состояние фрагмента и переводит фрагмент из одного состояния в другое.

- Когда фрагмент добавляется в **FragmentManager** и прикрепляется к определенному классу **Activity**, у фрагмента вызывается метод **onAttach()**. Данный метод вызывается до всех остальных методов жизненного цикла. После этого фрагмент переходит в состояние **CREATED**

- **onCreate()**: происходит создание фрагмента. В этом методе мы можем получить ранее сохраненное состояние фрагмента через параметр метода **Bundle savedInstanceState**. (Если фрагмент создается первый раз, то этот объект имеет значение **null**) Этот метод вызывается после вызова соответствующего метода **onCreate()** у activity.

```
override fun onCreate(savedInstanceState: Bundle?)
```

- **onCreateView()**: фрагмент создает представление (**View** или визуальный интерфейс). В этом методе мы можем установить, какой именно изуальный интерфейс будет использовать фрагмент. При выполнении этого метода представление фрагмента переходит в состояние **INITIALIZED**. А сам фрагмент все еще находится в состоянии **CREATED**

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?)
```

  Первый параметр - объект **LayoutInflater** позволяет получить содержимое ресурса layout и передать его во фрагмент.
  Второй параметр - объект **ViewGroup** представляет контейнер, в которой будет загружаться фрагмент.

Третий параметр - объект **Bundle** представляет состояние фрагмента. (Если фрагмент загружается первый раз, то равен null)

На выходе метод возвращает созданное с помощью **LayoutInflater** представление в виде объекта **View** - собственно представление фрагмента

- **onViewCreated()**: вызывается после создания представления фрагмента.
```
override fun onViewCreated(view: View, savedInstanceState: Bundle?)
```
Первый параметр - объект **View** - представление фрагмента, которое было создано посредством метода onCreateView.
Второй параметр - объект **Bundle** представляет состояние фрагмента. (Если фрагмент загружается первый раз, то равен **null**)
- **onViewStateRestored()**: получает состояние представления фрагмента. После выполнения этого метода представление фрагмента переходит в состояние **CREATED**
```
override fun onViewStateRestored(savedInstanceState: Bundle?)
```

- **onStart()**: вызывается, когда фрагмент становится видимым и вместе с представлением переходит в состояние **STARTED**
```
override fun onStart()
```

- **onResume()**: вызывается, когда фрагмент становится активным, и пользователь может с ним взаимодействовать. При этом фрагмент и его представление переходят в состояние **RESUMED**
```
override fun onResume()
```

- **onPause()**: фрагмент продолжает оставаться видимым, но уже не активен и вместе с представлением переходит в состояние **STARTED**
```
override fun onResume()
```

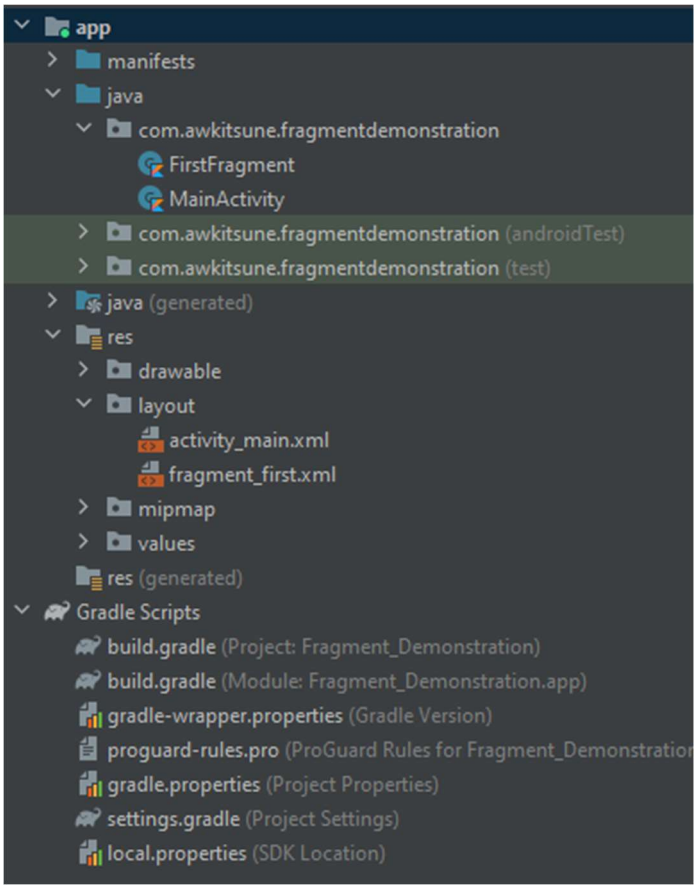
- **onStop()**: фрагмент больше не является видимым и вместе с представлением переходит в состояние **CREATED**
```
override fun onStop()
```
На этом этапе жизненного цикла мы можем сохранить состояние фрагмента с помощью метода **onSaveInstanceState()**.
Однако стоит учитывать, что вызов этого метода зависит от версии **API**. До **API 28** onSaveInstanceState() вызывается **до onStop()**, а начиная **API 28 после onStop()**.

- **onDestroyView()**: уничтожается представление фрагмента. Представление переходит в состояние **DESTROYED**

- **onDestroy()**: окончательно уничтожение фрагмента - он также переходит в состояние **DESTROYED**

- Метод **onDetach()** вызывается, когда фрагмент удаляется из **FragmentManager** и открепляется от класса **Activity**. Этот метод вызывается после всех остальных методов жизненного цикла.

В коде класса фрагмента мы можем переопределить все или часть из этих методов. Например, пусть у нас будет определен тот же проект:



Класс фрагмента использует данный файл для установки представления, а также определяет методы для управления жизненным циклом, так переопределим же их во имя летающего макаронного монстра:

```
package com.awkitsune.fragmentdemonstration
package com.awkitsune.fragmentdemonstration

import android.content.Context
import android.os.Bundle
import android.util.Log
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
```

```
import android.widget.TextView
import java.util.*

class FirstFragment : Fragment() {
    private val TAG = "FirstFragment"

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_first, container, false)
    }

    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val updateButton: Button = view.findViewById(R.id.updateButton)
        val dateText: TextView = view.findViewById(R.id.textView)

        updateButton.setOnClickListener {
            val curDate: String = Date().toString()
            dateText.text = curDate
        }
        Log.d(TAG, "onViewCreated")
    }

    override fun onViewStateRestored(savedInstanceState: Bundle?) {
        super.onViewStateRestored(savedInstanceState)
        Log.d(TAG, "onViewStateRestored")
    }

    override fun onStart() {
        super.onStart()
        Log.d(TAG, "onStart")
    }

    override fun onAttach(context: Context) {
        super.onAttach(context)
        Log.d(TAG, "onAttach")
    }
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        Log.d(TAG, "onCreate")
    }

    override fun onResume() {
        super.onResume()
        Log.d(TAG, "onResume")
    }

    override fun onPause() {
        super.onPause()
        Log.d(TAG, "onPause")
    }

    override fun onStop() {
        super.onStop()
        Log.d(TAG, "onStop")
    }

    override fun onDestroyView() {
        super.onDestroyView()
        Log.d(TAG, "onDestroyView")
```

```
    }

    override fun onDestroy() {
        super.onDestroy()
        Log.d(TAG, "onDestroy")
    }

    override fun onDetach() {
        super.onDetach()
        Log.d(TAG, "onDetach")
    }
}
```

В отличие от прошлой статьи, где рассматривалось создание фрагмента, здесь фрагмент устанавливает представление в методе **onCreateView**. Для этого в метод **inflate()** объекта LayoutInflater передается идентификатор ресурса layout и контейнер - объект ViewGroup, в который будет загружаться фрагмент. В итоге метод inflate() возвращает созданное представление.

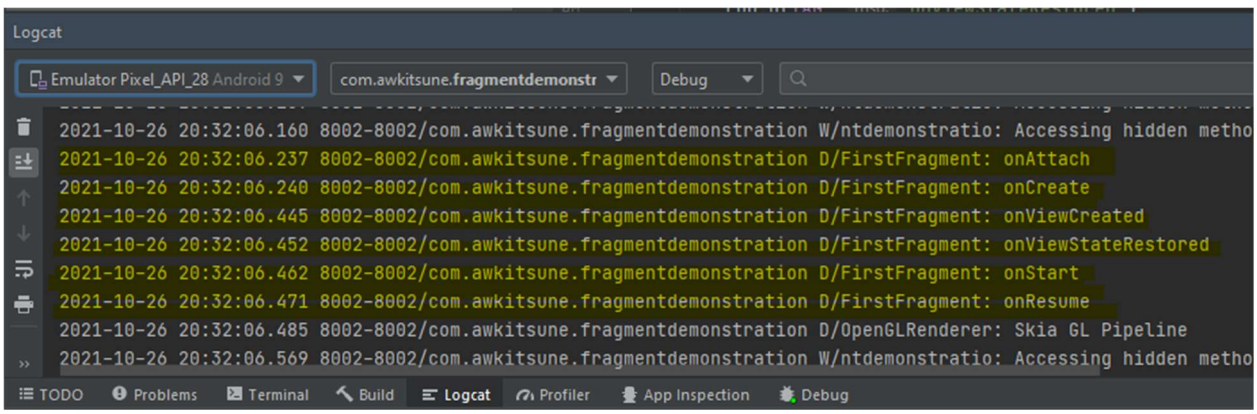
```
return inflater.inflate(R.layout.fragment_first, container, false)
```

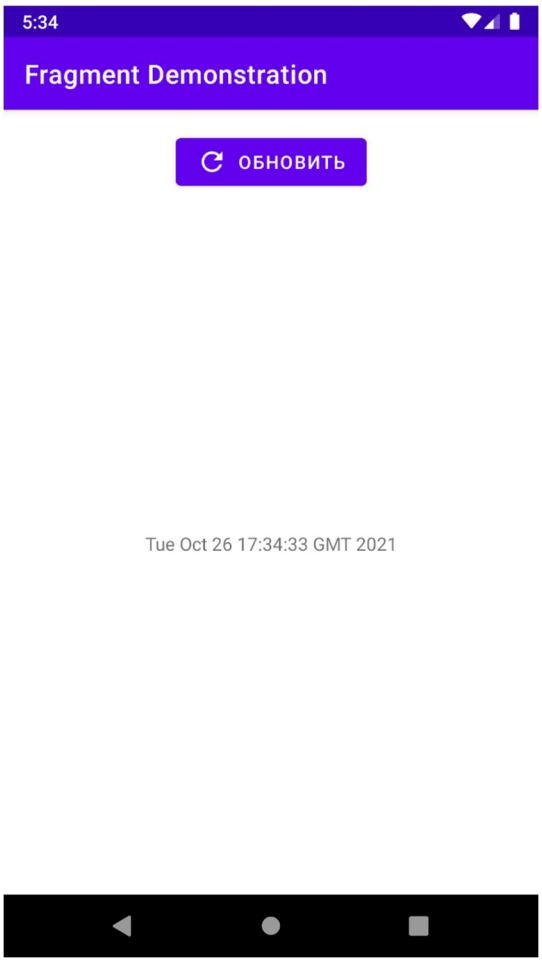
При выполнении метода **onViewCreated()** представление уже создано и оно передается в качестве первого параметра - объекта View, через который с помощью идентификаторов мы можем получить визуальные элементы - TextView и Button, которые определены в представлении.

Для остальных методов жизненного цикла установлено простое логгирование с помощью метода **Log.d()** с тегом **TAG** и **именем метода** в аргументе

Если мы запустим проект, то на экране устройства мы увидим визуальный интерфейс, определенный для фрагмента.

А в окне **Logcat** в **Android Studio** можно будет наблюдать логгирование методов жизненного цикла




5:34
Fragment Demonstration
ОБНОВИТЬ
Tue Oct 26 17:34:33 GMT 2021