

Практика 10: Работа с Fragments

В файл gradle подключим зависимости

```
dependencies {
    implementation(libs.androidx.navigation.fragment.ktx) //Реализует навигацию между фрагментами по API
    implementation(libs.androidx.lifecycle.viewmodel.ktx) //предоставляет данные для UI и умеет переживать
    // изменения конфигурации. ViewModel является посредником между репозиторием и UI. Кроме того, вы
    можете
    // использовать ViewModel для обмена данными между фрагментами.
    implementation(libs.androidx.activity.ktx) //Улучшает API для активностей
    implementation(libs.androidx.core.ktx) //Обеспечивает идиоматическую функциональность Kotlin для API
    // платформы Android
    implementation(libs.androidx.appcompat) //Позволяет свободно использовать темы, рисунки и элементы
    // управления пакета Material Design
    implementation(libs.material) //Используется для работы с дизайном
    implementation(libs.androidx.activity) //Добавление активностей
    implementation(libs.androidx.constraintlayout) //Адаптивная библиотека предназначена, чтобы сделать более
    // простым создание сложных пользовательских интерфейсов с различными устройствами экранов.
    testImplementation(libs.junit) //Библиотека для написания модульных тестов Java.
    androidTestImplementation(libs.androidx.junit) //Компонент JUnit Содержит расширение, которое позволяет
    запускать тест-методы на ui-потоке.
    androidTestImplementation(libs.androidx.espresso.core) //позволяет писать тесты для проверки пользовательского
    интерфейса в Android
}
```

Создаем пустой фрагмент и чистим его код

```
class BlankFragment : Fragment() {
    override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_blank, container, false)
    }

    companion object {
        @JvmStatic
        fun newInstance() = BlankFragment()
    }
}
```

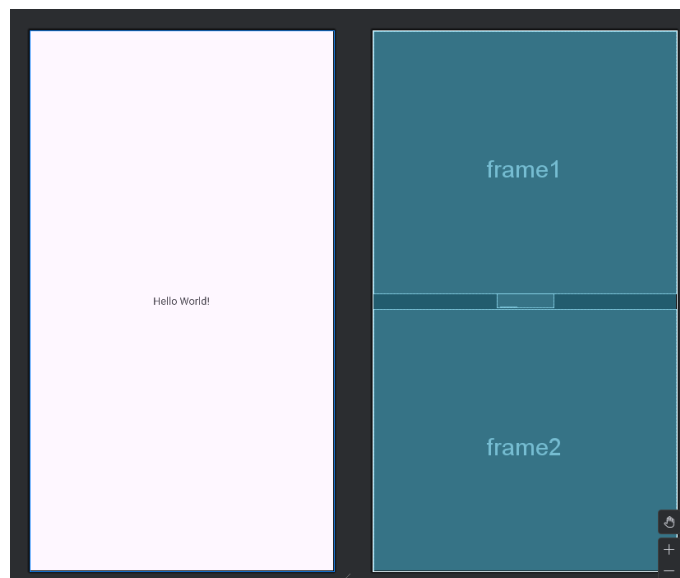
Далее вставляем код в activity_main

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <FrameLayout
        android:id="@+id/frame1"
        android:layout_width="409dp"
        android:layout_height="354dp"
```

```

    android:layout_marginStart="1dp"
    android:layout_marginTop="1dp"
    android:layout_marginBottom="1dp"
    app:layout_constraintBottom_toTopOf="@+id/textView"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
</FrameLayout>
<FrameLayout
    android:id="@+id/frame2"
    android:layout_width="409dp"
    android:layout_height="354dp"
    android:layout_marginTop="1dp"
    android:layout_marginEnd="1dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView">
</FrameLayout>
</androidx.constraintlayout.widget.ConstraintLayout>

```



Сделаем текст на фрагментах чуть заметнее (первого фрагмента)

```

<TextView
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:text="@string/hello_blank_fragment1"
    android:gravity="center"
    android:textSize="25sp"/>

```

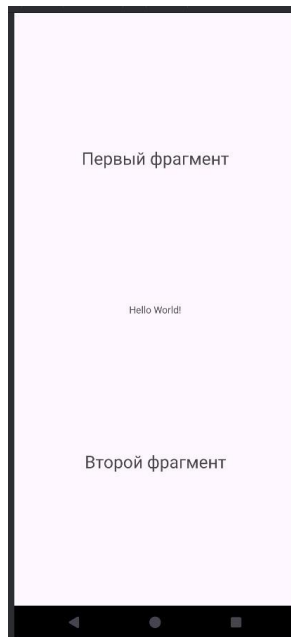
Код в классе MainActivity

```

//Функция задает фрагмент на выбранный frame
private fun openFragment(f: Fragment, idHolder: Int){
    supportFragmentManager
        .beginTransaction()
        .replace(idHolder,f)
        .commit()
}

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    binding = ActivityMainBinding.inflate(layoutInflater)
    setContentView(binding.root)
    //Установка 1-го и 2-го фрагментов
    openFragment(BlankFragment(), R.id.frame1)
    openFragment(BlankFragment2(), R.id.frame2)
}

```



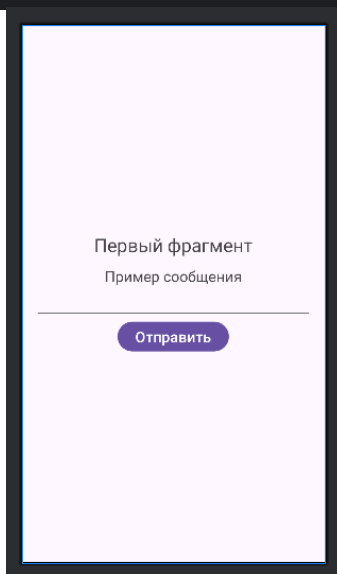
Задание: передать данные из одного фрагмента в другой и наоборот

В каждом фрагменте задаем такой макет

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".BlankFragment">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical"
        android:gravity="center"
        android:padding="18dp">
        <TextView
            android:id="@+id/tv"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/hello_blank_fragment1"
            android:textSize="25sp" />
        <TextView
            android:id="@+id/tvMessage"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="12dp"
            tools:text="Пример сообщения"
            android:textSize="20sp" />
        <EditText
            android:id="@+id/editText"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"/>
        <Button
            android:id="@+id/btnSend"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/buttonSend"
            android:textSize="20sp"/>
    </LinearLayout>
```

</FrameLayout>



Для передачи данных я использовал отдельный класс

```
open class DataModel: ViewModel() {  
    val messageForFragment1: MutableLiveData<String> by lazy {  
        MutableLiveData<String>()  
    }  
    val messageForFragment2: MutableLiveData<String> by lazy {  
        MutableLiveData<String>()  
    }  
}
```

В нем хранятся строки в `MutableLiveData` для возможности отслеживать изменения. `Lazy` делает инициализацию один раз.

Далее в классе `BlankFragment` добавим

```
//Объект модели данных  
private val dataModel: DataModel by activityViewModels()  
private lateinit var binding: FragmentBlankBinding  
  
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,  
    savedInstanceState: Bundle?): View? {  
    // Inflate the layout for this fragment  
    binding = FragmentBlankBinding.inflate(inflater)  
    return binding.root  
}  
  
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {  
    //Передача данных по нажатию на кнопку  
    binding.btnSend.setOnClickListener {  
        dataModel.messageForFragment2.value = binding.editText.text.toString()  
    }  
    //Получение данных, как только они будут отправлены из другого фрагмента  
    dataModel.messageForFragment1.observe(activity as LifecycleOwner) {  
        binding.tvMessage.text = it  
    }  
}
```

Тоже самое и для второго фрагмента, только нужно цифры поменять

```
private val dataModel: DataModel by activityViewModels()  
private lateinit var binding: FragmentBlank2Binding  
  
override fun onCreateView(inflater: LayoutInflater, container: ViewGroup?,
```

```

        savedInstanceState: Bundle?): View? {
// Inflate the layout for this fragment
binding = FragmentBlank2Binding.inflate((inflater))
return binding.root
}

override fun onCreateView(view: View, savedInstanceState: Bundle?) {
binding.btnSend.setOnClickListener {
dataModel.messageForFragment1.value = binding.editText.text.toString()
}
dataModel.messageForFragment2.observe(activity as LifecycleOwner) {
binding.tvMessage.text = it
}
}
}

```

Таким образом данные передаются по через текстовое поле после нажатия на кнопку

