САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ПЕТРА ВЕЛИКОГО

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

Отчет по лабораторной работе №3

«Lifecycle компоненты. Навигация в приложении»

Дисциплина

«Проектирование мобильных приложений»

Выполнил: Луняк Н. А.

Группа: 3530901/80202

Преподаватель: Алексюк А. О.

Санкт-Петербург

2020

СОДЕРЖАНИЕ

[Список иллюстраций 3](#_Toc52751943)

[Список Листингов 3](#_Toc52751944)

[1 Цели 5](#_Toc52751945)

[2 Задача 1. Обработка жизненного цикла с помощью Lifecycle-Aware компонентов 5](#_Toc52751946)

[2.1 Step 3 5](#_Toc52751947)

[2.2 Step 4 5](#_Toc52751948)

[2.3 Step 5 6](#_Toc52751949)

[2.4 Step 6 6](#_Toc52751950)

[3 Задача 2. Навигация (startActivityForResult) 8](#_Toc52751951)

[4 Задача 3. Навигация (флаги Intent/атрибуты Activity) 10](#_Toc52751952)

[5 Задача 4. Навигация (флаги Intent/атрибуты Activity) 11](#_Toc52751953)

[6 Задача 5. Навигация (Fragments, Navigation Graph) 15](#_Toc52751954)

[7 Выводы 20](#_Toc52751955)

# Список иллюстраций

[Рис. 2‑1 Ничего не работает 7](#_Toc52751956)

[Рис. 5‑1 Как оно выглядит 15](#_Toc52751957)

[Рис. 6‑1 Как оно выглядит 16](#_Toc52751958)

# Список Листингов

[Листинг 2‑1 Step 3 – Fix 1 5](#_Toc52751959)

[Листинг 2‑2 Step 3 – Fix 2 5](#_Toc52751960)

[Листинг 2‑3 Step 4 – Тык, 5](#_Toc52751961)

[Листинг 2‑4 Step 4 – Тык 5](#_Toc52751962)

[Листинг 2‑5 Step 4 – И тык! 5](#_Toc52751963)

[Листинг 2‑6 Step 4 – Логи 6](#_Toc52751964)

[Листинг 2‑7 Step 5 – Fix 1 6](#_Toc52751965)

[Листинг 2‑8 Step 5 – Fix 2 6](#_Toc52751966)

[Листинг 2‑9 Step 5 – Fix 3 6](#_Toc52751967)

[Листинг 2‑10 Step 6 – Fixing Manifest 6](#_Toc52751968)

[Листинг 2‑11 Step 6 – Fix 7](#_Toc52751969)

[Листинг 3‑1 Activity2 9](#_Toc52751970)

[Листинг 3‑2 Возврат из Activity3 в Activity1 9](#_Toc52751971)

[Листинг 3‑3 task\_2\_activity\_1 10](#_Toc52751972)

[Листинг 3‑4 task\_2\_drawer\_menu 10](#_Toc52751973)

[Листинг 4‑1 Activity3 11](#_Toc52751974)

[Листинг 5‑1 task\_4\_drawer\_manu 11](#_Toc52751975)

[Листинг 5‑2 Activity1 12](#_Toc52751976)

[Листинг 5‑3 ActivitySearch 13](#_Toc52751977)

[Листинг 5‑4 task\_4\_activity\_search 14](#_Toc52751978)

[Листинг 6‑1 task\_5\_activity\_main 16](#_Toc52751979)

[Листинг 6‑2 Fragment2 17](#_Toc52751980)

[Листинг 6‑3 task\_5\_view\_2 17](#_Toc52751981)

[Листинг 6‑4 task\_5\_navigation 18](#_Toc52751982)

[Листинг 6‑5 task\_5\_drawer\_menu 18](#_Toc52751983)

[Листинг 6‑6 ActivityMain 20](#_Toc52751984)

# Цели

* Ознакомиться с методом обработки жизненного цикла activity/fragment при помощи Lifecycle-Aware компонентов
* Изучить основные возможности навигации внутри приложения: создание новых activity, navigation graph

# Задача 1. Обработка жизненного цикла с помощью Lifecycle-Aware компонентов

## Step 3

В ChronoActivity3::subscribe() добавим регистрацию наблюдателя elapsedTimeObserver прямо под комментарием, который просит это сделать.

|  |
| --- |
| *//TODO: observe the ViewModel's elapsed time* mLiveDataTimerViewModel.getElapsedTime().observe(this, elapsedTimeObserver); |

Листинг 2‑1 Step 3 – Fix 1

И то же проделаем в LiveDataTimerViewModel, а то новые значения приходить не будут.

|  |
| --- |
| *// setValue() cannot be called from a background thread so post to main thread. //TODO post the new value with LiveData.postValue()* mElapsedTime.postValue(newValue); |

Листинг 2‑2 Step 3 – Fix 2

Теперь мы не будем видеть логов, когда активность не находится ни в STARTED, ни в RESUMED, но счетчик будет продолжать свой счет.

## Step 4

В общем, тут смысл в том, что мы имеет одного основного наблюдателя (MyLocationListener), который следит за юзером, но не сами его привязываем к сервису (LocationManager), а создаем дополнительного слушателя для наблюдения за lifecycle активности и используем его, чтобы либо привязать основного листенера, либо отвязать.

|  |
| --- |
| *//TODO: Add lifecycle observer* lifecycleOwner.getLifecycle().addObserver(this); |

Листинг 2‑3 Step 4 – Тык,

|  |
| --- |
| *//TODO: Call this on resume* @OnLifecycleEvent(Lifecycle.Event.*ON\_RESUME*) void addLocationListener() { |

Листинг 2‑4 Step 4 – Тык

|  |
| --- |
| *//TODO: Call this on pause* @OnLifecycleEvent(Lifecycle.Event.*ON\_PAUSE*) void removeLocationListener() { |

Листинг 2‑5 Step 4 – И тык!

Ну и в логах видим всякие сообщения.

|  |
| --- |
| 2020-09-29 19:24:54.429 5250-5250/com.example.android.codelabs.lifecycle D/BoundLocationMgr: Listener added  2020-09-29 19:25:07.503 5250-5250/com.example.android.codelabs.lifecycle D/BoundLocationMgr: Listener removed  2020-09-29 19:25:25.618 5250-5250/com.example.android.codelabs.lifecycle D/BoundLocationMgr: Listener added |

Листинг 2‑6 Step 4 – Логи

## Step 5

Получаем SeekBarViewModel по активности, а не по отдельным фрагментам – и чиним колбэки.

|  |
| --- |
| *// TODO: get ViewModel* mSeekBarViewModel = new ViewModelProvider(requireActivity()).get(SeekBarViewModel.class); |

Листинг 2‑7 Step 5 – Fix 1

|  |
| --- |
| *// TODO: Set the ViewModel's value when the change comes from the user.* mSeekBarViewModel.seekbarValue.setValue(progress); |

Листинг 2‑8 Step 5 – Fix 2

|  |
| --- |
| *// TODO: Update the SeekBar when the ViewModel is changed. // mSeekBarViewModel.seekbarValue.observe(...* mSeekBarViewModel.seekbarValue.observe(requireActivity(), new Observer<Integer>() {  @Override  public void onChanged(Integer progress) {  mSeekBar.setProgress(progress);  } }); |

Листинг 2‑9 Step 5 – Fix 3

## Step 6

А вот тут есть нюансы.

Сначала я пошел в манифест и везде вписал вот это, чтобы появились иконки запуска каждого отдельного шага. Я думал, это у меня из-за этого ничего не работает.

|  |
| --- |
| <category android:name="android.intent.category.LAUNCHER" /> |

Листинг 2‑10 Step 6 – Fixing Manifest

Но нет! Даже при нормальном выключении правильного приложения (step6\_solution) у меня все равно ничего не сохраняется, хотя я сделал все нормально.

|  |
| --- |
| public class SavedStateViewModel extends ViewModel {  *// TODO: Create constructor and use the LiveData from SavedStateHandle.* private static final String *NAME\_KEY* = "name";  private SavedStateHandle mState;   public SavedStateViewModel(SavedStateHandle handle) {  mState = handle;  }   *// Expose an immutable LiveData* LiveData<String> getName() {  return mState.getLiveData(*NAME\_KEY*);  }   void saveNewName(String newName) {  mState.set(*NAME\_KEY*, newName);  } } |

Листинг 2‑11 Step 6 – Fix

После перезапуска приложения я просто вижу дефолтный текст – и все.

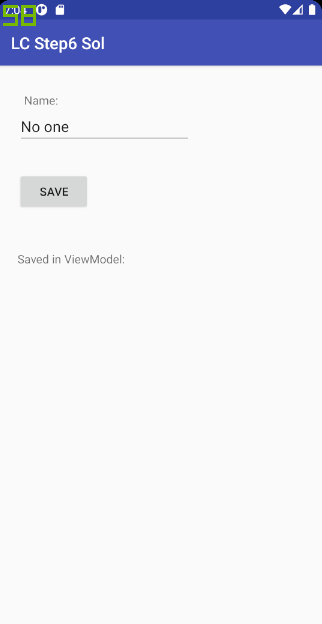


Рис. 2‑1 Ничего не работает

Если смотреть руками, то мы получаем null при вызове mSavedStateViewModel.getName().getValue(). А НЕТ СТОП!!!

Вот если руками убивать процесс через adb shell am kill com.example.android.codelabs.lifecycle, то как раз все восстанавливается правильно. То есть, видимо, при нормальном завершении приложения оно подчищает за собой все, поэтому при перезапуске после резкого отключения мы натыкаемся на старые данные (просто предполагаю).

// хорошо, что у меня стоит WSL, и я могу делать adb shell ps -A | bash -c "grep lifecycle" для тестов, а вот тем, кто работает на окнах без нее – [F].

# Задача 2. Навигация (startActivityForResult)

Так как я есть 7 вариант, то мне необходимо использовать Navigation Drawer для отображения About.

Тут выходит достаточно кода, поэтому я буду приводить лишь его части.

Активности все достаточно схожие, так что вот пример Activity2.

|  |
| --- |
| package com.example.androidlab3.task2  import android.content.Intent import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import androidx.core.view.GravityCompat import com.example.androidlab3.R import com.example.androidlab3.databinding.Task2Activity2Binding  class Activity2 : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)   val binding = Task2Activity2Binding.inflate(*layoutInflater*)  setContentView(binding.*root*)   binding.toFirst.setOnClickListener **{** finish()  **}** binding.toThird.setOnClickListener **{** startActivityForResult(  Intent(this, Activity3::class.*java*),  *LAUNCH\_INNER\_ACTIVITY* )  **}** binding.navView2.setNavigationItemSelectedListener **{** if (**it**.*itemId* == R.id.*open\_about*) {  startActivity(Intent(this, ActivityAbout::class.*java*))  binding.drawer2.closeDrawer(GravityCompat.*START*, true)  }   true  **}** }   override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {  super.onActivityResult(requestCode, resultCode, data)   if (requestCode == *LAUNCH\_INNER\_ACTIVITY*) {  if (resultCode == *RESULT\_OK*) {  val passThrough = data?.getIntExtra(*PASS\_THROUGH*, 0) ?: 0   if (passThrough > 0) {  setResult(*RESULT\_OK*, Intent().*apply* **{** putExtra(*PASS\_THROUGH*, passThrough - 1)  **}**)  finish()  }  }  }  } } |

Листинг 3‑1 Activity2

Ну то есть мы по нажатии запускаем следующую активность или закрываем текущую. Константа LAUNCH\_INNER\_ACTIVITY определена мною же в 0.. Просто результат мы получаем внутрь onActivityResult(), и его надо как-то идентифицировать.

Результат может содержать еще количество «пробрасываний». Это использует Activity3, когда из нее мы пытаемся вернуться сразу в Activity1, минуя Activity2. Третья активность тогда просто делает вот так.

|  |
| --- |
| setResult(*RESULT\_OK*, Intent().*apply* **{** putExtra(*PASS\_THROUGH*, 1) **}**)  finish() |

Листинг 3‑2 Возврат из Activity3 в Activity1

PASS\_THROUGH – моя константа.

Моно сразу видеть еще обработчик нажатия на кнопку в меню, который запускает новую активность About и закрывает drawer.

Layout’ы все примерно одинаковы.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <androidx.drawerlayout.widget.DrawerLayout  android:id="@+id/drawer\_1"  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:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:openDrawer="start">   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   <Button  android:id="@+id/to\_second"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/to\_second"  android:textSize="40sp"/>   </LinearLayout>   <com.google.android.material.navigation.NavigationView  android:id="@+id/nav\_view\_1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_gravity="start"  app:headerLayout="@layout/task\_2\_drawer\_header"  app:menu="@menu/task\_2\_drawer\_menu" />  </androidx.drawerlayout.widget.DrawerLayout> |

Листинг 3‑3 task\_2\_activity\_1

Атрибут openDrawer="start" просто позволяет в превью окошке видеть корректно отображающийся раскрытый drawer. А вот без android:layout\_gravity="start" drawer будет раскрыт сразу на весь экран и его нельзя будет убрать. Короче, работать не будет.

Заголовок drawer’а особо не интересен, а вот его меню выглядит вот так.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <menu xmlns:android="http://schemas.android.com/apk/res/android">   <item  android:id="@+id/open\_about"  android:title="@string/about" />  </menu> |

Листинг 3‑4 task\_2\_drawer\_menu

# Задача 3. Навигация (флаги Intent/атрибуты Activity)

Основное изменение – в Activity3.

|  |
| --- |
| package com.example.androidlab3.task3  import android.content.Intent import android.content.Intent.*FLAG\_ACTIVITY\_CLEAR\_TOP* import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import androidx.core.view.GravityCompat import com.example.androidlab3.R import com.example.androidlab3.databinding.Task2Activity3Binding import com.example.androidlab3.task2.ActivityAbout  class Activity3 : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)   val binding = Task2Activity3Binding.inflate(*layoutInflater*)  setContentView(binding.*root*)   binding.toFirst.setOnClickListener **{** startActivity(Intent(this, Activity1::class.*java*).*apply* **{** *flags* = *FLAG\_ACTIVITY\_CLEAR\_TOP* **}**)  **}** binding.toSecond.setOnClickListener **{** finish()  **}** binding.navView3.setNavigationItemSelectedListener **{** if (**it**.*itemId* == R.id.*open\_about*) {  startActivity(Intent(this, ActivityAbout::class.*java*))  binding.drawer3.closeDrawer(GravityCompat.*START*, true)  }   true  **}** } } |

Листинг 4‑1 Activity3

Теперь мы не закрываем текущую активность «с пробросом», а просим открыть новую с флагом, очищающим back stack для запрашиваемой активности, если она уже запущена. Так как мы достоверно знаем, что Activity1 обязательно к этому моменту была запущена, то мы не нарушаем ограничения текущего задания.

Правда, теперь открытие Activity1 изнутри Activity3 выглядит неконсистентно, но да ладно.

# Задача 4. Навигация (флаги Intent/атрибуты Activity)

Добавим новую кнопку в меню, ее обработку в Activity1 и саму новую активность.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <menu xmlns:android="http://schemas.android.com/apk/res/android">   <item  android:id="@+id/open\_about"  android:title="@string/about" />   <item  android:id="@+id/open\_search"  android:title="@string/search" />  </menu> |

Листинг 5‑1 task\_4\_drawer\_manu

|  |
| --- |
| package com.example.androidlab3.task4  import android.content.Intent import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import androidx.core.view.GravityCompat import com.example.androidlab3.R import com.example.androidlab3.databinding.Task4Activity1Binding import com.example.androidlab3.task2.ActivityAbout import com.example.androidlab3.task3.Activity2  class Activity1 : AppCompatActivity() {  override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)   val binding = Task4Activity1Binding.inflate(*layoutInflater*)  setContentView(binding.*root*)   binding.toSecond.setOnClickListener **{** startActivity(Intent(this, Activity2::class.*java*))  **}** binding.navView1.setNavigationItemSelectedListener **{** when (**it**.*itemId*) {  R.id.*open\_about* -> {  startActivity(Intent(this, ActivityAbout::class.*java*))  binding.drawer1.closeDrawer(GravityCompat.*START*, true)  }  R.id.*open\_search* -> {  startActivity(Intent(this, ActivitySearch::class.*java*))  binding.drawer1.closeDrawer(GravityCompat.*START*, true)  }  }   true  **}** } } |

Листинг 5‑2 Activity1

Новая активность будет имитировать поиск «как в гугле»: на одной странице будет отображаться поле ввода запроса и результаты прошлого запроса, а при отправке нового – эта активность будет «создана» заново. В кавычках – потому что используется флаг FLAG\_ACTIVITY\_SINGLE\_TOP, из-за которого новая будет создана, лишь если вверху back stack нет такой же активности – иначе она переиспользуется.

|  |
| --- |
| package com.example.androidlab3.task4  import android.content.Intent import android.content.Intent.*FLAG\_ACTIVITY\_SINGLE\_TOP* import android.os.Bundle import android.util.Log import androidx.appcompat.app.AppCompatActivity import com.example.androidlab3.R import com.example.androidlab3.databinding.Task4ActivitySearchBinding  class ActivitySearch : AppCompatActivity() {  companion object {  const val REQUEST = "REQUEST"  }   private var binding: Task4ActivitySearchBinding? = null   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)  Log.d("ActivitySearch", "Created")   val binding = Task4ActivitySearchBinding.inflate(*layoutInflater*)  this.binding = binding   setContentView(binding.*root*)  updateResult(*intent*)   binding.find.setOnClickListener **{** Log.d("ActivitySearch", "Starting")  startActivity(Intent(this, ActivitySearch::class.*java*).*apply* **{** *flags* = *FLAG\_ACTIVITY\_SINGLE\_TOP* putExtra(REQUEST, binding.request.*text*.toString())  **}**)  **}** binding.act1.setOnClickListener **{** startActivity(Intent(this, Activity1::class.*java*))  **}** }   private fun updateResult(intent: Intent) {  Log.d("ActivitySearch", "Updating")  val request = intent.*extras*?.getString(REQUEST, null)   request?.*let* **{** binding?.message?.*text* = getString(R.string.*nothing\_found\_for*, **it**)  Log.d("ActivitySearch", "Updated")  **}** }   override fun onNewIntent(intent: Intent?) {  super.onNewIntent(intent)  Log.d("ActivitySearch", "New Intent")   intent?.*let* **{** updateResult(**it**)  **}** } } |

Листинг 5‑3 ActivitySearch

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   <TextView  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/search"  android:textSize="23sp"  android:gravity="center"  android:padding="10dp" />   <EditText  android:id="@+id/request"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:inputType="text"  android:hint="@string/search"  android:autofillHints="@string/search"  android:padding="10dp"  android:textSize="23sp" />   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:gravity="center"  android:orientation="horizontal">   <Button  android:id="@+id/find"  android:layout\_gravity="center"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:paddingBottom="20dp"  android:paddingRight="40dp"  android:paddingLeft="40dp"  android:paddingTop="20dp"  android:textSize="23sp"  android:text="@string/find" />   <Button  android:id="@+id/act1"  android:layout\_gravity="center"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:paddingBottom="20dp"  android:paddingRight="40dp"  android:paddingLeft="40dp"  android:paddingTop="20dp"  android:textSize="23sp"  android:text="@string/activity1" />   </LinearLayout>   <TextView  android:id="@+id/message"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text=""  android:padding="10dp"  android:textColor="#ff0000"  android:textSize="23sp" />  </LinearLayout> |

Листинг 5‑4 task\_4\_activity\_search

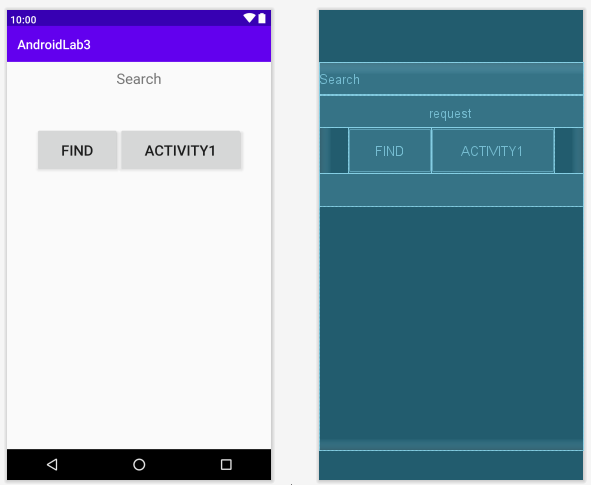


Рис. 5‑1 Как оно выглядит

При запуске наблюдаем, как у нас либо создается новая активность поверх других активностей, либо по-умному переиспользуется старая.

# Задача 5. Навигация (Fragments, Navigation Graph)

Ну, тут смысл в том, что мы делаем одну основную активность, у которой будет drawer и место, где переключаются разные фрагменты.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <androidx.drawerlayout.widget.DrawerLayout  android:id="@+id/drawer\_1"  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:layout\_width="match\_parent"  android:layout\_height="match\_parent"  tools:openDrawer="start">   <androidx.fragment.app.FragmentContainerView  android:name="androidx.navigation.fragment.NavHostFragment"  android:id="@+id/navigation\_host"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  app:navGraph="@navigation/task\_5\_navigation" />   <com.google.android.material.navigation.NavigationView  android:id="@+id/nav\_view\_1"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:layout\_gravity="start"  app:headerLayout="@layout/task\_2\_drawer\_header"  app:menu="@menu/task\_5\_drawer\_menu" />  </androidx.drawerlayout.widget.DrawerLayout> |

Листинг 6‑1 task\_5\_activity\_main

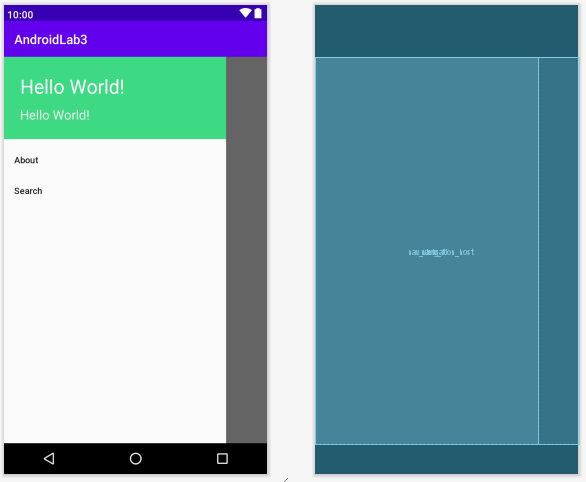


Рис. 6‑1 Как оно выглядит

При этом то, что раньше было активностями теперь выглядит просто и красиво.

|  |
| --- |
| package com.example.androidlab3.task5  import android.os.Bundle import android.view.LayoutInflater import android.view.View import android.view.ViewGroup import androidx.fragment.app.Fragment import androidx.navigation.findNavController import com.example.androidlab3.R import com.example.androidlab3.databinding.Task5View2Binding  class Fragment2 : Fragment() {  override fun onCreateView(  inflater: LayoutInflater,  container: ViewGroup?,  savedInstanceState: Bundle?  ): View? {  val binding = Task5View2Binding.inflate(*layoutInflater*)   binding.toFirst.setOnClickListener **{** *view*?.*findNavController*()?.navigate(R.id.*second\_to\_first*)  **}** binding.toThird.setOnClickListener **{** *view*?.*findNavController*()?.navigate(R.id.*second\_to\_third*)  **}** return binding.*root* } } |

Листинг 6‑2 Fragment2

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:orientation="vertical">   <Button  android:id="@+id/to\_first"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/to\_first"  android:textSize="40sp"/>   <Button  android:id="@+id/to\_third"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:text="@string/to\_third"  android:textSize="40sp"/>  </LinearLayout> |

Листинг 6‑3 task\_5\_view\_2

При этом у нас теперь есть файл, описывающий навигацию.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <navigation  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:app="http://schemas.android.com/apk/res-auto"  android:id="@+id/task\_5\_navigation"  app:startDestination="@id/task\_5\_destination\_1">   <fragment  android:id="@+id/task\_5\_destination\_1"  android:name="com.example.androidlab3.task5.Fragment1"  android:label="Activity 1">  <action  android:id="@+id/first\_to\_second"  app:destination="@id/task\_5\_destination\_2" />  </fragment>   <fragment  android:id="@+id/task\_5\_destination\_2"  android:name="com.example.androidlab3.task5.Fragment2"  android:label="Activity 2">  <action  android:id="@+id/second\_to\_first"  app:destination="@id/task\_5\_destination\_1" />  <action  android:id="@+id/second\_to\_third"  app:destination="@id/task\_5\_destination\_3" />  </fragment>   <fragment  android:id="@+id/task\_5\_destination\_3"  android:name="com.example.androidlab3.task5.Fragment3"  android:label="Activity 3">  <action  android:id="@+id/third\_to\_first"  app:destination="@id/task\_5\_destination\_1" />  <action  android:id="@+id/third\_to\_second"  app:destination="@id/task\_5\_destination\_2" />  </fragment>   <fragment  android:id="@+id/task\_5\_destination\_search"  android:name="com.example.androidlab3.task5.FragmentSearch"  android:label="Search">  <action  android:id="@+id/search\_to\_first"  app:destination="@id/task\_5\_destination\_1" />  </fragment>  </navigation> |

Листинг 6‑4 task\_5\_navigation

А меню drawer’а приняло следующий вид.

|  |
| --- |
| *<?*xml version="1.0" encoding="utf-8"*?>* <menu xmlns:android="http://schemas.android.com/apk/res/android">   <item  android:id="@+id/open\_about"  android:title="@string/about" />   <item  android:id="@id/task\_5\_destination\_search"  android:title="@string/search" />  </menu> |

Листинг 6‑5 task\_5\_drawer\_menu

То есть, кнопка для view поиска ссылается точно на id, указанный в навигации, что приведет к открытию фрагмента, а не активности. При таком подходе нельзя реализовать функционал аналогичный флагу FLAG\_ACTIVITY\_SINGLE\_TOP из прошлого пункта.

|  |
| --- |
| package com.example.androidlab3.task5  import android.content.Intent import android.os.Bundle import androidx.appcompat.app.AppCompatActivity import androidx.core.view.GravityCompat import androidx.navigation.findNavController import androidx.navigation.fragment.NavHostFragment import androidx.navigation.ui.\* import com.example.androidlab3.R import com.example.androidlab3.databinding.Task5ActivityMainBinding import com.example.androidlab3.task2.ActivityAbout  class ActivityMain : AppCompatActivity() {  private lateinit var appBarConfiguration: AppBarConfiguration   override fun onCreate(savedInstanceState: Bundle?) {  super.onCreate(savedInstanceState)   val binding = Task5ActivityMainBinding.inflate(*layoutInflater*)  setContentView(binding.*root*)   val navHostFragment =  *supportFragmentManager*.findFragmentById(R.id.*navigation\_host*) as NavHostFragment  val navController = navHostFragment.*navController   // configuration that knows that we're using  // a drawer and the appbar should have a button  // for opening it* appBarConfiguration = *AppBarConfiguration*(navController.*graph*, binding.drawer1)  *// synchronizes the navController with the appbar  setupActionBarWithNavController*(navController, appBarConfiguration)  *// synchronizes the navController with the drawer menu* binding.navView1.*setupWithNavController*(navController)   binding.navView1.setNavigationItemSelectedListener **{** when (**it**.*itemId*) {  R.id.*open\_about* -> {  startActivity(Intent(this, ActivityAbout::class.*java*))  binding.drawer1.closeDrawer(GravityCompat.*START*, true)  true  }  else -> {  *// navigates to the destination described  // by the id of this menu item via the  // menu asset* val that = **it**.*onNavDestinationSelected*(navController)  binding.drawer1.closeDrawer(GravityCompat.*START*, true)  that  }  }  **}** }   override fun onSupportNavigateUp(): Boolean {  *// appbar upper left button won't work  // without this code* val navController = *findNavController*(R.id.*navigation\_host*)  return navController.*navigateUp*(appBarConfiguration)  || super.onSupportNavigateUp()  } } |

Листинг 6‑6 ActivityMain

В коде написано, что зачем надо, поэтому не буду повторяться тут.

# Выводы

Из всех вышеописанных вариантом мне самым адекватным кажется последний. Во-первых, back stack автоматически обрабатывается. Во-вторых, мы пользуемся фрагментами, а их легко переиспользовать где-нибудь и комбинировать с чем-либо. В-третьих, тут есть умный менеджмент верхней левой кнопки в appbar’е. В-четвертых, в процессе решения задания 2 я копался полдня в доках и сделал в итоге его сразу, и только потом понял, что так делать не надо было, когда я формально дошел до последнего пункта, поэтому от всех предыдущих пунктов у меня остался неприятный осадочек. Мне действительно неясно, зачем была нужна вся эта возня с DrawerLayout без навигации, это просто странно.